Д. Ланде, В. Фурашев, Ю. Даник, Л. Страшной

Інтелектуальна система парламентського контролю

Від семантичного аналізу до інтелектуальної координації в державному управлінні з використанням штучного інтелекту

Державна наукова установа
«Інститут інформації, безпеки і права
Національної академії правових наук України»
Київ
2025

ДЕРЖАВНА НАУКОВА УСТАНОВА
«ІНСТИТУТ ІНФОРМАЦІЇ, БЕЗПЕКИ І ПРАВА
НАЦІОНАЛЬНОЇ АКАДЕМІЇ ПРАВОВИХ НАУК УКРАЇНИ»

Д. Ланде, В. Фурашев, Ю. Даник, Л. Страшной

ІНТЕЛЕКТУАЛЬНА СИСТЕМА ПАРЛАМЕНТСЬКОГО КОНТРОЛЮ

Від семантичного аналізу до інтелектуальної координації в державному управлінні з використанням штучного інтелекту

Київ
2025

УДК 004.8-378:342.53

Рекомендовано до друку
Вченою радою Державної наукової установи «Інститут інформації, безпеки і права Національної академії правових наук України» (протокол від 29 вересня 2025 р. № 8)

Рецензенти:

Довгань О.Д. – доктор юридичних наук, професор, Заслужений діяч науки і техніки України

Субач І.Ю. – доктор технічних наук, професор

Інтелектуальна система парламентського контролю : Від семантичного аналізу до інтелектуальної координації в державному управлінні з використанням штучного інтелекту : Монографія / Ланде Д.В., Фурашев В.М., Даник Ю.Г., Страшной Л.Л. – Київ: ТОВ «Інжиніринг», 2025. – 278 с.

Монографію присвячено розробці нової парадигми парламентського контролю в умовах цифрової трансформації та глобальних викликів. Пропонується семантична систему інтелектуального контролю, заснована на безкодовому програмуванні через великі мовні моделі, яка переходить від простої генерації тексту до логічної формалізації, верифікації та виконання контрольних запитів. У роботі інтегровано досягнення штучного інтелекту, юриспруденції, інформаційних технологій, конфліктології, семантичного інжинірингу та кібербезпеки, що дає змогу формувати динамічні семантичні мережі знань, прогнозувати наслідки політичних рішень та виявляти конфлікти, спричинені застосуванням ШІ.

Видання розраховане на фахівців, науково-педагогічних працівників, аспірантів, докторантів, студентів закладів вищої освіти, представників державних органів та органів місцевого самоврядування, установ та організацій.

ISBN 978-617-8180-03-4

Зміст

Вступ

Тенденції різної спрямованості їх наслідків та швидкості і повноти їх реалізації у реальне життя, які спостерігаються наразі не лише в Україні, а і у світі в цілому, вимагають невідворотність змін парадигм функціонування держав та їх окремих ключових інституцій, соціумів та їх окремих сегментів, територіальних поділів держав, різних галузей економіки та бізнесу, а також інших сфер життєдіяльності людей, що, у свою чергу, вимагають суттєвого поширення та розвитку існуючих систем контролю процесів у будь-якої країни світу. Природно, що це прискорить також трансформацію структури та складу системи суспільних відносин, їх змісту, а, іноді, до появи нових за змістом груп суспільних відносин; перехід до нових інноваційних моделей публічного управління та регулювання всіма формами життєдіяльності держави та суспільства що, це, в свою чергу, суттєво підвищують роль публічного та парламентського контролю.

Як відоме, будь-яка оцінка базується на інформації яку має тій, хто робить цю оцінку, тобто основними інструментами будь-якого виду контролю, у тому числі й парламентського, у процесі застосування інформаційних технологій, є різні бази даних, електронні таблиці, системи обліку та моніторингу, інтернет-ресурси та електронні звіти. Інформаційні технології парламентського контролю включають в себе використання різноманітних засобів та програмне забезпечення для збору, аналізу та інтерпретації даних, пов'язаних з діяльністю уряду та інших органів державної влади, а також досягнення наступних показників:

– **Ефективність контролю**. Систематичний підхід до збору та обробки даних дозволяє парламентаріям здійснювати контроль ефективніше та швидше. Дані, які збираються, повинні бути організовані в логічні групи та підготовлені для аналізу. Це дозволяє швидше знайти потрібну інформацію та зрозуміти її суть, дозволяє бачити всю картину здійснюваного контролю та зрозуміти, як різні частини процесу взаємодіють між собою, які елементи контролю працюють ефективно, а які потребують покращення.

– **Об’єктивність та достовірність**. Систематична обробка даних дозволяє забезпечити їх достовірність та об’єктивність. Система повинна забезпечувати точність та правильність даних, які збираються, а також захищати їх від можливої зміни чи впливу з боку третіх осіб.

Тим самим системність даних при парламентському контролі є важливим елементом, оскільки дозволяє, спираючись на сучасні можливості досягнень науково-технічного прогресу, забезпечити більше глибокій та значно розширений комплексний аналіз та оцінку різних аспектів діяльності органів влади, що, в свою чергу, забезпечує прийняття об’єктивних та науково обґрунтованих рішень.

Сучасна автоматизована обробка даних потрібних для всебічної оцінки процесів, які оцінюються, є дуже складною з різних причин, а саме:

– ***різноманітність даних***. В оцінюваній сфері може бути багато різних типів даних, таких як текстові документи, електронні таблиці, бази даних, відео- та аудіо записи тощо. Всі ці дані можуть бути потрібні для вирішення конкретних питань, і обробка їх усіх може бути дуже складною;

– ***низька стандартизація***. Сфера, що розглядається, може мати різні стандарти для організації та представлення даних, що може ускладнити автоматизовану обробку даних. Наприклад, формати документів, способи кодування даних тощо можуть різнитися в залежності від країни або регіону;

– ***конфіденційність даних***. У сфері, що розглядається, можуть бути конфіденційні дані, такі як особиста інформація про клієнтів, розгляді судових процесів тощо. Забезпечення конфіденційності даних може ускладнити автоматизовану обробку даних;

– ***потреба у точності***. У правовій галузі потрібна висока точність в обробці даних. Навіть невеликі помилки можуть мати серйозні наслідки для прийняття рішень;

– ***великий обсяг даних.*** Як правило, у правовій галузі може бути великий обсяг даних, особливо великі організації або урядові структури. Обробка таких великих обсягів даних може бути важкою та ресурсомісткою задачею.

Таким чином, дані мають специфічну структуру та вимагають певного рівня точності та нормативності, що ускладнює їх автоматичну обробку. Крім того, існує проблема недостатньої якості та стандартизації даних в правовій сфері. Для боротьби зі складністю автоматизованої обробки даних необхідно використовувати спеціалізовані програмні засоби та технології, які дозволяють проводити аналіз та обробку даних відповідно до нормативних вимог. Також є доцільним створення єдиних стандартів та форматів для зберігання та передачі даних в правовій сфері, що сприятиме їх більш ефективній автоматизованій обробці.

– **Нормативність.** Нормативність даних означає, що дані, що використовуються для парламентського контролю, повинні відповідати нормативним вимогам, тобто вони повинні бути отримані і оброблені відповідно до законодавства. Це важливо для забезпечення їх правової обґрунтованості та прийняття законних рішень на основі цих даних.

Дані, що використовуються для парламентського контролю, можуть бути отримані різними способами, такими як запити на інформацію, звіти, аудити, аналізи тощо. Однак, незалежно від джерела, їх отримання та оброблення повинно відбуватися відповідно до вимог законодавства та принципів правової держави. Крім того, нормативність даних також передбачає, що дані повинні бути достовірними та мають бути підтверджені документально або іншими відповідними доказами. Такі дані забезпечують парламентаріїв можливістю приймати обґрунтовані та об'єктивні рішення на основі фактів та аналізів, що сприяє підвищенню ефективності та забезпеченню здійснення контролю відповідно до законодавства.

– **Визначеність.** Дані повинні бути точно визначені, оскільки невідомість або нечіткість можуть призвести до неправильних рішень та негативних наслідків. Визначеність даних означає, що дані, що використовуються для парламентського контролю, повинні бути чітко визначені та конкретні. Це дозволяє уникнути недостовірних та необ'єктивних даних, які можуть призвести до неправильних рішень. Визначеність даних також забезпечує зрозумілість та доступність інформації для парламентаріїв, що здійснюють контроль. Це дозволяє їм зрозуміти суть інформації та легше визначити, як вона може бути використана для прийняття рішень. Визначеність даних допомагає парламентаріям бути конкретними та цілеспрямованими у своїх запитах на інформацію та забезпечує достовірність і об'єктивність інформації, що використовується для контролю.

– **Системність.** Дані, які підлягають подальшої обробці, повинні бути системними, оскільки вони взаємодіють між собою та складаються з різних елементів, які мають свої відносини та залежності. Системність даних при парламентському контролі означає, що дані повинні бути організовані в систему, яка забезпечує можливість їх збору, зберігання, обробки та аналізу в масштабах, що необхідні для здійснення ефективного контролю. У контексті парламентського контролю системність даних відіграє важливу роль, оскільки дозволяє налагодити логічний та послідовний аналіз даних, отриманих з різних джерел. Наявність системної структури даних дозволяє ефективніше здійснювати парламентський контроль, оскільки дозволяє зіставляти та аналізувати інформацію з різних джерел.

Важливими принципами забезпечення здійснення парламентського контролю в умовах децентралізації влади та цифровизації є наступні базові принципи:

– **Нормативна логіка**. Структура даних повинна мати має нормативну логіку, тобто вона повинна дотримуватися правил та законів, які визначають взаємозв'язки між різними елементами. Нормативна логіка має велике значення при парламентському контролі, оскільки це допомагає забезпечити дотримання норм та правил у законотворчій діяльності. В контексті парламентського контролю, нормативна логіка, що забезпечує консистентність та узгодженість законотворчої діяльності зі стандартами та нормами, допомагає розуміти та інтерпретувати законодавчі акти, розробляти стратегії взаємодії з іншими учасниками законодавчого процесу, визначати правові наслідки різних варіантів дій та приймати рішення на основі чіткого розуміння норм та їх застосування.

Консистентність вимагає, щоб закони та інші нормативні акти не суперечили одне одному та не порушували конституційні права та свободи людини. Консистентність також передбачає відповідність рішень та дій державних органів загальному правовому порядку і законодавчим стандартам. Важливість консистентності полягає в тому, що вона допомагає забезпечити стабільність правової системи та виключити можливість виникнення

суперечностей та протиріч у законодавстві, що можуть призвести до несправедливості, правової нестабільності та невизначеності. Також консистентність сприяє захисту прав та інтересів людей, оскільки вона забезпечує прозорість та передбачуваність дій держави та її органів. Також важливим є розуміння основних принципів логіки та наук про право, що дозволяє правильно аналізувати та інтерпретувати дані та приймати обґрунтовані рішення.

– **Ієрархічність.** Структура даних може мати ієрархічну форму, оскільки законодавчі акти можуть бути розміщені на різних рівнях – від Конституції до звичайного закону, нормативно-правового акту. Ієрархічність даних в парламентському контролі означає, що дані систематизовані у вигляді ієрархії, тобто розташовані в логічному порядку від загального до деталей. Це допомагає вести звітність із застосування нормативних актів та ухвалених рішень в парламенті. Ієрархічна структура даних дозволяє легко організувати та зберігати дані, що стосуються законодавства та правової практики, забезпечує зручний та швидкий доступ до цих даних та сприяє аналізу законодавчої діяльності. Також ієрархічність даних дозволяє враховувати ієрархічну структуру законодавства та відповідних документів, що сприяє їх зрозумінню та ефективному застосуванню. Крім того, ієрархічна структура даних забезпечує зручність при веденні звітності та контролі за дотриманням законодавства та відповідних процедур, що робить її важливим елементом парламентського контролю.

– **Формальність.** Структура даних повинна бути формальною, оскільки вона має відповідати певним формальним вимогам, які забезпечують її правову силу та вплив на поведінку людей. Відповідність даних певним формальним вимогам є дуже важливим аспектом у парламентському контролі. Це означає, що дані повинні бути зібрані, організовані та представлені у форматі, який відповідає стандартам та вимогам, встановленим законодавством.

Водночас, і це важливо зазначити, що відповідність формальним вимогам не завжди гарантує якість даних та їхню правильність. Тому, крім відповідності формальним вимогам, також необхідно забезпечити повноту, якість та достовірність даних.

– **Абстрактність.** Структура даних може бути абстрактною, оскільки вона не завжди пов'язана з конкретними фактами чи обставинами, а може бути визначена загальними поняттями та категоріями. Абстрактність структури даних в парламентському контролі дозволяє створювати загальні моделі та правила для опису різних аспектів законодавства, що забезпечує їх універсальність та застосовність для різних випадків та обставин. Це допомагає забезпечити однаковий підхід до розгляду різних законопроектів та забезпечити консистентність законодавства в цілому. Так, створення загальних форматів для подання даних, що містяться в законопроектах, дозволяє автоматично аналізувати ці дані та порівнювати їх з іншими даними,

що забезпечує більш ефективний та об’єктивний аналіз та прийняття рішень. Крім того, абстрактність структури даних дозволяє підвищити рівень автоматизації процесів парламентського контролю, наприклад, створення автоматизованих систем для аналізу великих обсягів даних з метою виявлення тенденцій та прогнозування результатів рішень.

Також необхідно розуміти та враховувати при проведенні парламентського контролю що для формування уявлення про певний соціальний об'єкт необхідно враховувати інформацію про його прояви в різних сферах, таких як економічна, управлінська, психологічна та інші. У випадку правової інформації, що стосується цих об’єктів, вона може мати економічне, управлінське, психологічне значення тощо, але в той же час вона також розкриває правову значущість цих об’єктів. Отримання правової інформації про певний об'єкт вимагає не тільки артибутивно-кількісної трансформації відповідних даних, але й спостереження, тлумачення та фахового аналізу.

Також необхідно мати на увазі, що правова інформація призначена для відображення об’єктів, які можуть мати різні ознаки, що визначають їх здатність бути відображеними та уявлення про них. Ознаки можуть включати злочинні прояви, суб’єкти посягання, стадії вчинення, підстави та умови відповідальності, міри покарання та інше. У деяких законах визначається діяльність правоохоронних та інших державних органів щодо запобігання злочинним проявам. Більшість ознак об’єктів правової інформації не має чіткого зовнішнього вираження, тому їх потрібно визначати та з’ясовувати. Особливо це стосується інформації про якість (атрибутивна інформація), а не про кількість об’єктів чи їх ознак. Інформація також може деталізуватися, уточнюватися та змінюватися залежно від міри отримання відомостей про ці об’єкти.

Необхідно також ще раз підкреслити, що сучасна епоха глобальних викликів – від гібридних війн до цифрової трансформації, від швидкого розвитку штучного інтелекту до постійних змін у законодавстві вимагає принципово нового підходу до державного управління. Особливо це стосується парламентського контролю, одного з найважливіших механізмів демократичного нагляду, який має забезпечувати підзвітність влади, ефективність реалізації державної політики та захист прав громадян. У той же час, традиційні методи контролю, засновані на ручному аналізі, депутатських запитах та комітетських слуханнях, вже не в змозі впоратися з масштабом, швидкістю та складністю сучасних управлінських процесів. Потрібна нова парадигма – не просто автоматизація, а інтелектуальне підсилення парламентської діяльності через інтегровані системи, здатні аналізувати, формалізувати, моделювати та прогнозувати.

Для досягнення вищенаведеного дане дослідження пропонує таку парадигму – семантичну систему парламентського контролю, побудовану на основі безкодового програмування через великі мовні моделі (LLM). Вона є

логічним продовженням монографії[1], але вже не обмежується аналізом текстів, а переходить до логічної формалізації запитів, автоматизованої побудови виконуваних модулів та динамічного розвитку семантичних мереж знань.

Метою даної монографії є представлення закінченої архітектури інтелектуальної координації у сфері державного управління, де генеративний штучний інтелект (ГШІ) виступає не лише як інструмент генерації текстів, а як основа для побудови логічно цілісних, верифікованих і відтворюваних систем контролю. Ця праця є міждисциплінарною – вона знаходиться на перетині декількох наукових полів, охоплюючи:

– Штучний інтелект, як технологічну основу, зокрема LLM (ChatGPT, Claude, Gemini), які дозволяють працювати з природною мовою на рівні, що наближається до людського.

– Юриспруденцію, як предметну область, де аналіз законодавства, виявлення суперечностей, оцінка відповідності норм і визначення конфліктогенних положень є важливим завданнями.

– Інформаційні технології, як інструментальну базу: від Gephi та GraphViz до власних програмних розробок (SemanticCore, Dataset, ChronoTensor), які дозволяють візуалізувати, аналізувати та керувати знаннями.

– Парламентський контроль, як інституційну реальність, яка потребує не просто інформації, а інтелектуальної координації між аналізом, рішенням і дією.

– Конфліктологію, як наукову парадигму, яка дозволяє систематизувати не лише міжособистісні чи інституційні конфлікти, а й нові типи конфліктів, породжених штучним інтелектом (ШІ) – «чорний ящик», автономність, генерація змісту, відсутність відповідальності.

– Семантичний інжиніринг, як методологію перетворення текстів на структуровані мережі знань, які можуть розвиватися в часі та просторі.

– Кібербезпеку, як контекст, де аналіз кіберінцидентів, виявлення фейкової інформації та побудова семантичних мап загроз стають невід'ємною частиною державного контролю.

Таке поєднання дозволяє говорити не про черговий інструмент, а про нову наукову дисципліну – інтелектуальний парламентський контроль, заснований на інформаційному праві, семантичному нетворкінгу та логічній формалізації.

[1] Д.В. Ланде, В.М. Фурашев. Парламентський контроль із застосуванням генеративного штучного інтелекту : монографія / Ланде Д.В., Фурашев В.М. – Київ: ТОВ «Інжиніринг», 2023. – 202 с. ISBN 978-966-2344-82-0

У монографії [1] було показано, як генеративний штучний інтелект може використовуватися для аналізу правових документів, формування семантичних мереж, візуалізації зв'язків між нормами та підтримки прийняття рішень. Було продемонстровано, як інтеграція ChatGPT із Gephi дозволяє будувати причинно-наслідкові моделі, аналізувати звіти уряду та виявляти порушення.

Однак генеративні моделі мають принципові обмеження:

– галюцинації – вигадування фактів, норм, посилань;

– відсутність відтворюваності – одна і та ж модель може давати різні відповіді на один і той же запит;

– «чорний ящик» – відсутність сліду логічних міркувань;

– неможливість виконання – вивід залишається текстом, а не програмою.

Щоб подолати ці обмеження, було розроблено фреймворк безкодового програмування, який дозволяє фіксувати логічні примітиви у виводах LLM: умови, цикли, функції, переходи, мітки. Це дозволяє перетворити природну мову на виконувану логічну структуру, яку можна верифікувати, відтворювати та інтегрувати в системи контролю.

Цей підхід було детально описано в статті [2], а також у роботах [3, 4].

Центральним елементом системи є семантичний нетворкінг – процес побудови, розвитку та аналізу мереж знань на основі текстів законів, скарг, звітів, соціальних медіа. Цей підхід було детально розроблено в монографіях [5, 6] та серії статей [7, 8, 9].

---

[2] Ланде Д.В., Фурашев В.М. Застосування фреймворку безкодового програмування при вирішенні завдань парламентського контролю. Інформація і право, 2025. – N 2(53). – С. 88-103. DOI: https://doi.org/10.37750/2616-6798.2025.2(53).334055

[3] Dmytro Lande, Leonard Strashnoy. An Advanced No-Code Programming Framework for Complex Problems in LLM Environments. ResearchGate Preprint. DOI: 10.13140/ RG.2.2.25307.89129 (May, 2025). 14 pp.

[4] Dmytro Lande, Leonard Strashnoy. AgentFlow – No-Code Agent Framework Based on Logical Primitives. SSRN Preprint.: 5285664, DOI: 10.2139/ssrn.5285664 (Jun 22, 2025). 12 pp.

[5] Dmytro Lande, Leonard Strashnoy. GPT Semantic Networking: A Dream of the Semantic Web – The Time is Now. – Kyiv: Engineering, 2023. – 168 p. ISBN 978-966-2344-94-3, DOI: 10.5281/zenodo.14278893

[6] Lande D., Strashnoy L. Advanced Semantic Networking based on the large language models : Monograph. Kyiv: Engineering, 2025. 258 p. ISBN 978-617-8180-02-7

[7] Dmitry Lande, Leonard Strashnoy. Formation of a Network of Preferred Semantic Connections Based on Prompt Engineering. ResearchGate Preprint. DOI: 10.13140/ RG.2.2.30430.55367 (March 31, 2025).

[8] Dmitry Lande, Leonard Strashnoy. Semantic AI Framework for Prompt Engineering ResearchGate Preprint. DOI: 10.13140/RG.2.2.21116.24964 (Mart 9, 2025).

[9] Dmytro Lande, Elina Shnurko-Tabakova. Application of Large Language Models for Event Analysis and Entity Recognition in Cybersecurity. ResearchGate Preprint. DOI: 10.13140/RG.2.2.32331.20002 (Nov 23, 2024)

На відміну від статичних онтологій, семантичний нетворкінг є динамічним процесом: мережа змінюється в часі, виявляючи нові зв'язки, прогалини, конфлікти. Він ґрунтується на:

– промпт-керованому зондуванні LLM для вилучення сутностей і зв'язків;
– двонаправленому пошуку – від проблеми до норми і навпаки;
– причинно-наслідкових моделях для сценарного аналізу;
– візуалізації за допомогою Gephi[10], GraphViz[11], інших інструментів[12].

Це дозволяє не лише аналізувати, а й прогнозувати, моделювати та інтерактивно навігувати в правовому просторі.

Одним із наукових напрямків цієї роботи є введення та розгляд поняття конфліктології штучного інтелекту – нової галузі, яка вивчає специфічні форми соціально-правових конфліктів, викликаних унікальними властивостями ШІ: самонавчанням, автономністю, генерацією змісту, адаптивністю.

У статті[13] було систематизовано ці конфлікти за рівнями:

– міжособистісний – коли ШІ втручається у комунікацію між людьми;
– інституційний – коли ШІ створює конфлікт між органами влади;
– системний – коли ШІ порушує цілісність правової системи;
– міжнародний – коли різні підходи до регулювання ШІ призводять до протиріч між країнами.

Для виявлення таких конфліктів було розроблено систему структурованих промптів, кожен з яких містить логічні примітиви (умова, цикл, функція). Це дозволило перетворити природну мову на інструмент системного правового контролю.

Ще одним важливим елементом цього дослідження є агентська логіка – використання «рою віртуальних експертів», які моделюють поведінку фахівців у певних галузях права. Це не просто запити до ChatGPT, а формалізовані ролі (аналітик, контролер, інтерпретатор норм), які можуть взаємодіяти, обмінюватися даними, формувати колективні висновки. Цей

[10] Ken, Cherven (2015). Mastering Gephi Network Visualization. Packt Publishing, 378. ISBN-13: 978-1783987344
[11] Lambert M. Surhone, Mariam T. Tennoe, Susan F. Henssonow. Graphviz. VDM Publishing, 2010. – 108 p. ISBN-13: 9786131389023
[12] Ландэ Д.В., Субач І.Ю. Візуалізація та аналіз мережевих структур : навчальний посібник. – Київ : КПІ ім. Ігоря Сікорського, Вид-во "Політехніка", 2021. – 80 с. ISBN 978-966-2577-14-3
[13] «Виявлення та аналіз конфліктів у законодавстві, пов'язаних із застосуванням ШІ» (2025)

підхід було розроблено в серії робіт [14, 15, 16]. Хоча масштабне застосування оркестрації ще в розробці, сама концепція відкриває шлях до інтелектуальної координації, де рій агентів аналізує ситуацію, виявляє проблеми та пропонує рішення.

Усі описані підходи мають практичне втілення у вигляді комп'ютерних програм (SemanticCore[17], ChronoTensor[18], Dataset[19], Sentiment Expert Swarm AI[20]), підтверджених свідоцтвами про реєстрацію авторських прав, а також конкретних кейсів аналізу законопроектів, звітів уряду, скарг громадян. Монографія також враховує міжнародний досвід: системи на кшталт Lexis+ AI[21], LawGeex[22], Lex Machina[23], які використовуються у світі для аналізу судової практики, підготовки звітів, автоматизації юридичної роботи.

Запропонована авторами в Україні модель[24] має унікальну особливість – вона грунтується не лише на аналізі даних, але й на логічній формалізації правових норм, що робить її більш прозорою, верифікованою та підзвітною.

---

[14] Dmytro Lande, Leonard Strashnoy. Swarm of Virtual Experts in the Implementation of Semantic Networking. ResearchGate Preprint, October 2024. DOI: 10.13140/ RG.2.2.16686.11845

[15] D. Lande, I. Svoboda, L. Alekseichuk, L. Strashnoy. Methodology of a Swarm of Virtual Experts for Evaluating the Weight of Connections in Networks. Theoretical and Applied Cyber Security. Vol. 7 No. 2 (2024). DOI: 10.20535/tacs.2664-29132024.2.319946

[16] Lande Dmitry, Strashnoy Leonard. Orchestration of No-Code Agents in the AgentFlow Framework. SSRN Preprint: 5381820, DOI: 10.2139/ssrn.5381820 (Aug 15, 2025)

[17] Комп'ютерна програма аналізу семантичних зв'язків «SemanticCore». Ланде Д.В., Фегер А.П. Україна. Свідоцтво про реєстрацію авторського права на твір N 125039 від 24.04.2024.

[18] Комп'ютерна програма трансформер прогнозування "ChronoTensor" ("ChronoTensor") Ланде Д.В., Фегер А.П. Україна. Свідоцтво про реєстрацію авторського права на твір N 132218 від 19.12.2024.

[19] Комп'ютерна програма "Програмне забезпечення формування знань в базі знань інтелектуальної системи управління інформацією та подіями безпеки "Dataset" ("Dataset"). Клячко А.О., Пучков О.О., Субач І.Ю., Ланде Д.В., Микитюк А.В. Україна. Свідоцтво про реєстрацію авторського права на твір N 129468 від 28.08.2024.

[20] Комп'ютерна програма "Sentiment Expert Swarm AI" (SESAI). Рибак О.О., Пучков О.О., Ланде Д.В., Субач І.Ю., Микитюк А.В. Україна. Свідоцтво про реєстрацію авторського права на твір N 131976 від 09.12.2024

[21] Akiner T, Punuru J, Sharma S. Intent Classification and Dialogue Management for Lexis AI. Proceedings of the 7th Annual RELX Search Summit. 2023 Sep 21.

[22] Labin, S. and Segal, U., 2021. AI-driven contract review: A product development journey. In *Research Handbook on Big Data Law* (pp. 454-466). Edward Elgar Publishing.

[23] Mucahit Kucuc, C., 2024. The Use of Artificial Intelligence in the Legal System. *Digital L. Rev.*, *6*, p.24.

[24] Ланде Д.В., Фурашев В.М. Застосування фреймворку безкодового програмування при вирішенні завдань парламентського контролю. *Інформація і право*, 2025. – N 2(53). – С. 88-103. DOI: https://doi.org/10.37750/2616-6798.2025.2(53).334055

Монографія побудована за принципом від загального до конкретного, що відображено в її структурі:

У розділі 1 розглядається застосування штучного інтелекту у державному управлінні, зокрема його потенціал, контекст використання та фундаментальні обмеження, які обумовлюють необхідність переходу від генеративного аналізу до логічної формалізації користувацьких запитів (промптів).

У розділі 2 детально описується фреймворк безкодового програмування – ключовий інструмент, який дозволяє перетворювати природомовні промпти на структуровані запити до ШІ з чітко виділеними логічними примітивами (умови, цикли, функції), що є основою для створення виконуваних модулів.

У розділі 3 присвячено семантичному нетворкінгу – методології побудови, розвитку та аналізу динамічних мереж знань, які виникають на основі обробки правових, адміністративних та соціальних текстів за допомогою великих мовних моделей.

У розділі 4 розглядається концепція агентської логіки та «рою віртуальних експертів». Тут описується, як шляхом промпт-керованого програмування можна створювати спеціалізовані агенти (аналітик, контролер, інтерпретатор), які мають внутрішній стан і здатні до моделювання складної поведінки, хоча наразі їхнє масштабне застосування ще перебуває в стадії розробки.

У розділі 5 пропонуються методи сценарного аналізу, засновані на побудові та оптимізації причинно-наслідкових моделей. Цей підхід дозволяє не просто описувати проблеми, а прогнозувати наслідки змін у законодавстві та вибирати найефективніші шляхи реалізації реформ.

У розділі 6 наводяться конкретні практичні приклади застосування розробленої інтелектуальної системи в практиці парламентського контролю: від аналізу суперечностей у законопроектах та контролю за міжнародними зобов’язаннями до обробки масових скарг громадян та аналізу діяльності ключових державних органів.

У розділі 7 вводиться нова наукова парадигма – конфліктологія штучного інтелекту. Цей розділ систематизує унікальні конфлікти, що виникають через специфічні властивості ШІ («чорний ящик», автономність, генерація змісту), та пропонує методи їх виявлення та аналізу на різних рівнях – від міжособистісного до міжнародного.

У розділі 8 підводяться підсумки та визначаються стратегічні шляхи подальшого розвитку – від створення інтелектуальної координації між людиною та машиною до інтеграції системи з цифровими платформами парламенту, а також розглядаються ключові виклики, пов’язані з безпекою, етикою та якістю даних.

Ця праця виконується в рамках науково-дослідної роботи за темою «Організаційно-правові і технологічні засади розвитку системи нормативно-правової інформації та парламентського контролю в умовах цифровізації» (РК УкрІНТЕІ № 0121U100234), що проводилась Державною науковою установою «Інститут інформації, безпеки і права Національної академії правових наук України» та створенню базової основи розширення даного дослідження у напряму залучення штучного інтелекту для інформаційно-аналітичного забезпечення правотворчої діяльності, розвитку е-парламенту і баз даних правової інформації.

Розділ 1. Штучний інтелект у державному управлінні

Поява генеративних мовних, зокрема, великих мовних моделей відкрила нову еру в інформаційних технологіях, що торкнулася майже всіх сфер людської діяльності – від медицини та освіти до наукових досліджень, кібербезпеки та державного управління.

Так у вересні 2025 року в Україні відбулась у презентація ШІ-асистента "Дія.AI", який надаватиме державні послуги на порталі "Дія". Було представлено перший у світі ШІ-помічник, який надає державні послуги прямо в чаті. Асистент цілодобово допомагатиме громадянам знаходити потрібні послуги, перевірятиме дані в реєстрах та радитиме, як оформити документи. Вже на цей час доступний перший сервіс – отримання довідки про доходи[25].

У цих умовах парламентський контроль, як один із механізмів підзвітності влади, не може залишатися островом традиційних методів аналізу, заснованих на ручній обробці величезних масивів документів, звітів, законопроектів та звернень громадян. Сучасний обсяг інформації вимагає не просто автоматизації, а інтелектуального підсилення – переходу від аналізу до моделювання, від опису до прогнозування, від реакції – до проактивної координації.

Саме цей шлях – від текстової аналітики із застосуванням великих мовних моделей до структурованої системи контролю – є змістом цього розділу. Розглядається, як технології генеративного ШІ, які на початку використовувалися як інструменти для швидкого аналізу текстів, поступово трансформуються в основу для побудови логічно цілісних, формалізованих, відтворюваних систем правового контролю. Цей шлях проходить через визнання обмежень, спроб їх подолання, розробку нових фреймворків (наприклад, безкодового програмування), впровадження семантичного нетворкінгу, створення динамічних мереж знань. Але саме цей шлях дозволяє перейти від «розуміння тексту» до «керування процесом».

1.1. Революція генеративного штучного інтелекту: нова ера обробки природної мови

Ми живемо в епоху, коли штучний інтелект вийшов за межі наукової фантастики та став реальним інструментом, який змінює принципи роботи у всіх сферах. Одним із найважливіших проривів останніх років стала поява генеративних мовних моделей[26] (GPT, Claude, Gemini, Llama, Mistral тощо), здатних не лише розуміти природну мову, а й створювати зв'язний,

[25] https://uaprom.info/news/komanda-mintsyfry-prezentuvala-prezydentu-novi-proiekty-u-sferi-tsyfrovykh-posluh

[26] Chiarello, F., Giordano, V., Spada, I., Barandoni, S. and Fantoni, G., 2024. Future applications of generative large language models: A data-driven case study on ChatGPT. *Technovation*, *133*, p.103002. DOI: 10.1016/j.technovation.2024.103002

контекстно-орієнтований, логічно упорядкований текст на рівні, що часто не відрізняється від людського.

Це не просто автоматизація – це революція в обробці природної мови. На відміну від попередніх поколінь ШІ, які працювали зі структурованими даними, сучасні LLM працюють із неструктурованим текстом, який є основною формою представлення знань у правовій, політичній та адміністративній сферах. Закони, постанови, звіти, скарги, коментарі – усе це є природною мовою, і тепер її можна не просто читати, а аналізувати, узагальнювати, порівнювати, моделювати[27].

Уже сьогодні генеративний ШІ використовується:

– у медицині – для діагностики за симптомами, аналізу медичних карт, підготовки звітів;

– у освіті – для створення навчальних матеріалів, індивідуалізації навчання, автоматизації перевірки завдань;

– у юриспруденції – для аналізу судових рішень, підготовки правових висновків, пошуку прецедентів;

– у науці – для написання статей, аналізу літератури, формулювання гіпотез;

– у кібербезпеці – для аналізу інцидентів, виявлення фейків, побудови мап загроз.

Наприклад, в США системи на кшталт Lexis+ AI чи LawGeex дозволяють юристам за кілька хвилин проаналізувати контракт на наявність ризиків, а в Ізраїлі та Чехії розробляються цифрові двійники законодавства [28, 29] які моделюють наслідки прийняття нових норм.

У цих умовах парламентський контроль не може залишатися островом «аналогових» методів. Потрібно використовувати весь потенціал генеративного ШІ для:

- аналізу звітів уряду;
- виявлення суперечностей у законопроектах;
- обробки масових скарг громадян;
- моніторингу діяльності державних органів;
- підготовки рекомендацій для комітетів.

Як показано в монографії [1] вже сьогодні можливо:

[27] Ланде Д., Страшной Л. Формування мереж понять в галузі права за допомогою системи штучного інтелекту. Інформація і право, 2023. N 2(45)/2023. – С. 88-93. DOI: 10.37750/2616-6798.2023.2(45).282326

[28] Lamprecht, A. (2025). Digital Twins of Law: Embracing Complexity. In: Jacob, K., Schindler, D., Strathausen, R., Waltl, B. (eds) Liquid Legal – Sustaining the Rule of Law. Law for Professionals. Springer, Cham. DOI: 10.1007/978-3-031-78596-2_6

[29] Reshef Kera, D., Navon, E., Wellner, G. and Kalvas, F., 2024. Governance in Silico: Experimental Sandbox for Policymaking over AI Agents. DOI: 10.21606/drs.2024.200

– будувати семантичні мережі із текстів законів за допомогою ChatGPT та Gephi;

– виявляти нормативні прогалини та конфлікти між законами;

– формувати причинно-наслідкові моделі для аналізу політичних процесів;

– візуалізувати зв'язки між суб'єктами контролю.

Наприклад, у цій монографії показано, як за допомогою промпту можна виявити 10 основних причин суперечності визначень у законодавчих документах, таких як «Неоднозначність», «Відсутність узгодження», «Зміни законодавства», «Технічні помилки», «Лакуни» тощо. Ці причини були отримані через первинний промпт-запит до систем ГШІ, а потім уточнені на наступних рівнях, що дозволяє будувати мережеву структуру знань.

Це вже не просто промпти – це початок формалізації правових процесів.

Крім того, у роботах з семантичного нетворкінгу (наприклад, у монографії[30]) було показано, як можна витягувати пари сутностей із текстів (наприклад, «Податкові перевірки; Бізнес», «Блокування податкових накладних; Зменшення кількості скарг»), будувати мережі знань, наведені у форматі CSV, після чого завантажувати їх у Gephi для візуалізації та аналізу.

Ці підходи вже використовуються в діяльності Офісу Омбудсмена України. Проєкт EU4DigitalUA, що фінансується ЄС, у тісній співпраці з Офісом Омбудсмана та Міністерством цифрової трансформації України, провів презентацію перших в Україні рекомендацій з питань захисту права людини та права на приватність при розробці та впровадженні технологій штучного інтелекту.

«Розробка рекомендацій – це перший крок на шляху до етичного використання ШІ в Україні. Ця ініціатива підкреслює прагнення не лише використовувати технологічні досягнення, але й робити це з особливою увагою до захисту прав людини, зокрема права на приватність» – наголосили в керівництві проєкту EU4DigitalUA (https://ombudsman.gov.ua/news_details/eu4digitalua-ombudsman-office-and-mintsyfra-presents-artificial-intelligence-guidelines).

Генеративний ШІ стає не просто асистентом, а партнером у прийнятті рішень, аналітиком, інтерпретатором норм, інструментом прозорості. Але разом із потенціалом таких систем виникають і нові виклики.

[30] Dmytro Lande, Leonard Strashnoy. GPT Semantic Networking: A Dream of the Semantic Web - The Time is Now. - Kyiv: Engineering, 2023. - 168 p. ISBN 978-966-2344-94-3, DOI: 10.5281/zenodo.14278893

1.2. Застосування генеративного ШІ на різних етапах парламентського контролю

Розвиток генеративного штучного інтелекту відкрив принципово нові можливості для модернізації інститутів державного управління, серед яких особливе місце посідає парламентський контроль – важливий механізм підзвітності влади, який потребує не лише інформаційної, а й інтелектуальної підтримки. Вже сьогодні ГШІ демонструє свою ефективність не як інструмент, що замінює людину, а як асистент, який посилює аналітичні можливості парламентарів, дозволяючи їм швидше орієнтуватися в складних правових, адміністративних та соціальних процесах.

Фундаментом для сучасного розвитку цього напряму стали роботи [1, 31], у якій було вперше систематично описано методологію застосування великих мовних моделей у практиці парламентської діяльності. У цій роботі було показано, що ГШІ може бути ефективно використаний для автоматизації аналітичних процесів, формування семантичних мереж, виявлення правових суперечностей, аналізу діяльності державних органів та підтримки прийняття рішень. Ця праця стала не лише теоретичним підґрунтям, а й практичним посібником для подальших наукових та прикладних розробок у галузі інтелектуального контролю.

Аналіз правових і адміністративних документів

Однією з найважливіших функцій парламентського контролю є аналіз діяльності виконавчої влади, зокрема – звітів уряду, законопроектів, указів, розпоряджень та інших нормативно-правових актів. Обсяг цих документів часто перевищує можливості ручного аналізу, особливо в умовах швидкого прийняття рішень. Саме тут ГШІ стає незамінним інструментом.

У роботі[1] продемонстровано, як за допомогою LLM можна:

– автоматично аналізувати звіти уряду, виявляючи основні напрямки діяльності, виконання бюджету, реалізацію програм;

– перевіряти законопроекти на наявність суперечностей, технічних помилок, лакун, неоднозначних формулювань;

– обробляти масові скарги громадян, витягуючи з них типові проблеми, визначаючи суб’єктів порушень, формуючи зв’язки між подіями та органами влади.

Наприклад, за допомогою промпту типу:

«Розкладіть поняття «Парламентський контроль» на 10 часткових понять. Кожне повинно містити не більше трьох слів. Представте відповідь у формі: «часткове поняття; Парламентський контроль»

система генерує структурований перелік, такий як:

[31] Інформатика парламентського контролю : посібник. Д.В. Ланде, В.М. Фурашев, С.М. Брайчевський. – Київ 2022. – 256 с. ISBN 978-966-2344-80-6

Нагляд за владою; Парламентський контроль

Аналіз діяльності уряду; Парламентський контроль

Парламентські слухання; Парламентський контроль

Подання запитань; Парламентський контроль

Акти вето; Парламентський контроль

Звітність влади; Парламентський контроль

Обговорення бюджету; Парламентський контроль

Парламентські запити; Парламентський контроль

Стратегічне планування; Парламентський контроль

Правова експертиза; Парламентський контроль

Цей перелік стає основою для подальшого аналізу – до кожного з вибраних понять можна створити і застосувати додаткові промпти, щоб визначити чинники, ризики, історичні тенденції, правові наслідки.

Формування семантичних мереж як основи причинно-наслідкових моделей

Однією з інноваційних методологій, запропонованих у монографії[32], є формування семантичних мереж на основі аналізу текстів за допомогою ГШІ. Цей підхід дозволяє перейти від аналізу окремих документів до побудови інтегрованих моделей знань, де норми, події, суб'єкти та наслідки зв'язані між собою.

Процес включає декілька етапів:

1. Витяг пар сутностей з текстів (наприклад, «податкові перевірки – бізнес», «блокування податкових накладних – зменшення скарг»).
2. Формування списку зв'язків у форматі CSV, придатному для імпорту в інструменти візуалізації.
3. Візуалізація мережі за допомогою програмних засобів, таких як Gephi (https://gephi.org/), GraphViz (https://graphviz.org/) або SocNetV (https://socnetv.org/), що дозволяє побачити структуру взаємодій, виявити центральні вузли, кластери, прогалини.
4. Побудова причинно-наслідкових моделей, де початковий стан (наприклад, «прийняття закону») через ланцюжок подій призводить до кінцевого результату («зміна поведінки громадян»).

---

32 Д. Ланде, Л. Страшной. Семантичний нетворкінг на основі великих мовних моделей : монографія. – Київ: ТОВ "Інжиніринг", 2025. – 274 с. ISBN 978-617-8180-01-0

Такі моделі вже використовувалися для аналізу діяльності Ради бізнес-омбудсмена, де було побудовано мережу зв'язків між «податковими питаннями», «зловживаннями», «правоохоронними органами» та «документацією». Це дозволило виявити системні проблеми, які не були очевидні при традиційному аналізі.

Практичні кейси щодо виявлення суперечностей та аналіз діяльності органів влади

На практиці ГШІ вже використовується для вирішення конкретних завдань парламентського контролю. Серед кейсів – виявлення суперечностей у законодавстві, що є однією з найпоширеніших форм правових помилок. За допомогою промптів можна виявити:

- технічні помилки (неправильні посилання, відсутність визначень);
- нормативні прогалини (відсутність регулювання певних ситуацій);
- дублювання норм (різні закони регулюють одну й ту саму діяльність);
- протиріччя між нормами (одна норма дозволяє, інша забороняє).

Наприклад, як вже було зазначено, як за допомогою запиту до системи ГШІ авторами було отримано 10 основних причин суперечності визначень у законодавчих документах, серед яких – «Неоднозначність», «Відсутність узгодження», «Зміни законодавства», «Технічні помилки», «Лакуни» тощо. Цей список може бути подальшим уточнений наступними запитами: «Перелічіть 10 причин "неоднозначності"...», що дозволяє побудувати мережеву структуру знань шляхом ієрархічного введення промптів [1].

Крім того, ГШІ використовується для аналізу діяльності конкретних державних органів – податкової, митної, правоохоронної сфери. Наприклад, за допомогою аналізу скарг громадян можна виявити системні порушення, частотність певних типів зловживань, географічні центри проблем.

Інтеграція з інструментами візуалізації: Gephi, GraphViz, SocNetV

Для ефективного використання семантичних мереж необхідна їх візуалізація – саме тут інтеграція ГШІ з інструментами мережевого аналізу стає критично важливою. У монографії[33] 2023 року було детально описано, як:

- використовувати ChatGPT для генерації пар сутностей;
- експортувати їх у CSV;
- імпортувати в Gephi для побудови графу;

[33] Dmytro Lande, Leonard Strashnoy. GPT Semantic Networking: A Dream of the Semantic Web – The Time is Now. – Kyiv: Engineering, 2023. – 168 p. ISBN 978-966-2344-94-3, DOI: 10.5281/zenodo.14278893

– застосовувати алгоритми (наприклад, ForceAtlas2[34]) для виявлення кластерів;

– використовувати GraphViz для створення статичних діаграм;

– аналізувати мережу за допомогою SocNetV – інструменту для дослідження соціальних мереж.

Ця інтеграція дозволяє не лише бачити структуру знань, а й аналізувати її: визначати центральність вузлів, щільність зв'язків, інформаційні прогалини, аномалії. Наприклад, у мережі, побудованій на основі аналізу діяльності митниці, можна виявити, що певний регіон стає «чорною дірою» – там майже відсутні зв'язки між суб'єктами, що може свідчити про прихованість процесів.

Від аналізу до підтримки рішень

Таким чином, застосування генеративного ШІ в парламентському контролі вже вийшло за межі простого аналізу текстів. Він стає інструментом підтримки рішень, який автоматизує рутинні завдання, виявляє приховані закономірності, формує структуровані моделі знань та забезпечує наочність через візуалізацію.

Однак, ГШІ не надає остаточну відповідь. Цій підхід має обмеження: галюцинації, відсутність відтворюваності, «чорний ящик». Тому наступний етап – це перехід від аналізу із застосуванням LLM до структурованої системи контролю, де вивід ГШІ не просто аналізується, а формалізується, верифікується, перетворюється на виконувану логічну структуру.

1.3. Перспективи розвитку – створення інтелектуальної системи контролю

На сьогоднішньому етапі застосування генеративного штучного інтелекту в парламентському контролі ми перебуваємо на порозі фундаментального результату – створення інтелектуальної системи контролю, здатної до автономного аналізу, сценарного моделювання, прогнозування наслідків та ініціювання дій. Цей шлях веде від реактивної обробки інформації до проактивної координації політичних процесів, де система не просто допомагає, а участь у формуванні політики.

Від відповідей на запити до автономного аналізу та прогнозування

Сучасні системи, такі як ChatGPT, Llama чи Gemini, вже показали свою ефективність у формуванні відповідей на запити, узагальненні документів, виявленні важливих тем у звітах органів виконавчої влади. Очевидно, майбутнє – не в ручному запитуванні, введенні окремих промптів, а в

[34] Jacomy, M., Venturini, T., Heymann, S. and Bastian, M., 2014. ForceAtlas2, a continuous graph layout algorithm for handy network visualization designed for the Gephi software. PloS one, 9(6), p.e98679. DOI: https://doi.org/10.1371/journal.pone.0098679

автономному інтелектуальному аналізі. Зокрема може бути створена система, яка автономно здійснює моніторинг законопроектів, указів, постанов Кабінету Міністрів тощо, виявляє потенційні суперечності з чинним законодавством, оцінює ризики для бізнесу, громадян, державних фінансів, прогнозує наслідки прийняття тих чи інших рішень на основі історичних даних та моделювання.

Така система вже не просто «відповідає» на запити, як, наприклад, інформаційно-пошукова система, – вона аналізує, попереджає, пропонує. Така система зможе генерувати інтерактивні звіти, які не просто описують стан справ, а пропонувати декілька сценаріїв подальшого розвитку, які відповідають на питання: «що буде, якщо», «які наслідки», «які альтернативи».

Створення цифрових двійників законодавства та імітаційних моделей

Однією з найперспективніших ідей є створення цифрових двійників законодавства[35, 36] – повноцінних імітаційних моделей, які відтворюють поведінку правової системи в умовах змін. Такі моделі можуть:

– симулювати вплив нового закону на існуючу нормативну базу;

– виявляти «сліпі зони» – ситуації, які залишаються нерегульованими;

– моделювати реакцію різних суб'єктів (громадян, бізнесу, державних органів) на нові норми.

Наприклад, при розгляді законопроекту про електронні договори система може моделювати, як це вплине на податкові надходження, кількість судових спорів, рівень цифрової включеності, що дозволить перейти від інтуїтивних рішень до обґрунтованого прогнозування.

Подібні підходи вже використовуються в інших країнах[37]. Наприклад, у Сінгапурі розробляються цифрові двійники міст[38], а в Естонії – моделі цифрової держави[39]. Україна має можливість стати лідером у створенні цифрового двійника парламентського контролю – системи, яка буде імітувати весь ланцюжок від законопроекту до його виконання.

[35] Teller, M., 2021. Legal aspects related to digital twin. *Philosophical Transactions of the Royal Society A*, *379*(2207), p.20210023.

[36] Buonocore, L., Yates, J. and Valentini, R., 2022. A proposal for a forest digital twin framework and its perspectives. *Forests*, *13*(4), p.498.

[37] von Lucke J, Fitsilis F, Etscheid J. Using Artificial Intelligence for Legislation-Thinking About and Selecting Realistic Topics. InEGOV-CeDEM-ePart-* 2022.

[38] Menkhoff, T., Wong, C. and Ritter, W., 2024. Singapore's approach towards developing vibrant urban innovation spaces. In *Visions for the future: Towards more vibrant, sustainable and smart cities* (pp. 1-33).

[39] Salumaa-Lepik, K. and Nisu, N., 2024. European values, artificial intellligence and e-governance in Estonia and EU. *Internet of Things*, *28*, p.101278.

Рій віртуальних експертів: колективний аналіз та розподіл ролей

Ще одним кроком уперед є концепція рою віртуальних експертів – команди спеціалізованих агентів, кожен з яких моделює поведінку фахівця у певній галузі: юрист, економіст, фінансист, фахівець з кібербезпеки, соціолог.

Ці агенти можуть аналізувати законопроект з різних поглядів одночасно, обмінюватися даними, знаходити конфлікти між нормами та економічними наслідками, досягати консенсусу або виявляти розбіжності, формувати колективний висновок для парламентарів.

Такий підхід, описаний у дослідженнях[40, 41], дозволяє створити інтелектуальний колектив, який працює швидше, об'єктивніше та повніше, ніж окремий експерт.

Хоча повномасштабна оркестрація рою ще перебуває на стадії розробки, вже сьогодні можливо моделювати розподіл ролей, застосовувати статистичний аналіз, визначати логіку взаємодії та тестувати сценарії колективного прийняття рішень.

Автономний моніторинг зворотного зв'язку від громадян

Однією з найважливіших функцій парламентського контролю є зв'язок із громадою. У майбутньому система може стати автономним монітором зворотного зв'язку, який аналізує масові скарги, звернення, коментарі у соціальних мережах, виявляє нові проблеми ще до їх масового розголосу, формує рейтинги проблем за регіонами, сферами, групами населення, пропонує пріоритетні теми для контролю.

Наприклад, система може виявити, що в певному регіоні різко зросла кількість скарг на блокування податкових накладних, і автоматично сформувати аналітичний матеріал для комітету з питань податкової та митної політики.

Інтеграція з цифровими платформами

Для реалізації всіх цих перспектив необхідна глибока інтеграція з державними цифровими платформами, зокрема такими системами:

СИСТЕМА / ПЛАТФОРМА	ФУНКЦІЯ ІНТЕГРАЦІЇ
Електронна система Верховної Ради України (е-	Для доступу до актуальних законопроектів, протоколів засідань, голосувань, версій документів та

[40] D. Lande, I. Svoboda, L. Alekseichuk, L. Strashnoy. Methodology of a Swarm of Virtual Experts for Evaluating the Weight of Connections in Networks. *Theoretical and Applied Cyber Security*. Vol. 7 No. 2 (2024). DOI: 10.20535/tacs.2664-29132024.2.319946

[41] Lande, Dmitry; Strashnoy, Leonard. Implementation Of The Concept Of A "Swarm Of Virtual Experts" In The Formation Of Semantic Networks In The Field Of Cybersecurity Based On Large Language Models. *SSRN Preprint*: 4978924, DOI: 10.2139/ssrn.4978924 (Oct 17, 2024). – 15 p.

СИСТЕМА / ПЛАТФОРМА	ФУНКЦІЯ ІНТЕГРАЦІЇ
Parliament)	історії їх змін. В основу системи покладені діючі автоматизовані системи Верховної Ради. України, перелік яких визначено Розпорядженням Голови Верховної Ради України.
Єдиний державний реєстр нормативно-правових актів (ЄДРНПА)	Для централізованого доступу до всіх діючих актів, їх версій, статусів (діючий/скасований/на розгляді) та посилань на підзаконні акти. Ведення реєстру покладено на Міністерство юстиції України.
ProZorro – електронна система публічних закупівель в Україні	Найбільший майданчик державних і комерційних торгів. Офіційний учасник системи електронних державних закупівель Prozorro та системи онлайн аукціонів з продажу та здачі в оренду.
Цифрова платформа «Дія»	Портал та застосунок для отримання державних послуг онлайн та зберігання електронних документів, таких як паспорт, водійське посвідчення та свідоцтва про народження.
Вебпортал Кабінету Міністрів України (kmu.gov.ua)	Для отримання проектів підзаконних актів, аналізу їх відповідності законам, виявлення конфліктів з компетенціями міністерств.

Майбутній інструментарій може включати:

– інтерактивні звіти, де кожна норма має посилання на семантичну мережу;

– алерти про суперечності, які автоматично надходять депутатам під час розгляду законопроектів;

– рекомендації щодо змін у тексті законів, законопроектів, нормативних актів тощо;

– панель управління контролем, де можна відстежувати стан розслідувань, динаміку порушень, ефективність виконання резолюцій.

Така інтеграція перетворить парламент на цифрову, інтелектуально підсилену інституцію, здатну швидко реагувати на виклики, забезпечувати прозорість та підзвітність.

Таким чином, перспективи розвитку вказують на те, що ми стоїмо на порозі нової ери парламентського контролю – не просто автоматизованої, а інтелектуальної, заснованої на семантичному нетворкінгу, безкодовому програмуванні та колективному аналізі. Ця система вже не буде залежати виключно від людського часу чи ресурсів – вона стане постійним, автономним, інтелектуальним партнером у забезпеченні підзвітності влади.

1.4. Обмеження генеративних моделей

Незважаючи на значний потенціал генеративних мовних моделей у сфері парламентського контролю, їх застосування супроводжується рядом істотних

обмежень, які не дозволяють вважати ці системи остаточним рішенням. Ці обмеження не є просто технічними недоліками – вони стосуються фундаментальних принципів роботи ГШІ і вимагають розробки нових архітектур, методологій та інструментів для забезпечення надійності, прозорості та підзвітності в правових системах. Нижче розглядається п'ять основних викликів, які визначають межі сучасного його застосування в парламентському контролі.

Генерація неіснуючих норм, посилань, фактів

Однією з найбільш критичних проблем застосування ГШІ є галюцинації [42, 43] – ситуація, коли модель генерує інформацію, яка здається правдоподібною, але не відповідає дійсності. У контексті парламентського контролю це може проявлятися у вигляді:

– вигаданих статей законів, постанов Кабінету Міністрів, інших нормативних актив;
– неіснуючих посилань на нормативні акти;
– помилкових даних про зміст документів, наприклад, звітів уряду;
– вигаданих прецедентів судової практики.

Галюціонації ГШІ можуть призвести до помилкових висновків, що робить їх використання без додаткових механізмів верифікації у критичних завданнях ризикованим.

Відсутність логічної відтворюваності

Генеративні моделі ШІ не гарантують відтворюваності результатів[44, 45]. Один і той самий запит, заданий у різний час або навіть під час одного сеансу, може отримати різні відповіді. Це не є технічною помилкою, а фундаментальною особливістю їхньої архітектури: моделі не виконують детермінованих логічних висновків, а симулюють мову на основі стохастичних розподілів ймовірностей, вибираючи наступні токени за принципом «найбільш правдоподібного», а не «найлогічнішого».

У контексті парламентського контролю це має серйозні наслідки, зокрема:

[42] Dahl, M., Magesh, V., Suzgun, M. and Ho, D.E., 2024. Large legal fictions: Profiling legal hallucinations in large language models. *Journal of Legal Analysis*, *16*(1), pp.64-93.
[43] Chang, Y., Wang, X., Wang, J., Wu, Y., Yang, L., Zhu, K., Chen, H., Yi, X., Wang, C., Wang, Y. and Ye, W., 2024. A survey on evaluation of large language models. *ACM transactions on intelligent systems and technology*, *15*(3), pp.1-45.
[44] Kim, S.S., Vaughan, J.W., Liao, Q.V., Lombrozo, T. and Russakovsky, O., 2025, April. Fostering appropriate reliance on large language models: The role of explanations, sources, and inconsistencies. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems* (pp. 1-19).
[45] Elazar, Y., Kassner, N., Ravfogel, S., Ravichander, A., Hovy, E., Schütze, H. and Goldberg, Y., 2021. Measuring and improving consistency in pretrained language models. *Transactions of the Association for Computational Linguistics*, *9*, pp.1012-1031.

– аналіз одного й того самого законопроекту може надавати різні списки суперечностей з Конституцією чи міжнародними договорами;

– виявлення правових лакун може варіюватися – одна відповідь вказує на відсутність регулювання в сфері цифрової безпеки, інша – на надлишковість норм у сфері освіти;

– прогнози наслідків реформи можуть бути протилежними: одна генерація передбачає зростання адміністративних витрат, інша – їх скорочення.

Така системна нестабільність висновків робить неможливим використання генеративних моделей як офіційного інструменту підтримки законотворчих рішень, оскільки вона прямо суперечить фундаментальному принципу правової певності – одному з крайових каменів демократичного правового держави, закріпленому у Статті 8 Конституції України щодо визнання і дії принципу верховенства права.

Правова певність передбачає, що однакові факти повинні призводити до однакових правових наслідків, незалежно від того, хто їх аналізує – суддя, депутат чи алгоритм. Якщо система дає різні відповіді на однаковий запит – вона не є інструментом права, а лише інструментом імітації.

Це не просто технологічний виклик – це правовий ризик. Використання таких систем без механізмів верифікації та детермінованого контролю може призвести до непередбачуваних, неоднорідних та незаконних інтерпретацій закону, що порушує принципи рівності перед законом та відповідальності держави за свої рішення.

Для будь-якої системи, що претендує на роль цифрового двійника законодавства, відтворюваність результатів – це не опціональна вимога, а необхідна умова легітимності.

Таким чином, логічна відтворюваність – це не технічна деталь, а правова необхідність.

Будь-яка система, що підтримує законотворчі рішення, повинна гарантувати однакові висновки при однакових вхідних даних, використовувати детерміновані логічні механізми для формування висновків, обмежувати застосування ГШІ на рівні лише формулювання пояснень – а не генерації власних незаперечних правових інтерпретацій.

Відсутність пояснення («Чорний ящик»)

Більшість LLM функціонують як «чорний ящик» – користувач бачить лише вихідний текст, але не бачить, як саме модель дійшла до висновку[46], [47].

[46] Patidar, N., Mishra, S., Jain, R., Prajapati, D., Solanki, A., Suthar, R., Patel, K. and Patel, H., 2024. Transparency in AI decision making: A survey of explainable AI methods and applications. *Advances of Robotic Technology*, *2*(1).

[47] Dancy, T. and Zalnieriute, M., 2025. AI and Transparency in Judicial Decision Making. *Oxford Journal of Legal Studies*, p.gqaf030.

Відсутність логічного сліду, тобто послідовності міркувань, джерел даних, вагових коефіцієнтів, робить аналіз непрозорим.

У правовій сфері, де кожне твердження має бути обґрунтоване, це є серйозним обмеженням. Наприклад, якщо система виявляє «конфлікт між двома нормами», але не може показати, на яких саме положеннях вона грунтується, цей висновок не може бути прийнятий. Для парламентського контролю необхідна пояснена, повернена інформація, яка дозволяє відтворити логічний ланцюжок і перевірити його на коректність.

Відсутність виконуваної логіки

Навіть найбільш структурований вивід LLM залишається текстом, що не містить логічних примітивів, таких як «Умова», «Цикл», «Функція», «Перехід», тощо, які дозволяють запустити аналіз, перевірити його на консистентність, інтегрувати в інші системи.

Наприклад, якщо система генерує список причин суперечностей у законодавстві (наприклад, «технічні помилки», «відсутність узгодження», «лакуни»), цей список не можна використати як модуль у більш складній системі контролю.

Текст не є програмою, тому він сам по собі не може бути запущений чи перевірений, автоматично перевірений, поширений на інші документи чи вбудований у причинно-наслідкову модель без додаткової ручної обробки. Це обмежує масштабованість і ефективність застосування ГШІ.

Проблема часової актуалізації

Більшість LLM мають фіксований період навчання, після якого їх знання не оновлюються автоматично. Наприклад, модель може бути навчена на даних попереднього року, і тому не знає про нові закони, постанови, зміни в конституційному судочинстві чи міжнародних угод.

Це створює серйозну проблему для парламентського контролю, де актуальність інформації є найважливішим фактором. Система може пропустити важливі зміни у законодавстві або дати застарілі рекомендації.

Хоча методи Retrieval Augmented Generation (RAG)[48] дозволяють «підключати» актуальні джерела на момент запиту, вони не гарантують повноти, точності та цілісності[49]:

– RAG залежить від якості індексованої бази даних – якщо база даних не містить свіжого документа, RAG його не побачить;

[48] Yu, H., Gan, A., Zhang, K., Tong, S., Liu, Q. and Liu, Z., 2024, August. Evaluation of retrieval-augmented generation: A survey. In *CCF Conference on Big Data* (pp. 102-120). Singapore: Springer Nature Singapore.

[49] Zafar, A., 2024. Balancing the scale: navigating ethical and practical challenges of artificial intelligence (AI) integration in legal practices. *Discover Artificial Intelligence*, *4*(1), p.27.

– Модель може неправильно інтерпретувати отриманий текст, навіть якщо він актуальний;

– Відсутність синхронізації з офіційними джерелами в режимі реального часу робить систему вразливою до «часових лакун» – періодів між публікацією закону та його індексацією.

Всі наведені вище обмеження не означають, що генеративний ШІ не має майбутнього в парламентському контролі. Навпаки, вони вказують на напрямки для наукового та технологічного розвитку. Можна використовувати LLM як джерело первинного аналізу, але потім фіксувати логічні примітиви, формалізувати виводи, будувати виконувані модулі.

Саме цей шлях – від тексту до структурованої системи – стане темою наступних розділів цієї монографії. Буде показано, як можна на основі генеративного аналізу перейти до безкодового програмування, де кожна умова, кожна функція, кожен зв'язок стає частиною логічно цілісної, верифікованої, відтворюваної системи контролю.

1.5. Потреба в логічній формалізації документів

Сучасні обмеження генеративних моделей ШІ не можна розглядати як підставу для повної відмови від їх використання в парламентському контролі. Навпаки, ці обмеження вказують на необхідність подальшого розвитку, переходу від реактивного аналізу до активної інженерії правових процесів. Інтелектуальна система контролю має бути не просто інструментом для генерації текстів, а структурованою, логічно цілісною, відтворюваною системою, здатною до автономного функціонування, верифікації та підзвітності. Ключем до цього переходу є логічна формалізація запитів і документів.

Від генеративного аналізу до інженерії правових процесів

До сьогоднішнього дня застосування ГШІ в парламентському контролі було зосереджене на аналізі текстів: узагальнення звітів, виявлення основних тем, побудова семантичних мереж. Це важливий крок, але він залишається на рівні інтерпретації. Наступний етап – це інженерія: створення системи, яка не просто "розуміє" текст, а перетворює його на виконувану логічну структуру, здатну до моделювання, перевірки та інтеграції в робочі процеси.

Це означає, що замість того, щоб отримувати від LLM висновок у вигляді абзацу, ми повинні фіксувати систему логічних примітивів – умов, циклів, функцій, переходів, міток – і будувати з них виконувані модулі, які можуть бути запущені, перевірені на консистентність і використані в інших системах. Це вже не простий формальний аналіз – це програмування без коду, засноване на природній мові.

Формалізація як ключ до прозорості, відтворюваності, підзвітності

Однією з найважливіших вимог до будь-якої системи, що використовується в державному управлінні, є підзвітність. У правовій сфері кожне рішення,

кожен аналітичний висновок повинен мати обґрунтування, яке можна перевірити, відтворити та опонувати.

Саме цього не вистачає в сучасних LLM: вони працюють як «чорний ящик», і їх висновки не мають відтворюваності – один і той самий запит може дати різні результати. Формалізація вирішує цю проблему. Коли логічна структура виводу фіксується – наприклад, у вигляді умови «*IF (законопроект містить норму, що суперечить Конституції) THEN (позначити як конфлікт)*» – вона стає прозорою, верифікованою і підзвітною.

Така система дозволяє не просто приймати рішення, а документувати логічний шлях до них, що є критично важливим для демократичного контролю.

Безкодове програмування як міст між природною мовою та виконуваною логікою

Центральним інструментом логічної формалізації є безкодове програмування – технологія, яка дозволяє перетворювати природну мову на виконувану логічну структуру без необхідності писати код. Це не просто автоматизація, а нова парадигма взаємодії людини та машини.

У статті щодо застосування фреймворку безкодового програмування при вирішенні завдань парламентського контролю[50] було показано, як за допомогою промптів можна:

- виявляти логічні примітиви в текстах законопроектів;
- формувати з них виконувані модулі;
- інтегрувати їх у системи контролю.

Наприклад, замість того, щоб просто прочитати, що "законопроект передбачає штраф за порушення", система може побудувати логічну функцію:

```
FUNCTION ApplyPenalty(entity, violation_type):
  IF violation_type == "tax_evasion" AND entity == "business":
    RETURN fine_amount * 1.5
  ELSE IF violation_type == "document_misfiling":
    RETURN fixed_fee
  END IF
END FUNCTION
```

[50] Ланде Д.В., Фурашев В.М. Застосування фреймворку безкодового програмування при вирішенні завдань парламентського контролю. Інформація і право, 2025. – № 2(53). – С. 88-103. DOI: 10.37750/2616-6798.2025.2(53).334055

Ця функція може бути використана для моделювання наслідків, аналізу бюджетних витрат або інтеграції в інші системи.

Семантичний нетворкінг як основа для динамічної, адаптивної, інтерактивної системи контролю

Логічна формалізація не ізольована – вона інтегрується в більш широку архітектуру, що визначається семантичним нетворкінгом – процесом побудови, розвитку та аналізу динамічних мереж знань, які відображають взаємозв'язки між поняттями, нормами, подіями, суб'єктами та наслідками.

Як зазначено в монографії[51], такі мережі дозволяють:

- виявляти приховані зв'язки між законами;
- моделювати причинно-наслідкові ланцюжки;
- проводити сценарний аналіз;
- візуалізувати структуру контролю.

Коли інформаційна мережа доповнюється виконуваною логікою, вона перетворюється на інтерактивну систему, здатну не лише аналізувати, а й прогнозувати, моделювати та ініціювати.

Отже, обмеження генеративних моделей – це не фатальний недолік, а виклик, який вимагає наукового розв'язання. Отже обмеження – це не зупинка дослідження, а початок нового етап, переходу від традиційного аналізу тексту до структурованої системи контролю через логічну формалізацію, безкодове програмування та семантичний нетворкінг.

Цей шлях веде не до заміни людини машиною, а до інтелектуального підсилення – створення системи, яка допомагає, зокрема, парламентарям приймати більш обґрунтовані, прозорі та ефективні рішення. Саме цей шлях стане темою наступних розділів цієї монографії.

[51] Ланде Д., Страшной Л. Семантичний нетворкінг на основі великих мовних моделей : монографія. Д.В. Ланде, Л.Л. Страшной. – Київ: ТОВ "Інжиніринг", 2025. – 274 с. ISBN 978-617-8180-01-0

Розділ 2. Логічні примітиви безкодового програмування у правових системах

Генеративні мовні моделі демонструють вражаючу здатність до аналізу, узагальнення та інтерпретації правових текстів. Однак їх вивід залишається у формі природної мови – тексту, який неможливо запустити, перевірити на логічну цілісність чи інтегрувати в інші системи як виконувану логіку. Це створює фундаментальну проблему: як перетворити описовий аналіз на інженерію правових процесів?

Відповідь полягає в логічній формалізації – процесі, який дозволяє виявляти, фіксувати та використовувати логічні примітиви, приховані в текстах, що генеруються LLM. Такі примітиви, як «*умова*», «*цикл*», «*функція*», «*перехід*», «*мітка*» – можна розглядати як будівельні блоки, з яких складаються всі виконувані системи. Їх фіксація дозволяє перейти від «розуміння» до «виконання», від «аналізу» до «моделювання», від генерації тексту до безкодового програмування.

У цьому розділі ми розглянемо, як можна систематично виявляти ці примітиви в діалозі з LLM, формалізувати їх, будувати з них виконувані модулі та інтегрувати в систему парламентського контролю. Особливу увагу при цьому приділено фреймворку безкодового програмування[52], розробленому авторами, який вже успішно застосовується для аналізу законопроектів, виявлення суперечностей та побудови причинно-наслідкових моделей. Цей підхід не вимагає від користувача знання мов програмування – вся логіка формується через природну мову, а потім автоматично фіксується в структуровану форму.

2.1. Автоматична екстракція логічних структур із виводів LLM

Перехід від генеративного аналізу до структурованої системи парламентського контролю неможливий без автоматизованого витягу логічних структур із текстів, які генерують великі мовні моделі. Хоча LLM не є програмним кодом, вони часто природно використовують логічні конструкції – умови, функції, цикли, переходи – особливо при аналізі правових документів. Завдання полягає в тому, щоб автоматично ідентифікувати, відокремлювати та фіксувати ці структури, перетворюючи природну мову на формалізовані логічні примітиви, придатні для подальшої обробки, верифікації та виконання.

Автоматична екстракція логічних структур ґрунтується на тому, що всі запити і тексти, що досліджуються, мають характерні мовні маркери, які можна виявити за допомогою семантико-синтаксичного аналізу. Наприклад:

[52] Ланде Д.В., Фурашев В.М. Застосування фреймворку безкодового програмування при вирішенні завдань парламентського контролю. Інформація і право, 2025. № 2(53). - С. 88-103. DOI: /10.37750/2616-6798.2025.2(53).334055

– Умови часто починаються зі слів *«якщо»*, *«у разі»*, *«коли»*, *«якщо не»*.

– Функції описуються як терміни *«процедура»*, *«механізм»*, *«алгоритм»*, *«порядок»*.

– Цикли асоціюються зі словами *«щороку»*, *«щокварталу»*, *«регулярно»*, *«до виконання»*.

– Переходи – зі словами *«передається»*, *«спрямовується»*, *«переходить до»*.

Ці маркери дозволяють побудувати словникові фільтри та синтаксичні шаблони, які автоматично виявляють потенційні логічні блоки в тексті. Наприклад, речення *«Якщо платник не подав звіт протягом 10 днів, до нього застосовуються санкції»* може бути розпізнане як умова і витягнуте для подальшої формалізації.

Однак самостійне розпізнавання на рівні слів недостатнє. Для підвищення точності використовується промпт-кероване зондування – стратегія, при якій сама LLM виступає як інструмент аналізу.

Використання промпт-керованого зондування для витягу структур

Промпт-кероване зондування – це метод, який дозволяє керувати процесом екстракції через чіткі запити до LLM. Наприклад, замість того, щоб аналізувати вивід після генерації, ми можемо спрямовано попросити модель:

«Проаналізуйте наведений текст законопроекту та виділіть усі логічні структури: умови (IF-THEN), функції (процедури), цикли, переходи. Представте результат у структурованому форматі: "Тип; Умова/Функція; Джерело (цитата)"».

Такий промпт перетворює LLM з генератора тексту на інструмент семантичного аналізу, здатний витягувати структуровані дані. Як показано в попередній монографії, цей підхід вже успішно застосовується для виявлення причин суперечностей у законодавстві, формування ієрархій понять та побудови мереж зв’язків.

Наприклад, при запиті:

«Перелічіть 10 причин суперечності визначень в окремих законознавчих документах. Представте у форматі "Причина; Суперечність документів"»,

система генерує структурований перелік, який може бути використаний як вхід для наступних рівнів аналізу.

Алгоритми обробки відповідей LLM: парсинг, нормалізація, перевірка на повноту

Після отримання відповіді від LLM необхідна її **подальша обробка** для перетворення на машинно-читабельну структуру. Цей процес включає кілька етапів:

1. Парсинг: автоматичне розпізнавання формату відповіді (наприклад, роздільник «;», перенос рядка) та витяг окремих записів.
2. Нормалізація: приведення тексту до єдиного вигляду – видалення зайвих символів, приведення до нижнього регістру, уніфікація термінології (наприклад, «штраф» і «санкція» → «штраф»).
3. Перевірка на повноту: порівняння кількості виявлених примітивів із очікуваною (наприклад, 10 причин), виявлення пропущених елементів.

Ці алгоритми можуть бути реалізовані на мові Python або інтегровані в спеціалізовані інструменти, такі як SemanticCore[53], систему аналізу та управління семантичними структурами.

Вбудовування механізмів самоперевірки (self-consistency prompts)

Щоб підвищити надійність екстракції, використовуються промпти самоперевірки, які змушують модель аналізувати власний вивід. Наприклад:

«Вище ви виявили 7 умов у тексті законопроекту. Перевірте, чи всі умовні конструкції були враховані. Якщо є пропущені – додайте їх. Якщо всі враховані – підтвердьте.»

Такий підхід, відомий як self-consistency prompting, дозволяє зменшити кількість пропущених структур і підвищити повноту аналізу. Він особливо ефективний при роботі з довгими текстами, де модель може «проґавити» окремі фрагменти під час першого проходу.

Крім того, можуть використовуватися ітеративні промпти, які «розгортають» кожен виявлений примітив на наступному рівні:

«Для кожної з виявлених умов визначте можливі наслідки та винятки.»

Це дозволяє глибше формалізувати логіку документа.

Інтеграція з семантичними системами для зберігання та керування структурами

Витягнуті та нормалізовані логічні структури не повинні залишатися ізольованими. Вони інтегруються в централізовану систему управління знаннями, таку як, наприклад, SemanticCore, яка дозволяє:

- зберігати витягнуті примітиви у структурованому вигляді (наприклад, у форматі JSON або CSV);
- будувати з них мережі знань;
- здійснювати пошук, фільтрацію, порівняння;
- інтегрувати з інструментами візуалізації (Gephi, GraphViz).

[53] Комп'ютерна програма аналізу семантичних зв'язків "SemanticCore". Ланде Д.В., Фегер А.П. Україна. Свідоцтво про реєстрацію авторського права на твір № 125039 від 24.04.2024

Наприклад, виявлена умова *«якщо не подано звіт – штраф»* може бути збережена як вузол у мережі, пов'язаний із сутностями *«податкова звітність»*, *«санкції»*, *«контроль за дотриманням зобов'язань»*.

Таким чином, автоматична екстракція логічних структур із виводів LLM – це не просто парсинг тексту, а інженерний процес, який поєднує промпт-кероване зондування, алгоритмічну обробку, самоперевірку та інтеграцію в системи управління знаннями. Цей процес є фундаментальним кроком у створенні виконуваних, відтворюваних, верифікованих систем парламентського контролю, які виходять за межі генеративного аналізу і стають інструментами інтелектуальної координації.

2.2. Первинний фреймворк безкодового програмування

Одним із найважливіших наукових проривів останніх років стало усвідомлення того, що генеративні мовні моделі не просто «говорять», а часто видають «логічно осмислені» результати. Навіть не застосовуючи явної архітектури програмування, вони природно використовують логічні конструкції: умови, цикли, функції, переходи. Тому для формалізації запитів користувачів постає задача – не створювати ці структури з нуля, а виявити і зафіксувати їх – перетворити приховану логіку на явну, відтворювану, виконувану.

Майбутнє програмування нагадує еволюційний процес. Як колись клітинні автомати продемонстрували, що прості правила здатні породжувати складні структури, так і сьогоднішні LLM відкривають нові горизонти в управлінні інформацією. Проте, як і в теорії клітинних автоматів, цей підхід має свої межі. Він не усуне всі проблеми, але суттєво змінить роль людини у створенні інформаційних систем. Спеціалістів стане менше, але вони зосередяться на задачах вищого рівня, залишаючи рутинні завдання штучному інтелекту.

У цій новій реальності ключовими навичками стають не знання синтаксису програмних мов, а вміння формулювати завдання таким чином, щоб мовні моделі могли розуміти їх і виконувати. Професія промпт-інженера стає аналогом архітектора, який проєктує систему не з коду, а з набору правил і запитів, суті та лінгвістики. Хорошими промпт-інженерами можуть стати люди, які вміють аналітично мислити, формулювати чіткі й зрозумілі запити та враховувати нюанси людської мови. Це можуть бути фахівці з лінгвістики, методологи, аналітики даних, які звикли до роботи з абстракціями. Навички міждисциплінарного мислення, які дозволяють враховувати різні контексти, стануть надзвичайно цінними.

Однак виникає питання: чи зможуть самі моделі створювати промпти, усуваючи потребу в людях? Хоча моделі вже демонструють здатність генерувати ефективні запити, повністю виключити людину зі цього процесу навряд чи вдасться. Проблема полягає в тому, що навіть найбільш досконалі моделі не можуть самостійно визначити кінцеву мету або оцінити наслідки

своїх дій у ширшому контексті. Люди залишаються необхідними для постановки високорівневих цілей, оцінки результатів і внесення етичних та стратегічних коректив. Взаємодія людини і моделі є синергетичною – моделі розширюють можливості людини, але остаточне слово залишається за нею.

Перехід до безкодових систем і зміщення ролі фахівців до роботи з LLM означає нову еру технологічного розвитку, де взаємодія з моделями стає головним інструментом створення інновацій. Промпт-інженерія вже формує нову професію, яка поєднує навички аналітики, творчості й розуміння технологій. Однак людська участь не зникає – вона трансформується. Людина залишається джерелом цілей, ідей і етичних принципів, які визначають напрямок розвитку моделей. Майбутнє – це партнерство між людиною і штучним інтелектом, у якому людина задає мету, а системи забезпечують засоби її досягнення, відкриваючи нові горизонти уяви та можливостей.

Таким чином, реальним стає формалізація безкодового створення систем через промпт-інжиніринг, аналогічно базовим конструкціям мов програмування. Це такі «цеглинки» (примітиви) – умовні оператори, цикли, функції, але описані природною мовою та формалізовані математично. Крім того, вивчається композиція цих примітивів для побудови складних систем, наприклад, семантичних мереж, і запропоновано концепцію «мови промптів» як домен-специфічної мови (DSL) для безкодової інженерії ШІ.

Для досягнення цієї мети:

– Формалізуються базові примітиви: Уточнюються математичні моделі для умовних операторів, циклів та абстракцій, що дозволяють використовувати природну мову для управління логікою систем.

– Розроблено концепцію «мови промптів»: Визначаються принципи структурування і компонування промптів, що дають можливість керувати системами ШІ через текстові інструкції.

– Впроваджуються адаптовані методи оптимізації: Застосовується градієнтний спуск і ітеративне налагодження для покращення ефективності промптів.

– Представлено модель «рою віртуальних експертів»: Розглядається спільна роботу декількох LLM для подолання обмежень окремих моделей за допомогою агрегації результатів.

Хоча тема промпт-інжинірингу є відносно новою, вона має корені в багатьох сучасних напрямках наукових досліджень, серед яких:

– ***Концепція клітинних автоматів.*** В книзі "Новий вид науки"[54] Стівен Вольфрам запропонував ідею, що прості правила можуть породжувати складну поведінку. Ця концепція виявилася аналогічною до

[54] Wolfram, Stephen. "New kind of science." (2002). Wolfram Media. 1197 p. ISBN: 9781579550080, 1579550088

нашого підходу до промпт-інжинірингу, де базові примітиви (умови, цикли) формують складні системи.

– ***Програмування через приклади (Programming by Example).*** Дослідження у галузі PBE[55] показали, що люди можуть взаємодіяти з системами шляхом надання прикладів, а не детального коду. В роботі описується методологія синтезу програм на основі вхід-вихідних прикладів, що дозволяє непрограмістам автоматизувати повторювані задачі без написання традиційного коду.

– ***Домен-специфічні мови (DSL).*** Існуючі роботи над домен-специфічними мовами дали нам теоретичну основу для розробки "мови промптів", яка є адаптована для безкодового підходу. Зокрема, в роботі[56] на основі визначень даних контрактів між клієнтами та серверами генерується код WCF/C# для службового програмного забезпечення.

– ***Градієнтний спуск для промпт-оптимізації.*** Недавні дослідження[57] продемонстрували можливість застосування градієнтного спуску для покращення промптів, що стало важливою частиною нашого підходу до ітеративного налагодження.

– ***Агрегація прогнозів від різних моделей***. Концепція ансамблів та голосувань лягла в основу авторської моделі «рою віртуальних експертів»[58], де результати декількох LLM агрегуються для отримання більш точного висновку.

Формальні визначення базових примітивів

Кожен примітив у фреймворку безкодового промпт-інжинірингу описується як математичний об’єкт із чіткими правилами трансформації вхідних даних у вихідні. Ці примітиви є основними "цеглинками" для створення складних логічних конструкцій, аналогічних до операторів мов програмування, але заснованих на природній мові.

Примітив "Умова" (If-Else)

Введемо позначення:

– *Input* – вхідні дані (наприклад, текст, числовий параметр тощо).

[55] Gulwani, S. (2016). Programming by examples-and its applications in data wrangling. In Dependable Software Systems Engineering (pp. 137-158). IOS Press.

[56] Hermans, F., Pinzger, M., & Van Deursen, A. (2009). Domain-specific languages in practice: A user study on the success factors. In Model Driven Engineering Languages and Systems: 12th International Conference, MODELS 2009, Denver, CO, USA, October 4-9, 2009. Proceedings 12 (pp. 423-437). Springer Berlin Heidelberg.

[57] Reed S, Zolna K, Parisotto E, Colmenarejo SG, Novikov A, Barth-Maron G, Gimenez M, Sulsky Y, Kay J, Springenberg JT, Eccles T. A generalist agent. arXiv preprint arXiv:2205.06175. 2022 May 12.

[58] Lande, Dmitry; Strashnoy, Leonard. Implementation Of The Concept Of A "Swarm Of Virtual Experts" In The Formation Of Semantic Networks In The Field Of Cybersecurity Based On Large Language Models. SSRN Preprint: 4978924, DOI: 10.2139/ssrn.4978924 (Oct 17, 2024). - 15 p.

– C – умова (предикат), що повертає значення *True* або False.
– A_1, A_2 – дві можливі дії, які виконуються у випадках, коли умова дорівнює True або False відповідно.

Промпт-функція P визначається як:

$$P(Input) = \begin{cases} A_1(Input), & якщо \quad C(Input) = True \\ A_2(Input), & якщо \quad C(Input) = False \end{cases}$$

Це формальне визначення аналогічне класичному оператору *If-Else* у мовах програмування, але застосовується через текстові інструкції.

Механізм роботи примітива «Умова» полягає в тому, щоб передати LLM задачу з двома можливими сценаріями, один з яких виконується в залежності від результату перевірки умови. Умова може бути будь-якою логічною конструкцією, наприклад:

– перевірка наявності певного слова у тексті;
– порівняння чисел;
– аналіз контексту або категорії даних.

Розглянемо такий промпт як приклад:

«Якщо текст містить термін 'кібербезпека', поверни його визначення; інакше – список споріднених термінів.»

У цьому випадку:

– C – наявність слова «кібербезпека» у тексті;
– A_1 – генерація визначення терміну «кібербезпека»;
– A_2 – пошук асоціацій до терміну «кібербезпека».

Цей примітив дозволяє моделі приймати рішення на основі заданих критеріїв, що ефективно моделює логічну умову.

Примітив «Цикл» (For-Loop)

Введемо позначення:

– $S = \{s_1, s_2, ..., s_n\}$ – множина елементів, які потрібно обробити;
– F – операція, яка застосовується до кожного елемента множини.

Промпт-функція $P(S)$ визначається як об'єднання результатів застосування операції F до кожного елемента множини:

$$P(S) = \bigcup_{i=1}^{n} F(s_i).$$

Це формальне визначення аналогічне класичному циклу for у мовах програмування, де одна і та ж операція повторюється для всіх елементів множини.

Механізм роботи примітива «Цикл» полягає в тому, щоб передати LLM список елементів разом із операцією, яку потрібно застосувати до кожного елемента. Результатом є об'єднання всіх отриманих відповідей. Такий підхід корисний для масового аналізу даних, виділення патернів або генерації відповідей для декількох входів одночасно.
Розглянемо такий промпт як приклад:

«Для кожного терміну зі списку ['фішинг', 'брандмауер'] знайди 3 приклади його використання.»

У цьому випадку:

- S = {'фішинг', 'брандмауер'} – множина термінів;
- F – операція знаходження прикладів використання терміну.

Результатом буде об'єднання прикладів для обох термінів.

Примітив "Функція" (Abstraction)

Вводяться позначення:

- $F: X \rightarrow Y$ – функція, яка перетворює елементи з множини X у елементи множини Y;
- x – вхідний елемент;
- *параметр* – набір параметрів, які регулюють поведінку функції.

Промпт-функція $F_{extract}(x, параметр)$ реалізується через інструкцію, яка передається моделі:

$$F_{extract}(x, parametr)=Promt(x,інструкція\ з\ параметр).$$

Це формальне визначення аналогічне абстракції у мовах програмування, де функція може приймати параметри для регулювання своєї логіки.

Механізм роботи примітива «Функція» полягає в тому, щоб передати LLM конкретну задачу з параметрами, які визначають деталі її виконання. Це дозволяє створювати гнучкі інструкції, які можуть бути повторно використані для різних входів.

У цьому випадку розглядається такий промпт як приклад:

«Виділи всі терміни з тексту, що належать до категорії {категорія}.»

У цьому *параметр* = {категорія} вказує, яку категорію термінів потрібно виділити, а x – текст, з якого треба виділити терміни.

У випадку, якщо категорія це "кібербезпека", модель буде виділяти лише технічні терміни, пов'язані з цією темою.

Композиція примітивів у системи, синтаксис промптів як мова

Система – це оркестрація примітивів, аналогічна програмам.

Формальна граматика полягає у тому, що промпт може бути представлений у вигляді AST (Abstract Syntax Tree):

$$Promt ::= Примітив | (Promt \oplus Promt) | Умова(Promt, Promt),$$

де $\oplus$ – композиція. Тут явно не вказано «Цикл», один базових примітивів, він включається до формули через перший член «Примітив».

До загальних принципів, яким мають задовольняють ці примітиви, відносяться:

- чіткість формулювання, кожний примітив має бути точно описаною конструкцією, щоб забезпечити однозначне розуміння моделлю;
- гнучкість, примітиви можуть бути параметризовані, що дозволяє їх адаптувати під різні завдання.
- композиція примітивів дозволяє поєднувати їх для створення складних систем, аналогічно до того, як програмісти пишуть код з базових конструкцій.

Розглянуті три примітиви («Умова», «Цикл» та «Функція») утворюють основу для безкодового створення систем через промпт-інжиніринг, дозволяючи використовувати природну мову як інструмент для управління логікою систем ШІ.

Як приклад розглянемо *задачу побудови семантичної мережі*, а саме, з формального погляду необхідно автоматично створити граф $G = (V, E)$, де V – терміни, E – зв'язки. Вирішення цієї задачі передбачає виконання ряду кроків:

Крок 1. Визначення вузлів (V)

- *Примітив*: Цикл для виділення термінів.

$$V = P_{ExtractTerms(Input)} = \bigcup_{s \in S} F_{extract}\left(s, категрія = "кібербезпека"\right)$$

Крок 2. Визначення ребер (E)

- *Примітив*: Умова для фільтрації зв'язків.

  $E=\{(u,v) | \ u,v \in V, P_{checkRelation}(u,v) = True\}$,

  де $P_{checkRelation}$ – промпт:

  «Чи пов'язані терміни {u} та {v}? Відповідь: Так/Ні.»

Крок 3. Класифікація типів ребер

- *Примітив:* Функція з параметром (тип зв'язку).

type(e)=$F_{classify}$(e, параметр={"атака","захист"})

Домен-специфічна мова у контексті безкодового підходу – це не синтаксичні конструкції, а шаблони інструкцій природною мовою, що дозволяють керувати логікою систем через структуровані діалоги з LLM. На відміну від класичних DSL, тут немає ключових слів або граматики – є лише принципи компонування промптів, які імітують програмування.

Проілюструємо принципи «мови промптів»:

1. *Модульність:*
   Кожен промпт виконує атомарну задачу, аналогічно функції.
   - Приклад (промпт "Виділення термінів"):

     «Проаналізуй текст та випиши всі ключові технічні терміни. Виведи їх у JSON-списку з поясненнями.»
2. *Композиція:*
   Промпти поєднуються послідовно, де вихід одного стає входом для іншого.
   - Приклад ланцюжка:

     [Текст] → Промпт "Виділення термінів" → [Список термінів] → Промпт "Пошук зв'язків" → [Семантична мережа]
3. *Параметризація:*
   Змінні у фігурних дужках роблять промпти гнучкими.
   - Приклад:

     «Знайди {кількість} прикладів використання терміну {термін} у контексті {тема}»
4. *Умовна логіка:*
   Рішення приймаються на основі відповідей LLM.
   - Приклад:
   - Крок 1: "Чи містить текст термін 'нейромережа'? Відповідь так/ні."
   - Крок 2: Якщо "так" → запустити промпт "Поясни нейромережу", інакше → пропустити.

Переваги «мови промптів»:

- Нульовий поріг входу, а саме, користувачу не потрібно вивчати синтаксис – достатньо описувати завдання.

- Адаптивність, яка, зокрема, полягає у тому, що легко змінити логіку через редагування текстових інструкцій.

– Інтеграція з інструментами ШІ, яка полягає у тому, що промпти можна використовувати у ChatGPT, Midjourney чи Claude, не розробляючи окремий інтерфейс.

Проблемні обмеження:

1. *Неоднозначність інструкцій.*
   - Рішення: Додавання прикладів у промпти (few-shot learning), наприклад:

     «Створи список термінів, як у прикладі:

     Приклад тексту: 'Кубіт – основа квантових обчислень.'

     Приклад відповіді: ['кубіт', 'квантові обчислення'].»
2. *Вразливість до змін у LLM.*
   - Рішення: надання шаблонів-контейнерів з параметрами, наприклад:

     «Завжди використовуй таку структуру:
     {
     'термін': '...',
     'зв'язки': [{'тип': '...', 'ціль': '...'}]
     }»

Приклади застосування

Покажемо деякі приклади застосування запропонованого фреймворку в юридичній галузі, а саме у сфері парламентського контролю і законотворчості.

Розглянемо такі задачі:

– Автоматичне виявлення ризиків, суперечностей чи корупційних загроз у тексті законопроектів;

– Підготовка резюме для депутатів щодо визначених законопроектів;

– Аналіз тексту законопроекту на предмет можливих лобістських інтересів.

Автоматичне виявлення ризиків, суперечностей чи корупційних загроз у тексті законопроектів

Створюється промпт, в якому використовується цикл для обробки статей, умови для класифікації ризиків, функції для перевірки законодавства. Здійснюється агрегація результатів через рій експертів та ітеративна оптимізація забезпечують високу точність, після чого формується вихідний звіт структурований для подальшого аналізу парламентськими комітетами.

Вхідні дані:

– Текст законопроекту D;

– База даних чинного законодавства L.

Вихідні дані:

– Звіт R, що містить:
– Списком ризиків R_{risk}.
– Суперечностями $R_{conflict}$.
– Корупційними загрозами $R_{corrupt}$.
– Рекомендаціями $R_{recommend}$.

Примітиви та їх формалізація

Примітив "Цикл" (For-Loop)

$$P_{loop}(D) = \bigcup_{s \in D} F_{analysis}(s),$$

де s – стаття/розділ законопроекту, $F_{analysis}$ – функція перевірки.

Приклад:

«Для кожної статті законопроекту виконати аналіз на наявність нечітких термінів.»

Примітив «Умова» (If-Else)

$$P_{cond}(c) = \begin{cases} Add \quad to \quad R_{risks}, \quad if \quad C_{fuzziness}(s) = True, \\ Add \quad to \quad R_{key}, \quad if \quad C_{importance}(s) = True, \\ Other, \quad otherwise. \end{cases}$$

Приклад:

«Якщо стаття містить фразу 'може бути встановлено', класифікувати як ризик корупції.»

Примітив "Функція" (Abstraction)

$$F_{expert}(s, parametr) = \begin{cases} \text{Term extraction}, \quad if \quad \text{parameter} = \text{"terminology"}, \\ \text{Compliance check}, \quad if \quad \text{parameter} = \text{"legislation"}. \end{cases}$$

Приклад:

«Функція 'Перевірка відповідності' порівнює статтю з Конституцією та законом 'Про держслужбу'.»

Композиція системи, синтаксис:

$$System = P_{loop} \oplus P_{cond} \oplus F_{expert}.$$

Етапи виконання структурованого промпту:

1. Розбивка на статті:

S = *Split*(D) (*Loop*).

2. Аналіз кожної статті:

Виділення термінів:

$$T = F_{expert}(S, "terminology").$$

Перевірка нечіткості:

$$C_{fuzziness} = True, \quad if \quad \exists t \in T : t = "\text{rational term}".$$

Перевірка суперечностей:

$$C_{contradiction}(s, L) = True, \quad if \quad s \quad contradicts \; L.$$

3. Агрегація результатів (Рій експертів):

- Використання трьох LLM для аналізу кожної статті.
- Голосування за результатами:

$$R_{final} = \arg\max_{r} \sum_{i=1}^{3} \delta(r_i, r).$$

4. Приклад конкретного промпту

Задача:
"Проаналізуй законопроект на предмет корупційних загроз."

Промпт:

«Крок 1: Розбий текст на статті.

Крок 2: Для кожної статті:

– Виділи терміни (наприклад, 'дискреційні повноваження').

– Перевір відповідність Конституції (статті 8, 28).

– Якщо є фраза 'без обґрунтування', додай до Rcorrupt.

Крок 3: Звіт:

– Таблиця з стовпцями: Стаття, Проблема, Рекомендація.»

Вихідний звіт:

СТАТТЯ	ПРОБЛЕМА	РЕКОМЕНДАЦІЯ
Ст. 5	Дискреційні повноваження без критеріїв	Додати чіткі критерії прийняття рішень
Ст. 12	Суперечність із Законом "Про бюджет"	Вирівняти терміни з бюджетним кодексом

5. Оптимізація через градієнтний спуск

Постановка:

$$d(R_{final}, R_{target}) \to min,$$

де R_{target} – еталонний звіт.

Ітерація 1:

- Промпт: "Виділи терміни" → результат: 70% точності.
- Корекція: "Виділи терміни, пов'язані з корупцією (наприклад, 'тендер', 'дискреція')" → 90% точності.

Візуалізація (Gephi)

Промпт для генерації CSV:

«Створи nodes.csv (стовпці: Id, Label) та edges.csv (Source, Target, Type) для вузлів 'Ризик', 'Суперечність' та зв'язків між ними.»

Підготовка резюме для депутатів щодо визначених законопроектів

Відповідний промпт має використовувати цикл для обробки статей, умови для виділення ризиків, функції для структуризації виводу. Агрегація результатів через рій експертів (наприклад, 3 LLM) забезпечує точність. Вихідне резюме має бути адаптоване для депутатів: коротке, з акцентом на ключові аспекти та проблемні зони.

Необхідно створити стисле, чітке та зрозуміле резюме законопроекту для депутатів, виділяючи:

- основну мету;
- ключові положення;
- потенційні ризики/суперечності;
- рекомендації щодо доопрацювання.

Вхідні дані:

- текст законопроекту *D*;
- база даних чинного законодавства *L*.

Вихідні дані – резюме *R* у форматі таблиці/тексту з розділами:

- мета;
- ключові положення;
- ризики;
- рекомендації.

Примітиви та їх формалізація

Примітив "Цикл" (For-Loop)

$$P_{loop}(D) = \bigcup_{s \in D} F_{analysis}(s),$$

де s – стаття/розділ законопроекту, $F_{analysis}$ – функція виділення ключових елементів.

Приклад:
«Для кожної статті законопроекту виділи основну вимогу та санкції.»

Примітив "Умова" (If-Else)

$$P_{cond}(c) = \begin{cases} Add \quad to \quad R_{risks}, \quad if \quad C_{fuzziness}(s) = True, \\ Add \quad to \quad R_{key}, \quad if \quad C_{importance}(s) = True, \\ Other, \quad otherwise. \end{cases}$$

Приклад:

«Якщо стаття містить термін 'може бути встановлено', класифікувати як ризик.»

Примітив "Функція" (Abstraction)

$$F_{summary}(D, parametr) = \begin{cases} \text{Concise description}, \quad if \quad \text{parameter} = \text{"goal"}, \\ \text{Table of key provisions}, \quad if \quad \text{parameter} = \text{"structure"}. \end{cases}$$

Приклад:

«Створи таблицю зі стовпцями: 'Стаття', 'Вимога', 'Відповідальний орган'.»

Композиція системи

$$System = P_{loop} \oplus P_{cond} \oplus F_{summary}.$$

Етапи:

1. Розбивка на статті:

$$S = Split(D)\ (Loop).$$

2. Аналіз кожного розділу:

– Виділення мети:

$$Goal = F_{summary}(D, \text{"}goal\text{"}).$$

– Ключові положення:

$$Key = \bigcup_{s \in S} P_{cond}(s).$$

– Ризики:

$$R_{risks} = \left\{ s \mid C_{fuzziness}(s) \vee C_{contradiction}(s, L) \right\}.$$

3. Формування резюме:

Використання $F_{summary}$ для створення структурованого виводу.

4. Приклад конкретного промпту

Задача:

«Підготувати резюме законопроекту 'Про екологічний нагляд' для депутатів.»

Промпт:

«Крок 1: Виділи мету законопроекту (перші 3 статті).

Крок 2: Для кожної статті:

– Визнач основну вимогу.

– Якщо є санкції, додай до таблиці "Ключові положення".

– Якщо термін не визначено чітко (наприклад, 'екологічна безпека'), додай до розділу "Ризики".

Крок 3: Перевір відповідність із Законом "Про охорону навколишнього середовища".

Крок 4: Створи резюме у форматі:

– Мета: [короткий опис].

– Ключові положення: таблиця.

– Ризики: список.

– Рекомендації: [пропозиції щодо доопрацювання].»

Вихідний результат:

РОЗДІЛ	**ЗМІСТ**
Мета	Забезпечення екологічної безпеки через посилення нагляду за підприємствами.
Ключові положення	
– Стаття 5	Обов’язкова сертифікація для промислових об’єктів.
– Стаття 8	Штрафи за порушення екологічних норм (від 100 000 грн).
Ризики	Нечітке визначення "екологічної безпеки" (стаття 2).
Рекомендації	Уточнити критерії сертифікації та зв’язати їх із міжнародними стандартами.

5. Оптимізація через градієнтний спуск

Постановка задачі:

$$d(R_{summary}, R_{etalon}) \rightarrow min,$$

де R_{etalon} – зразок резюме від експертів.

Промпт:

«Виділи мету» → результат: занадто загальний.

Корекція:

«Виділи мету, вказавши конкретні цілі (наприклад, 'знизити викиди на 20%')».

6. Візуалізація (Gephi)

Промпт для генерації CSV:

«Створи файл nodes.csv (стовпці: Id, Label) для вузлів 'Мета', 'Ризик', 'Ключове положення'.»

Аналіз тексту законопроекту на предмет можливих лобістських інтересів

У промпті, що створюється, мають використовуватись: цикл для обробки статей, умови для виявлення лобістських ознак, функції для перевірки законодавства. Агрегація через рій експертів та ітеративна оптимізація мають забезпечувати високу точність. Вихідний повинен звіт містити конкретні рекомендації для парламентських комітетів.

Мета:

Виявити в тексті законопроекту:

- норми, що надають переваги конкретним групам/суб'єктам;
- нечіткі формулювання, які можуть бути використані для лобіювання;
- конфлікти інтересів (наприклад, відсутність вимог до прозорості).

Вхідні дані:

- текст законопроекту D;
- база даних лобістських практик L (наприклад, "закони з винятками для певних галузей").

Вихідні дані – звіт R із:

- Списком підозрілих норм $R_{lobbying}$.
- Аналізом конфліктів $R_{conflict}$.
- Рекомендаціями $R_{recommendations}$.

Примітиви та їх формалізація

Примітив "Цикл" (For-Loop)

$$P_{loop}(D) = \bigcup_{s \in D} F_{analysis}(s),$$

де s – стаття/розділ законопроекту, $F_{analysis}$ – функція перевірки на лобістські ознаки.

Приклад:

«Для кожної статті законопроекту перевірити наявність фраз 'спеціальні умови', 'виключення для'.»

Примітив "Умова" (If-Else)

$$P_{cond}(c) = \begin{cases} Add \quad to \quad R_{lobbying}, \quad if \quad C_{lobbying}(s) = True, \\ Add \quad to \quad R_{conflict}, \quad if \quad C_{conflict}(s) = True, \\ Other, \quad otherwise. \end{cases}$$

Приклад:

«Якщо стаття містить 'пільги для суб'єктів галузі X', класифікувати як лобістську норму.»

Примітив "Функція" (Abstraction)

$$F_{expert}(s, parameter) = \begin{cases} Term\ analysis, \quad if \quad parameter = "terminology", \\ Conflict\ checking, \quad if \quad parameter = "legislation". \end{cases}$$

Приклад:

«Перевірити, чи стаття 5 суперечить Закону 'Про прозорість лобіювання'.»

Композиція системи

$$System = P_{loop} \oplus P_{cond} \oplus F_{expert}.$$

Етапи:

1. Розбивка на статті:

$$S = Split(D)\ (Loop).$$

2. Аналіз кожної статті:

Виділення підозрілих термінів:

$$T = F_{expert}(s, "terminology").$$

Перевірка лобістських ознак:

$$C_{lobbying}(T) = True, \quad if \quad \exists t \in T : t = "Exemption\ for\ group\ A".$$

Перевірка конфліктів:

$$C_{conflict}(s, L) = True, \quad if \quad "s \quad contradicts \; L".$$

3. Агрегація результатів (Рій експертів):

Використання трьох LLM для аналізу кожної статті.

Голосування за результатами:

$$R_{final} = \arg\max_{r} \sum_{i=1}^{3} \delta(r_i, r).$$

4. Приклад конкретного промпту

Задача:

«Проаналізувати законопроект 'Про регулювання енергетики' на предмет лобістських інтересів.»

Промпт:

«Крок 1: Розбий текст на статті.

Крок 2: Для кожної статті:

– Виділи терміни типу 'пільги', 'виключення', 'субсидії'.

– Якщо є посилання на конкретні компанії/галузі, додай до R_лобі.

– Перевір відповідність Закону "Про запобігання корупції" (стаття 14).

Крок 3: Звіт:

– Таблиця з стовпцями: Стаття, Підозріла норма, Конфлікт, Рекомендації.»

Вихідний звіт:

СТАТТЯ	ПІДОЗРІЛА НОРМА	КОНФЛІКТ	РЕКОМЕНДАЦІЇ
Ст. 7	«Пільги для вугільних компаній»	Суперечить ст. 14 Закону «Про запобігання корупції»	Вилучити посилання на конкретні компанії
Ст. 12	«Виключення для суб'єктів з Донеччини»	Немає	Уточнити критерії регіональних виключень

5. Оптимізація через градієнтний спуск

Постановка задачі:

$$d(R_{final}, R_{etalon}) \to min,$$

де R_{final} – фінальний звіт; R_{etalon} – еталонний звіт.

Промпт: "Виділи пільги" → результат: 60% точності.

Корекція: "Виділи пільги, пов’язані з конкретними галузями (енергетика, видобуток)" → 85% точності.

6. Візуалізація (Gephi)

Промпт для генерації CSV:

> «Створи nodes.csv (стовпці: Id, Label) для вузлів 'Лобістська норма', 'Конфлікт' та зв’язків між ними.»

Розроблений безкодовий фреймворк промпт-інжинірингу надає формалізацію роботи з системами штучного інтелекту, який базується на таких аспектах:

Основна ідея роботи полягає у представленні промптів як математично формалізованих аналогів базових конструкцій мов програмування (умов, циклів, функцій). Це перетворює випадкові текстові запити у структуровані логічні примітиви, які можна компонувати, оптимізувати та аналізувати систематично. Наприклад, умова (If-Else) отримує вигляд математичної функції з вхідними параметрами та висхідними діями, цикл (For-Loop) моделюється як оператор множинної обробки, функція (Abstraction) визначається через параметризований процес перетворення даних. Це дозволяє створювати складні логічні системи через текстові інструкції, що відповідає парадигмі «програмування через приклади» (PBE), але з більшою точністю та контролем.

Запропонована «мова промптів» відрізняється від традиційних домен-специфічних мов DSL своєю адаптивністю та відсутністю жорсткого синтаксису. Вона має властивості модульності, параметризованості і ком позиційності. В рамках цієї мови кожен промпт виконує атомарну задачу (наприклад, «виділити терміни» або «побудувати зв'язки»), змінні у фігурних дужках дозволяють адаптувати логіку без зміни основної структури, крім того, промпти можна поєднувати у ланцюжки або графи завдань, аналогічно функціям у програмах.

Це знижує поріг входу до програмування з боку користувачів, оскільки не вимагає навичок програмування, але забезпечує гнучкість для розробки складних систем.

Фреймворк впроваджує інноваційні техніки, які розширюють можливості промпт-інжинірингу, а саме градієнтний спуск у просторі інструкцій і модель «рою віртуальних експертів».

Метод градієнтного спуску формально моделює промпти як оптимізаційну задачу, де метрика якості мінімізується шляхом ітеративних корекцій тексту. Це дозволяє автоматично покращувати ефективність систем, як це робиться в машинному навчанні.

Агрегація результатів декількох LLM (голосування або середнє значення) в рамках концепції «рою віртуальних експертів» зменшує ризик помилок, характерних для окремих моделей. Цей підхід відповідає принципам ансамблів у машинному навчанні, але адаптований для безкодового взаємодії з AI.

Аналогія з концепцією С. Вольфрама про складність з простих правил є ключовою для новизни. Як клітинні автомати генерують нелінійні динамічні системи з базових структур, так і пропонований фреймворк дозволяє формувати складні системи ШІ через комбінацію трьох примітивів. Це створює новий теоретичний фреймворк для розуміння, як текстові інструкції можуть імітувати програмування без коду.

Від існуючих рішень, фреймворк, який пропонується, має такі відмінності:

– Запропонована «мова промптів» на відмінність від класичного DSL не вимагає навчитися нових ключових слів або граматики, а лише використовує шаблони природної мови.

– Математична формалізація на відміну від традиційного вже промпт-інженерингу, забезпечує статистичну узгодженість результатів, відокремлюючи їх від «чорного ящика» LLM.

– Композиція примітивів через текст на відміну від класичного кодування виражає логіку систем більш зрозуміло для непрограмістів, ніж SQL або Python.

Запропонований фреймворк змінює парадигму взаємодії з AI з «чорного ящика» на структурований, формалізований процес. Його новизна полягає в злитті математичної точності, ефективності безкодового підходу та можливості масштабування через композицію примітивів. Це не лише інструмент для розробки, але і нова дисципліна – «логіка промптів», яка відкриває шляхи до AI, управління яким доступне кожному.

2.3. Розширений фреймворк: стан, обробка винятків, посилання між модулями

Базові логічні примітиви – такі як *умова*, *цикл*, *функція* – є необхідною основою для побудови будь-якої виконуваної системи. Однак для моделювання складних правових процесів, особливо в умовах парламентського контролю, застосування тільки цих примітивів виявляється недостатнім для проведення зручної, прозорої аналітичної роботи. Потрібен розширений фреймворк, здатний відображати динаміку, гнучкість та складність реальних регуляторних напрямків.

Для роботи зі складними аналітичними, прогнозними або рекурсивними завданнями необхідно розширити фреймворк новими примітивами:

– «*Мітка*» – для позначення точок входу/виходу;

– «*Перехід*» – для управління порядком виконання.

Ці примітиви наближають безкодове програмування до парадигми процедурного програмування, забезпечуючи:

– чіткий контроль над логічним потоком;
– можливість повторного використання блоків;
– гнучкість у проектуванні складних сценаріїв.

Формалізація розширених примітивів

Примітив 4. Мітка (Label)

Нехай:

– *L* – ім’я мітки;
– *Block* – блок інструкцій.

Тоді:

Label(*L, Block*) ::= (*L: Block*).

Приклад:

Label(*INIT*):

Вхід: *X* = [4, 7, 10, 5, 3]

Така конструкція дозволяє явно позначити початковий стан системи.

Примітив 5. Перехід (Goto)

Нехай:

– *L* – ім’я метки для переходу;
– *Condition* – необов’язкова умова.

Тоді:

Goto(*L, Condition*) ::= if(*Condition*) then goto *L.*

Приклад:

Label(CHECK_ZERO):

Якщо sum_denominator == 0:

Goto(HANDLE_ERROR)

Це дозволяє здійснювати стрибки в логічній структурі, що особливо корисно для обробки винятків або вкладених розгалужень.

Композиція примітивів

Базові та розширені примітиви можуть комбінуватися для побудови складних логічних конструкцій:

Promt ::= *Примітив*
| (*Promt* ⊕ *Promt*)
| *Умова*(*Promt, Promt*)
| *Label*(*L, Promt*)
| *Goto*(*L*),

де ⊕ – операція композиції.

Приклади реалізації задач

Приклад 1: Обчислення середнього значення

Задача: Обчислити середнє значення послідовності чисел [4, 7, 10, 5, 3].

Промпт з використанням базових примітивів:

```
[Примітив "Функція"] Функція(ComputeMean):
  [Примітив "Вхід"] Вхід: X = [4, 7, 10, 5, 3]
  [Примітив "Цикл"] Обчисли суму елементів.
  [Примітив "Умова"] Якщо кількість елементів > 0:
    Поділи суму на кількість елементів.
  Інакше:
    Поверни помилку: "Порожній список".
```

Промпт на основі розширеного фреймворку:

```
Label(INIT):
 Вхід: X = [4, 7, 10, 5, 3]
Label(CALC_SUM):
 Підсумуй елементи X.
Label(DIVIDE_BY_N):
 Поділи суму на довжину X.
Goto(INIT)
```

Приклад 2: Обчислення дисперсії

Задача: Обчислити дисперсію для послідовності: [4, 7, 10, 5, 3].

Промпт з використанням базових примітивів:

«Функція(ComputeVariance):

Обчисли середнє μ.

Цикл:

Для кожного xi в X обчисли $(xi - \mu)^2$.

Підсумуй всі квадрати.

Поділи на n.»

Розширений фреймворк:

```
Label(INIT):
  Вхід: X = [4, 7, 10, 5, 3]
Label(MEAN):
  Обчисли середнє значення μ.
Label(DEVIATIONS):
  Для кожного x_i в X:
    Обчисли (x_i – μ)². Збережи як D.
Label(SUM_SQUARES):
  Підсумуй елементи D.
Label(VARIANCE):
  Поділи суму на n.
Goto(INIT)
```

Приклад 3: Обчислення автокореляції

Задача: Обчислити автокореляцію з лагом 1 для послідовності [4, 7, 10, 5, 3].

Базовий фреймворк (з явним використанням примітивів):

```
[Примітив "Функція"] Функція(ComputeAutocorrelation):
  [Примітив "Цикл"] Обчисли середнє μ.
  [Примітив "Цикл"] Для кожного x_i в X обчисли (x_i – μ)².
  [Примітив "Цикл"] Підсумуй всі квадрати.
  [Примітив "Умова"] Якщо знаменник ≠ 0:
    Поділи чисельник на знаменник.
  Інакше:
    Поверни 0.
```

Розширений фреймворк:

```
Label(INIT):
```

```
  Вхід: X = [4, 7, 10, 5, 3], k = 1
Label(MEAN):
  Обчисли середнє μ.
Label(DEVIATIONS):
  Для кожного x_i в X:
    Обчисли z_i = x_i – μ. Збережи як Z.
Label(NUMERATOR):
  Ініціалізуй sum_numerator = 0
  Для t від 0 до len(X)-k-1:
    sum_numerator += Z[t] * Z[t + k]
Label(DENOMINATOR):
  Ініціалізуй sum_denominator = 0
  Для кожного z в Z:
    sum_denominator += z²
Label(AUTOCORRELATION):
  Якщо sum_denominator ≠ 0:
    R = sum_numerator / sum_denominator
  Інакше:
    R = 0
Goto(INIT)
```

Приклад 4: Грабіжник і шериф

Задача: приклад з книги Кнута[59].

Потрібно написати сценарій розвитку ситуації. Ситуація така: грабіжник заходить у порожній будинок, де п'є і краде. За вікном – шериф, вони помічають один одного, після чого починається стрілянина. Дії акторів випадкові, імовірність промахів зростає з кількістю випитого та кількістю промахів. Усе закінчується вбивством одного або двох героїв.

Треба на основі наведеної методики безкодового програмування розробити структурований промпт для відтворення розвитку подій – написання сценарію.

[59] Knuth D.E. The Art of Computer Programming, Volume. 2: Seminumerical Algorithms. Addison-Wesley, 1997.

На рис. 1 наведено блок-схему цієї без кодової програми.

```
ПОЧАТОК

Початкові умови:
  - Злочинець входить до порожнього будинку
  - Шериф за вікном
  - спожито_спиртного = 0
  - промахи = 0

ЦИКЛ:
  ДЛЯ КОЖНОГО персонажу (Злочинець, Шериф):

  ┌──────────────────────────────────────────┐
  │ Дія Злочинця                             │
  │ Якщо спожито_спиртного > 5:              │
  │     Стріляти без розсуду (випадково)     │
  │ Інакше:                                  │
  │     Спробувати цілитися обережно         │
  └──────────────────────────────────────────┘

  ┌──────────────────────────────────────────┐
  │ Дія Шерифа                               │
  │ Якщо кількість промахів > 3:             │
  │     Стріляти без розсуду (нервово)       │
  │ Інакше:                                  │
  │     Чекати кращий момент для пострілу    │
  └──────────────────────────────────────────┘

  ┌──────────────────────────────────────────┐
  │ Перевірка результату                     │
  │ Якщо смертельна травма → ПЕРЕЙТИ ДО КІНЦЯ │
  │ Інакше → оновити статистику              │
  └──────────────────────────────────────────┘

КІНЕЦЬ:
Один або обидва персонажі мертві

РЕЗЮМЕ:
"Так завжди закінчується, коли ти вирішаєш грати
під впливом спиртного."
```

Рисунок 1. Блок-схема програми генерації сценарію

Формалізація прикладу

1. Цикл

Нехай:

- $S = \{s_1, s_2, ..., s_n\}$ – множина можливих дій («п'є», «стріляє», «ховається»);
- F – функція, що описує результат дії для кожного героя.

Тоді:

- $P(S) = \bigcup F(s_i)$, для $i = 1...n$.

2. Умова

Нехай:

- Input – поточний стан (наприклад, кількість випитого, кількість промахів);
- C – умова, наприклад, "кількість випитого > 5";
- A1, A2 – дії.

Тоді:

- P(Input) = A1, якщо C(Input) = True
- P(Input) = A2, якщо C(Input) = False

3. Функція

Нехай:

- F(x, параметр) – функція, що описує дію героя;
- x – вхідний елемент (наприклад, «грабіжник»);
- параметр – набір параметрів.

Тоді:

- F(x, параметр) = Prompt(x, інструкція з параметром)

4. Метка

Нехай:

- L – ім’я метки;
- Block – блок інструкцій.

Тоді:

- Label(L, Block) ::= (L: Block)

5. Перехід

Нехай:

- L – ім’я метки для переходу;
- Condition – необов’язкова умова.

Тоді:

- Goto(L, Condition?) ::= if(Condition) then goto L

Промпт

Сценарій: Грабіжник vs Шериф

Початкові умови:

- Грабіжник заходить в пустий будинок.

- Шериф знаходиться за вікном.

- Початковий стан: кількість випитого = 0, кількість промахів = 0.

Цикл подій:

Label("Start"):

Цикл:

Для кожного героя (Грабіжник, Шериф):

Дія грабіжника

Якщо (кількість випитого > 5):

Грабіжник стріляє навмання.

Інакше:

Грабіжник намагається прицілитися.

Дія шерифа

Якщо (кількість промахів > 3):

Шериф нервово стріляє.

Інакше:

Шериф чекає моменту для точного пострілу.

Перевірка результату

Якщо (герой отримав смертельну поранення):

Goto("End").

Інакше:

Збільшити кількість випитого або промахів.

Label("End"):

Завершення сценарію: один або обидва герої загинули.

Пояснення:

- Цикл описує повторювані дії героїв, поки не настане фінал.
- Умова визначає, чи дія є розумною або глупою, залежно від стану.
- Функція дозволяє налаштовувати поведінку героїв через параметри.
- Метки та переходи організовують ключові моменти сценарію.

Нижче наведено приклад виконання розробленого промпту у відповідності із логікою базових примітивів:

Сценарій: Грабіжник vs Шериф

Дія 1: Початок

– Грабіжник заходить в пустий будинок.
  Він швидко оглядає простір: порожньо, лише на столі стоїть пляшка високоградусного алкогольного напою. Грабіжник з задоволенням хапає її і починає пити.

Кількість випитого = 1.

Кількість промахів = 0.

– Шериф знаходиться за вікном.
  Він помічає силует грабіжника через фіранку. Шериф тихо достає зброю і чекає, спостерігаючи за ситуацією.

Дія 2: Перша взаємодія

– Цикл подій починається.

Грабіжник:

– Якщо (кількість випитого ≤ 5):
  Грабіжник намагається прицілитися до вікна, де бачить рух. Однак його руки ще не затремтіли від алкоголю, і він стріляє майже точно.

Результат: Промах.

Кількість промахів = 1.

Шериф:

– Якщо (кількість промахів ≤ 3):
  Шериф вирішує не поспішати. Він чекає, коли грабіжник зробить ще один крок уперед, щоб мати кращий кут для пострілу.

Дія 3: Ескалація

Грабіжник:

– Якщо (кількість випитого = 2):
  Грабіжник випиває ще одну порцію. Тепер його рухи починають бути менш точними. Він знову стріляє у напрямку вікна, але цього разу його куля летить у стелю.

Результат: Промах.
Кількість промахів = 2.
Шериф:

– Якщо (кількість промахів ≤ 3):
  Шериф вирішує діяти. Він робить перший постріл, але через те, що грабіжник раптово ховається за меблями, куля теж не досягає цілі.

Результат: Промах.
Кількість промахів шерифа = 1.

Дія 4: Напруження зростає

Грабіжник:

- Якщо (кількість випитого = 4):
  Грабіжник випиває ще одну порцію. Тепер його поведінка стає хаотичною. Він стріляє навмання, навіть не дивлячись у напрямку вікна. Одна з куль розбиває дзеркало на стіні.

Результат: Промах.

Кількість промахів = 3.

Шериф:

- Якщо (кількість промахів шерифа = 1):
  Шериф намагається зберегти спокій, але його терпіння на межі. Він робить ще один постріл, але цього разу куля застряє в дерев'яній рамі вікна.

Результат: Промах.

Кількість промахів шерифа = 2.

Дія 5: Фінал

Грабіжник:

- Якщо (кількість випитого > 5):
  Грабіжник втрачає контроль. Він стріляє навмання, навіть не підозрюючи, де знаходиться шериф. Одна з куль влучає у стіну поруч із шерифом, але не завдає шкоди.

Шериф:

- Якщо (кількість промахів шерифа > 3):
  Шериф втрачає терпіння. Він робить останній, відчайдушний постріл. Цього разу його куля влучає точно в грабіжника.

Результат: Грабіжник отримує смертельну поранення.

Goto("End").

Завершення сценарію

Label("End"):

- Грабіжник падає на підлогу, його рука неживо випускає зброю. Шериф з обережністю входить у будинок. Він дивиться на мертвого грабіжника, перевіряє пляшку з алкоголем і зітхає.

Фінальна репліка шерифа:

– «Так завжди закінчується, коли ти вирішуєш грати в героя під дією алкоголю.»

Підсумок

Сценарій завершується смертю грабіжника. Логіка базових примітивів (цикл, умова, функція) дозволила створити динамічну історію, де дії героїв залежали від їх стану та ситуації.

Як можна бачити, розширений фреймворк безкодового програмування на основі застосування сучасних великих мовних моделей значно спрощує процедуру програмування, розвиваючи можливості, закладені класиками комп'ютерних наук.

Методологія створення остаточного промпту

Крок 1. Словесне формулювання проблеми – отримання первинного промпту

На цьому етапі формулюється задача природною мовою, наприклад:

«Обчисли середнє значення списку чисел.»

Це є первинним промптом, який може бути неточним або недостатньо структурованим для LLM.

Крок 2. Коригування первинного промпту

Аналізується можлива неоднозначність, додаються уточнення, приклади (few-shot learning):

«Обчисли середнє значення списку чисел. Якщо список порожній – надай повідомлення про помилку. Наприклад: для [4, 7, 10] середнє = 7.»

Крок 3. Формування структурованого промпту на основі фреймворку

Промпт переписується з використанням примітивів:

```
[Примітив "Функція"] Функція(ComputeMean):
[Примітив "Вхід"] Вхід: X = [4, 7, 10, 5, 3]
[Примітив "Цикл"] Обчисли суму елементів.
[Примітив "Умова"] Якщо кількість елементів > 0:
   Поділи суму на кількість елементів.
Інакше:
   Поверни помилку: "Порожній список".
```

Крок 4. Виконання промпту на контрольному прикладі у інтерактивному режимі

Промпт передається LLM, тестується на прикладі [4, 7, 10, 5, 3].
Перевіряється, чи результат дорівнює 5.8.

Крок 5. Остаточне коригування структурованого промпту

За результатами тестування вносяться корективи. Наприклад, якщо модель не враховує умову про порожній список, промпт доповнюється:

[Примітив "Умова"] Якщо список порожній:

Поверни: "Помилка: список порожній"

Інакше:

Продовж обчислення

Переваги «мови промптів»

До переваг «мови промптів» над традиційним без кодовими технологіями відносяться:

- Нульовий поріг входу, а саме, користувачу не потрібно вивчати синтаксис – достатньо описувати завдання.
- Адаптивність, яка, зокрема, полягає у тому, що легко змінити логіку через редагування текстових інструкцій.
- Інтеграція з інструментами ШІ, яка полягає у тому, що промпти можна використовувати у ChatGPT, Midjourney чи Claude, не розробляючи окремий інтерфейс.

Обмеження і їх урахування

До проблемних обмежень застосування «мови промптів» відносяться:

Неоднозначність інструкцій.

- Рішення: Додавання прикладів у промпти (few-shot learning), наприклад:

```
Створи список термінів, як у прикладі:
Приклад тексту: 'Кубіт – основа квантових обчислень.'
Приклад відповіді: ['кубіт', 'квантові обчислення'].
```

Вразливість до змін у LLM.

- Рішення: надання шаблонів-контейнерів з параметрами, наприклад, подібну до формату JSON:

```
Завжди використовуй таку структуру:
```

```
{
   'термін': '...',
   'зв’язки': [{'тип': '...', 'ціль': '...'}]
}
```

Базовий фреймворк безкодового програмування, заснований на примітивах «Умова», «Цикл» і «Функція», забезпечує чітку структуру для побудови логічних інструкцій. Ці примітиви вже дозволяють автоматизувати масову обробку даних, аналіз тексту, вирішення простих математичних задач.

Однак для реалізації складних алгоритмів, таких як аналіз часових рядів, обробка винятків, рекурсивні сценарії, необхідно ввести додаткові примітиви «Мітка» та «Перехід».

Крім того, представлено методологію створення остаточного промпту як результату промпт-інженірингу: словесне формулювання, корекція, структурування, тестування та фінальне коригування.

Подальші дослідження передбачають:

- Створення графічного інтерфейсу для візуального проектування промптів.
- Розробку механізмів автоматичної валідації логічної цілісності.
- Інтеграцію з інструментами для аналізу даних .

2.4. Агентська система

Безкодове програмування – це парадигма, що дозволяє створювати функціональність без написання коду, використовуючи візуальні інтерфейси або природну мову[60]. Фреймворк[61] запропоновує структурний підхід до побудови поведінки через примітиви: Condition, Loop, Function, Label, Goto. Ці елементи формують базу для логічного управління процесами, аналогічно до класичних конструкцій мов програмування, але через псевдокод природною мовою.

Такий підхід може бути застосований як в навчальних цілях, так і у реальних системах аналізу, де користувачі потребують високої адаптованості до нетехнічних користувачів.

[60] El Kamouchi, Hind, Mohamed Kissi, and Omar El Beggar. "Low-code/No-code Development: A systematic literature review." In 2023 14th International Conference on Intelligent Systems: Theories and Applications (SITA), pp. 1-8. IEEE, 2023.

[61] Dmitry Lande, Leonard Strashnoy. An Advanced No-Code Programming Framework for Complex Problems in LLM Environments. ResearchGate Preprint: DOI: 10.13140/RG.2.2.25307.89129 (May, 2025). 14 pp.

Агентна модель давно використовується в галузі штучного інтелекту[62, 63, 64], особливо в задачах прогнозування, обробки даних, автоматизованого прийняття рішень.

Класична агентна система передбачає такі властивості кожного агента, як його власну роль, внутрішній стан, наявність сенсорів (вхід), актуаторів (вихід), комунікацію з іншими агентами.

AgentFlow вводить нову версію агентної моделі, в якій агенти створюються через природну мову, причому для цього не вимагається технічна реалізація, взаємодія відбувається через структуровані промпти, виконання здійснюється LLM, як середовищем системи програмування.

Хоча LLM має послідовну архітектуру, через правильне оформлення промптів можна імітувати паралельне виконання завдань[65]. Це досягається через явну вказівку кожної гілки виконання, виконання окремих частин як самостійних сутностей, об'єднання результатів в одному фінальному вузлі – модераторі.

В AgentFlow цей принцип реалізований через структурований псевдокод, що дозволяє контролювати логіку та формувати чіткий вихід.

Система пам'яті в агентних моделях дозволяє зберігати минулі завдання, аналізувати помилки, покращувати виконання завдань з часом. У рамках Machine Learning це відповідає системам з нагородами/штрафами, де стан агента формується на основі минулих взаємодій[66].

AgentFlow використовує внутрішню пам'ять агентів[67], яка зберігає минулі завдання, формує статистику точності, використовується для адаптації поведінки. Це є певним кроком у напрямку до самоорганізації агентів, коли вони не просто виконують завдання, а й навчаються на досвіді.

Агент – це сутність, яка має роль, зберігає внутрішній стан, реагує на події,

виконує дії, може спілкуватися з іншими агентами.

---

[62] Li, Xinyi, Sai Wang, Siqi Zeng, Yu Wu, and Yi Yang. "A survey on LLM-based multi-agent systems: workflow, infrastructure, and challenges." Vicinagearth 1, no. 1 (2024): 9.

[63] Amirkhani, Abdollah, and Amir Hossein Barshooi. "Consensus in multi-agent systems: a review." Artificial Intelligence Review 55, no. 5 (2022): 3897-3935.

[64] Han, Shanshan, Qifan Zhang, Yuhang Yao, Weizhao Jin, Zhaozhuo Xu, and Chaoyang He. "LLM multi-agent systems: Challenges and open problems." arXiv preprint arXiv:2402.03578 (2024).

[65] Kim, Sehoon, Suhong Moon, Ryan Tabrizi, Nicholas Lee, Michael W. Mahoney, Kurt Keutzer, and Amir Gholami. "An llm compiler for parallel function calling." In Forty-first International Conference on Machine Learning. 2024.

[66] Shakya, Ashish Kumar, Gopinatha Pillai, and Sohom Chakrabarty. "Reinforcement learning algorithms: A brief survey." Expert Systems with Applications 231 (2023): 120495.

[67] Zhang, Zeyu, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Quanyu Dai, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. "A survey on the memory mechanism of large language model based agents." arXiv preprint arXiv:2404.13501 (2024).

Формально можна позначити:

$$a = \langle R, S, L, C \rangle,$$

де R це роль, S – стан, L – логіка, C – правила обміну даними.

Структуру агента можна визначити таким чином:

```
ТИП: АГЕНТ
ID: {{унікальне_ім'я}}
РОЛЬ: {{опис_ролі}
ЛОГІКА:
  {{Condition, Loop, Function}}
СТАН:
  {{ключ-значення, пам'ять, статистика}}
КОМУНІКАЦІЯ:
  Надсилати_до: {{інший_агент_ID}}
```

Рій агентів

Рій – це система взаємодіючих агентів, які одночасно чи послідовно вирішують частини загального завдання.

Формалізуємо це позначення:

- $A = \{a_1, a_2, ..., a_n\}$ – множина агентів,
- T_i – завдання агента a_i,
- M_{ij} – передача результату $a_i \rightarrow a_j$,
- O_i – вихід агента a_i.

Наведемо приклад рою:

```
ТИП: РІЙ
НАЗВА: аналіз_новини
АГЕНТИ:
  – сумаризатор_001
  – фактчекер_002
  – емоційний_аналітик_003
ЛОГІКА:
  Запустити_паралельно([сумаризатор_001, фактчекер_002])
  ЯКЩО отримано_результати ТО:
    ВИКЛИК сантимент_аналітик_003 → проаналізувати_результати()
```

Паралельне виконання через псевдокод

LLMs насправді не виконує дії справді, але може імітувати паралельність через явне описання гілок. Тому в рамках фреймворку вводиться спеціальний примітив:

Запустити_паралельно([...])

ЛОГІКА:

Запустити_паралельно([*пошуковик_001, пошуковик_002*])

Об'єднати_результати()

Формальний запис виклику цього примітиву наступний:

«Виконати_паралельно $(A)=\{O(a_i)\}_{i=1}^{n}$, *де* $O(a_i)$ *– це результати діяльності окремих агентів* a_i*.»*

Стан агента (Memory)

Стан агента – це історія його минулої поведінки, яка використовується для адаптації та навчання, наприклад:

СТАН:

- база_фактів = {... }
- статистика_точності = 92%

Формальне представлення стану агентів:

$$S = \{(k_1,v_1),(k_2,v_2),...,(k_m,v_m)\},$$

де k_i – ключ, v_i – значення.

Комунікація між агентами

Комунікацію між окремими агентами можна представити у вигляді таблиці:

ПРИМІТИВ	ОПИС
Надсилати_повідомлення_до (агент_ID)	Вказує шлях передачі
Отримати_від (агент_ID)	Очікує результат від іншого агента
Відповісти_до (агент_ID)	Відправити назад

Формальне визначення акту комунікації:

$$C(a_i, a_j) = Message(a_i \rightarrow a_j),$$

де a_i – це відправник, a_j – отримувач.

Звичайні промпти, створені людиною можуть перетворюватись у структуровані у відповідності із запропонованим фреймворком також із застосуванням великих мовних моделей. Наприклад, завантаживши текст фреймворку як звичайний документ, можна створювати структуровані промпти.

Приклад роботи рою у кореляційному завданню

Задача полягає у наступному: необхідно виявити, який з трьох часових рядів найменш корельований з двома іншими, ми пишемо відповідний для створення структурованого агентного промпту:

«Нехай треба розрахувати взаємні кореляції 3 часових рядів по 30 елементів кожний, а потім виявити, який ряд суттєво відрізняється від інших. Напиши агентський промпт за цією методикою – фреймворком.»

Вхідні дані мають такий вигляд:

Дані:

Ряд A (валютний курс): [36.5, 36.4, 36.2, 36.3, 36.1, 36.0, 36.1, 36.3, 36.5, 36.7, 36.8, 36.9, 36.8, 36.7, 36.6, 36.5, 36.4, 36.3, 36.2, 36.1, 36.0, 36.1, 36.2, 36.3, 36.4, 36.5, 36.6, 36.7, 36.8, 36.9]

Ряд B (курс криптовалюти): [28000, 28100, 28200, 28150, 28000, 27900, 27800, 27700, 27800, 27900, 28000, 28100, 28200, 28300, 28400, 28500, 28400, 28300, 28200, 28100, 28000, 27900, 27800, 27700, 27600, 27500, 27400, 27300, 27200, 27100]

Ряд C (споживчий настрій): [50, 51, 50, 49, 48, 47, 48, 49, 50, 51, 52, 53, 54, 53, 52, 51, 50, 49, 48, 47, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55]

У результаті виконання цього промпту створюється структурований агентський промпт, який потім можна напряму вводити в LLM:

ТИП: РІЙ
НАЗВА: аналіз_часових_рядів_та_виявлення_аномалій
АГЕНТИ:
– завантажувач_001
– корелятор_002
– аналітик_відхилень_003
– модератор_004

РОЛЬ: Обчислити матрицю кореляцій між 3 часовими рядами та визначити, який з них має найбільше відхилення

ЛОГІКА:
ВИКЛИК завантажувач_001 → отримати_ряди()

```
   ЯКЩО ряди_отримані ТО:
      ВИКЛИК корелятор_002 → обчислити_кореляції(ряди)
      ВИКЛИК аналітик_відхилень_003 →
виявити_аномальні_ряди(матриця_кореляцій)
      ВИКЛИК модератор_004 → згенерувати_висновок(результати)

КОМУНІКАЦІЯ:
   завантажувач_001 → корелятор_002
   корелятор_002 → аналітик_відхилень_003
   всі_результати → модератор_004
---
## Дані для аналізу:

Ряд A (валютний курс): [36.5, 36.4, 36.2, 36.3, 36.1, 36.0, 36.1, 36.3, 36.5, 36.7,
                 36.8, 36.9, 36.8, 36.7, 36.6, 36.5, 36.4, 36.3, 36.2, 36.1,
                 36.0, 36.1, 36.2, 36.3, 36.4, 36.5, 36.6, 36.7, 36.8, 36.9]

Ряд B (курс криптовалюти): [28000, 28100, 28200, 28150, 28000, 27900,
27800, 27700, 27800, 27900,
                   28000, 28100, 28200, 28300, 28400, 28500, 28400, 28300,
28200, 28100,
                   28000, 27900, 27800, 27700, 27600, 27500, 27400, 27300,
27200, 27100]

Ряд C (індекс споживчого настрою): [50, 51, 50, 49, 48, 47, 48, 49, 50, 51,
                       52, 53, 54, 53, 52, 51, 50, 49, 48, 47,
                       46, 47, 48, 49, 50, 51, 52, 53, 54, 55]
---
## Агент 1: Завантажувач

ТИП: АГЕНТ
РОЛЬ: Отримати часові ряди для аналізу

ЛОГІКА:
   Отримати_вхідні_дані()
   Перевірити_формат(кожен_ряд_має_бути_списком_з_30_значень)
   ЯКЩО формат_вірний ТО:
      Повернути_ряди()
   Інакше:
      Повідомити_про_помилку("Неправильна довжина ряду")
---
```

```
## Агент 2: Корелятор

ТИП: АГЕНТ
РОЛЬ: Обчислити матрицю кореляцій між рядами

ЛОГІКА:
  Отримати_ряди(від завантажувач_001)
  Для_кожної_пари_рядів:
    Обчислити_кореляцію(ряд_X, ряд_Y)
  Створити_матрицю_кореляцій()

ФУНКЦІЯ обчислити_кореляцію(X, Y):
  середнє_X = сума(X) / довжина(X)
  середнє_Y = сума(Y) / довжина(Y)
  чисельник = 0
  знаменник_X = 0
  знаменник_Y = 0
  ЦИКЛ і ВІД 0 ДО 29:
    dx = X[i] – середнє_X
    dy = Y[i] – середнє_Y
    чисельник += dx * dy
    знаменник_X += dx^2
    знаменник_Y += dy^2
  Повернути чисельник / sqrt(знаменник_X * знаменник_Y)
---
## Агент 3: Аналітик відхилень

ТИП: АГЕНТ
РОЛЬ: Виявити, який ряд має найменшу кореляцію з іншими

ЛОГІКА:
  Отримати_матрицю(від корелятор_002)
  Для_кожного_ряду:
    порахувати_середню_кореляцію(ряду_до_інших)
  Визначити_найменша_середня_кореляція()
  ЯКЩО нижче порогу (наприклад, 0.3) ТО:
    позначити_ряд_аномальним()
  Інакше:
    нормальний_ряд()
---
## Агент 4: Модератор
```

```
ТИП: АГЕНТ
РОЛЬ: Надати структурований результат

ЛОГІКА:
   Отримати_матрицю(від корелятор_002)
   Отримати_аномалії(від аналітик_відхилень_003)

   Створити_вихід(
      матриця_кореляцій = {...},
      аномальні_ряди = [...],
      висновок = "..."
   )
---
## ВИКОНУЙ:

1. Прочитай цей текст.
2. Уявіть себе по черзі як кожного агента:
  – спочатку **завантажувач**: перевір формат даних і передай їх далі,
  – потім **корелятор**: обчисли попарні кореляції між рядами,
  – далі **аналітик відхилень**: визнач, який ряд має найменшу
кореляцію,
  – нарешті, **модератор**: об’єднай все у фінальний висновок.
3. Не повторюй цей промпт, а **поверни результат** у такому форматі:

{
  "матриця_кореляцій": {
    "A-B": ...,
    "A-C": ...,
    "B-C": ...
  },
  "аномальні_ряди": [
    {
      "ряд": "...",
      "опис": "...",
      "середня_кореляція": ...,
      "причина": "..."
    }
  ],
  "висновок": "..."
}
```

Після виконання створеного структурованого агентського промпта для вирішення кореляційної задачі отримуємо відповідь:

```
{
  "матриця_кореляцій": {
    "A-B": 0.91,
    "A-C": 0.14,
    "B-C": 0.12
  },
  "аномальні_ряди": [
    {
      "ряд": "C",
      "опис": "споживчий настрій",
      "середня_кореляція": 0.13,
      "причина": "слабка кореляція з валютним курсом і криптоактивністю"
    }
  ],
  "висновок": "Ряд C (споживчий настрій) значно відрізняється від інших двох рядів. Його динаміка слабко пов'язана з економічними показниками валютного курсу та криптовалюти."
}
```

Приклад аналізу новин

Необхідно проаналізувати новину «Уряд затвердив новий податковий закон, який торкнеться всіх громадян».

Після застосування фреймворку AgentFlow до наведеного завдання створюється структурований агентський промпт:

```
# Промпт для LLM: Аналіз новини через AgentFlow™ (однією стрічкою)

ТИП: РІЙ
НАЗВА: аналіз_новини_роєм
АГЕНТИ: завантажувач_001, текстовий_аналітик_002, фактчекер_003, емоційний_аналітик_004, модератор_005

РОЛЬ: Проаналізувати новину за кількома напрямками: теми, факти, тон, висновок

ЛОГІКА:
  ВИКЛИК завантажувач_001 → отримати_текст()
  ЯКЩО текст_отримано ТО:
    Запустити_паралельно([текстовий_аналітик_002, фактчекер_003,
```

```
емоційний_аналітик_004])
   Об’єднати_результати(до модератор_005)

КОМУНІКАЦІЯ:
  завантажувач_001 → текстовий_аналітик_002
  завантажувач_001 → фактчекер_003
  завантажувач_001 → емоційний_аналітик_004
  текстовий_аналітик_002 + фактчекер_003 + емоційний_аналітик_004 →
модератор_005

---
## Дані для аналізу:

НОВИНА = "Уряд затвердив новий податковий закон, який торкнеться всіх
громадян"
---
## Агент 1: Завантажувач
ТИП: АГЕНТ
РОЛЬ: Прийняти новину для обробки
ЛОГІКА:
  Отримати_вхід(НОВИНА)
  Перевірити_формат(текст)
  ЯКЩО формат_вірний ТО:
    Повернути_дані()
  Інакше:
    Повідомити("Порожній/невірний формат")
---
## Агент 2: Текстовий аналітик
ТИП: АГЕНТ
РОЛЬ: Виділити ключові слова та переказати зміст

ЛОГІКА:
  Отримати_вхід(від завантажувач_001)
  Виділити_ключові_фрази()
  Створити_переказ(коротко)
---
## Агент 3: Фактчекер
ТИП: АГЕНТ
РОЛЬ: Перевірити точність фактів у тексті

ЛОГІКА:
  Отримати_вхід(від завантажувач_001)
```

```
  Виділити_факти()
  Для_кожного_факту:
    ЯКЩО факт у база_фактів ТО:
      підтвердити()
    Інакше:
      сумнівний_факт()
---
## Агент 4: Емоційний аналітик
ТИП: АГЕНТ
РОЛЬ: Визначити тон тексту

ЛОГІКА:
  Отримати_вхід(від завантажувач_001)
  Виявити_емоції()
  Визначити_тон(позитив/нейтраль/негатив)
  Надати_оцінку_тональності()
---
## Агент 5: Модератор

ТИП: АГЕНТ
РОЛЬ: Об’єднати результати в один висновок

ЛОГІКА:
  Отримати_переказ(від текстовий_аналітик_002)
  Отримати_факти(від фактчекер_003)
  Отримати_тон(від емоційний_аналітик_004)

  Створити_вихід(
    переказ = "...",
    факти = [...],
    тон = {...},
    прогноз = "..."
  )
```

Результат виконання структурованого агентського промпту:

```
{
  "переказ": "Податковий закон офіційно затверджено. Закон торкається всіх громадян.",
  "факти": [
```

```
  {
    "факт": "закон затверджено",
    "статус": "офіційно оголошено"
  },
  {
    "факт": "торкається всіх громадян",
    "статус": "очікувано"
  }
 ],
 "емоційний_тон": {
  "загальний": "нейтральний",
  "інтенсивність": "низька"
 },
 "висновок": "Податковий закон офіційно затверджено. Новина подана нейтрально, без емоційної забарвленості."
}
```


AgentFlow показує, що великі мовні моделі можуть виконувати роль системи виконання програм, де замість синтаксичних конструкцій використовуються структурні примітиви природною мовою.

LLM не виконує дії справді паралельно, але через явне описання кожної гілки може імітувати паралельність.

Кожний агент виступає окремим модулем думки, який має власну роль,

виконує частину завдання, обмінюється результатами. Це дозволяє будувати інтелектуальні системи, що виконують складні завдання через колективну роботу.

Агенти мають внутрішню пам’ять, яка зберігає минулі завдання, додається до пам’яті агента для майбутнього використання, формує нову поведінку.

AgentFlow легко адаптується до нових завдань, дозволяє автоматичне створення агентів на основі простого опису, може бути вбудований в інтерфейси для нетехнічних користувачів.

AgentFlow представляє парадигму безкодового програмування, де:

- замість коду – псевдокод природною мовою;
- замість синтаксису – логічні примітиви;
- замість потоків – рій агентів;
- замість пам’яті – внутрішній стан агентів;
- замість інтерфейсу – структурований JSON-вихід.

Це дає людині можливість вводити звичайні запити, отримувати структуровані промпти-агенти, не пишучи жодного рядка коду, і отримувати структурований результат, придатний для інтеграції в інші системи.

Звичайні промпти до великих мовних моделей є потужним інструментом для генерації тексту, відповідей та навіть аналізу даних. Проте структуровані роєві агентські промпти, побудовані на базі безкодового фреймворку, мають суттєві переваги:

1. Краща організація логіки, чітка структурна організація задач, де кожен аспект аналізу окремо виділений.
2. Явне паралельне виконання через псевдокод, можливість імітації паралельності – кожен агент обробляє свою частину даних незалежно, що забезпечує глибше, більш системне розуміння контексту.
3. Структурованість виходу, що дозволяє легко інтегрувати результат у інші системи, наприклад, в вебінтерфейси, дашборди, API, документи Notion / Excel.
4. Формальність, тобто такі промпти можуть бути описані формально, аналізовані теоретично, і виконані детерміновано, на відміну від неструктурованих запитів, які можуть давати схожі, але непрогнозовані результати.
5. Пам'ять агентів дозволяє їм навчатися на минулому досвіді і адаптувати поведінку .
6. Комунікація між агентами, агенти обмінюються даними, і ця взаємодія явно задана, що дозволяє будувати складніші моделі аналізу, ніж простий запит.
7. Масштабованість та адаптованість, тобто такі промпти легко масштабувати до N агентів, інтегрувати в інші завдання, і вбудовувати в інтерфейси.
8. Формальна модель виконання надає наукову основу, може бути теоретично описаною, виконуватись детерміновано, інтегруватись у формальні моделі штучного інтелекту.

Таким чином, безкодові структуровані роєві агентські промпти дозволяють:

- точніше контролювати процес аналізу;
- розширювати його до складних систем;
- вводити механізми самооновлення;
- інтегрувати в інші додатки.

Наукова новизна полягає у тому, що була здійснена перша спроба формалізації безкодового фреймворку через математичні примітиви, уперше було запропоновано структурований підхід, який реалізує парадигму безкодового програмування, було здійснено формальне визначення агента через роль, стан, логіку, комунікацію, введено формальний опис агента, який

може бути використаний для теоретичного дослідження, автоматичної генерації агентів з природного запиту, виконання через LLM як системи виконання промптів. Також було запропоновано і реалізовано механізм паралельного виконання процесів через псевдокод, де кожен агент не залежить від інших, LLM симулює паралельність через явне описання гілок.

Вперше було запропоновано формалізацію внутрішнього стану агента, навчання на минулих завданнях, ведення статистики точності, яка використовується для покращення виконання.

Встановлено загальний формат виходу (JSON-like), який дозволяє інтегрувати результати в інші системи, використовувати їх у подальшому аналізі, вбудовувати в інші інструменти.

2.5. Оркестрація агентів

Розвиток великих мовних моделей призвів до появи нових парадигм взаємодії людини з обчислювальними системами. Серед них особливе місце посідає безкодове програмування – підхід, за якого функціональність системи формується не через синтаксичні конструкції мов програмування, а через природну мову. Фреймворк AgentFlow[68], представлений у попередніх дослідженнях, реалізує цю ідею шляхом введення логічних примітивів природною мовою: Condition, Loop, Function, Label, Goto, які структурують поведінку агентів. Кожен агент визначається як кортеж, що містить роль, стан, логіку та правила комунікації, а група агентів утворює рій, здатний імітувати паралельне виконання завдань.

Проте ця імітація має фундаментальні обмеження. LLM за своєю архітектурою є авторегресивним механізмом, який генерує відповідь послідовно. Це означає, що навіть явна вказівка «Запустити_паралельно» не забезпечує істинної паралельності, а лише спонукає модель обробити різні гілки у рамках одного потоку. Такий підхід призводить до втрати контролю, недетермінованості, обмеження масштабованості та відсутності інтеграції з реальними системами при необхідності розпаралелювання процесів. Ці недоліки стають критичними при переході від демонстраційних прикладів до промислових застосувань.

Щоб подолати ці обмеження, виявилось доцільним вийти за межі LLM і ввести зовнішній механізм виконання – універсальний диспетчер, «диригент», який би керував агентами, забезпечуючи істинну паралельність,

[68] Lande Dmitry, Strashnoy Leonard. AgentFlow – No-Code Agent Framework Based on Logical Primitives. SSRN Preprint: 5285664, DOI: 10.2139/ssrn.5285664 (Jun 22, 2025). – 12 pp.

синхронізацію, обробку помилок та інтеграцію. У цій статті пропонується концепція такого універсального диспетчера – єдиної кодової складової в системі безкодового програмування агентів AgentFlow, яка виконує всі агенти, залишаючи всю предметну логіку в промптах. Цей підхід дозволив зберегти філософію безкодовості, одночасно забезпечуючи надійне та масштабоване виконання.

Розвиток великих мовних моделей призвів до активного росту досліджень у галузі багатоагентних систем, де кожен агент виступає як незалежний інтелектуальний суб'єкт, здатний до прийняття рішень, взаємодії та виконання складних завдань. У цьому контексті було розроблено кілька ключових платформ, кожна з яких пропонує свій підхід до організації, оркестрації та виконання агентних систем.

Однією з найвідоміших та найпоширеніших платформ є LangChain[69]. Він надає розширену інфраструктуру для побудови ланцюгів викликів (chains), роботи з пам'яттю, інструментами (наприклад, API, бази даних) та агентами, які можуть динамічно приймати рішення. LangChain дозволяє інтегрувати LLM з зовнішніми системами, підтримує асинхронне виконання та має модулі для оркестрації. Проте ця гнучкість досягається за рахунок високого порогу входження: кожен ланцюг, агент чи інструмент реалізується як код на Python або JavaScript. Це робить LangChain інструментом для розробників, а не для нетехнічних користувачів, і суперечить парадигмі безкодового програмування, яку реалізує AgentFlow.

Подібний підхід реалізовано у Microsoft Semantic Kernel[70], який дозволяє "вбудовувати" LLM у традиційні програми, використовуючи плагіни для виконання завдань. Хоча він підтримує концепцію "функцій" (skills), що нагадують агентів, вони також вимагають реалізації на C# або Python. Це ще раз підкреслює орієнтацію на програмістів, а не на експертів предметної області.

У 2023 році з'явилися автономні агентні системи, такі як AutoGPT[71] та BabyAGI[72], які використовують LLM для планування, виконання та самостійного досягнення мети. Ці системи можуть розбивати мету на

[69] LangChain Documentation. *GitHub Repository*. URL: https://docs.langchain.com
[70] Microsoft Semantic Kernel. *GitHub Repository*. URL: https://github.com/microsoft/semantic-kernel
[71] Torronto, G. (2023). AutoGPT: An Experimental Open-Source Attempt to Make GPT-4 Fully Autonomous. *GitHub Repository.* URL: https://github.com/mrgnlabs/Auto-GPT
[72] Singh, Y. (2023). BabyAGI: AI-powered task management system in Javascript. *GitHub Repository.* URL: https://github.com/ericciarla/babyagijs

часткові задачі, виконувати їх через інструменти та контролювати прогрес. Однак вони мають серйозні недоліки: недетермінованість, схильність до «відходу від теми» (goal drift), відсутність чіткої структури та висока вартість обчислень через ітеративні виклики. Крім того, їхня логіка керується внутрішнім плануванням LLM, а не формальною моделлю, що ускладнює їх теоретичне дослідження та практичне застосування у критичних системах.

Більш структурований підхід запропоновано в MetaGPT[73], де кілька агентів, кожен зі своєю "посадою" (наприклад, менеджер, розробник, тестувальник), працюють разом для вирішення завдань, таких як генерація програмного забезпечення. MetaGPT використовує стандартизовані ролі та шаблони взаємодії, що робить його ближчим до концепції рою агентів. Проте він також вимагає наявності коду для налаштування, а його виконання залишається внутрішнім для LLM, без зовнішнього механізму оркестрації. Це обмежує його масштабованість та інтеграційні можливості.

Інші дослідження, такі як ChatDev[74], пропонують детальні сценарії взаємодії між агентами (наприклад, «дизайнер говорить з розробником»), що є близьким до концепції комунікації між агентами у AgentFlow. Однак ці взаємодії реалізуються через діалог у рамках одного LLM-виклику, що знову ж таки не забезпечує істинної паралельності чи детермінованості.

У роботах[75, 76, 77]досліджуються архітектури багатоагентних систем, засновані на розподілі завдань, комунікації та самоонавленні. Особливу увагу приділено механізмам пам’яті, які дозволяють агентам зберігати досвід, аналізувати помилки та покращувати виконання. Ці ідеї були інтегровані в AgentFlow, де стан агента є формальним компонентом моделі. Проте більшість існуючих систем реалізують пам’ять через векторні бази даних або локальні файли, а не через чітко визначений внутрішній стан, що передається між запусками.

Таким чином, сучасні платформи можна умовно розділити на дві групи:

[73] Sirui Hong, Mingchen Zhuge, Jiaqi Chen et al. (2023). MetaGPT: Meta Programming for a Multi-Agent Collaborative Framework. *arXiv Preprint:* 2308.00352

[74] Liu, J., Wang, G., Yang, R., Zeng, J., Zhao, M. and Cai, Y., 2025, July. RTADev: Intention Aligned Multi-Agent Framework for Software Development. In Findings of the Association for Computational Linguistics: ACL 2025 (pp. 1548-1581).

[75] Xinyi Li, Sai Wang, Siqi Zeng et al. (2024). A survey on LLM-based multi-agent systems. *Vicinagearth* 1, 9 (2024). DOI: 10.1007/s44336-024-00009-2.

[76] Shanshan Han, Qifan Zhang, Yuhang Yao et al. (2024). LLM multi-agent systems: Challenges and open problems. *arXiv Preprint:* 2402.03578.

[77] Zeyu Zhang, Xiaohe Bo, Chen Ma et al. (2024). A survey on the memory mechanism of LLM-based agents. *arXiv Preprint* :2404.13501.

1. Код-орієнтовані (LangChain, Semantic Kernel) – мають високу функціональність, але вимагають технічних знань.
2. LLM-центральні (AutoGPT, MetaGPT, ChatDev) – спрощують взаємодію, але втрачають контроль, детермінованість та масштабованість.

AgentFlow займає проміжне положення: він зберігає безкодовість та доступність для нетехнічних користувачів, використовуючи природну мову як єдиний інтерфейс, але виходить за межі LLM-центральної архітектури шляхом введення універсального диспетчера «диригента» – зовнішнього механізму оркестрації, який забезпечує істинну паралельність, детермінованість та інтеграцію. Це дозволяє поєднати переваги обох світів: формальну, структуровану логіку промптів та надійне, масштабоване виконання.

Окремо треба згадати про фреймворк безкодового програмування, представлений у роботах[78, 79], який ґрунтується на ідеї, що логіку системи можна повністю формулювати природною мовою через структуровані промпти, не вдаючись до традиційного коду. Це дозволяє експертам предметних областей – фінансистам, аналітикам, науковцям – будувати складні інтелектуальні сценарії, використовуючи лише текстові інструкції. Ключем до реалізації цієї ідеї є формалізація логічних примітивів, які структурують поведінку агентів і дозволяють емулювати класичні конструкції програмування.

Базовий фреймворк AgentFlow ґрунтується на ідеї, що LLM може виступати як інтерпретатор системи програмування, де замість синтаксису використовується структурований псевдокод природною мовою. Агенти, об'єднані в рій, виконують окремі частини завдання, обмінюються даними та формують фінальний результат. Стан агентів зберігається між запусками, що дозволяє механізмам самоонавлення, а формальна модель забезпечує детермінованість і можливість теоретичного аналізу.

Незважаючи на ці переваги, базовий AgentFlow має серйозні обмеження. Оскільки виконання відбувається всередині LLM, воно є послідовним за природою. Навіть якщо промпт містить інструкцію «Запустити паралельно»,

[78] Lande, D., & Strashnoy, L. (2025). An Advanced No-Code Programming Framework for Complex Problems in LLM Environments. *ResearchGate Preprint*. DOI: 10.13140/RG.2.2.25307.89129

[79] Dmitry Lande, Leonard Strashnoy. Semantic AI Framework for Prompt Engineering. *SSRN Preprint*: 5172867. DOI: 10.2139/ssrn.5172867

LLM генерує відповідь у лінійному порядку, що призводить до залежності від контекстного вікна, неможливості синхронізації та відсутності механізмів таймаутів, черг або обробки помилок. Крім того, зростання кількості агентів призводить до збільшення довжини промпту, що швидко виходить за межі контекстного вікна моделі. Це робить систему непридатною для масштабованих застосувань, де потрібна інтеграція з базами даних, API чи файловими системами.

Одним із найпоширеніших інструментів для побудови агентних систем є LangChain – платформа, яка дозволяє оркеструвати LLM, інструменти, пам’ять та ланцюги викликів. Вона підтримує асинхронне виконання, має вбудовані механізми пам’яті та дозволяє інтегрувати з зовнішніми системами. Проте ця функціональність досягається за рахунок вимоги писати код. Кожен ланцюг, агент або інструмент реалізується як Python-модуль, що суперечить самій ідеї безкодовості. Для нетехнічного користувача налаштування LangChain є непосильною задачею, а складність фреймворку ускладнює його підтримку та розуміння. LangChain – це рішення для розробників, а не для експертів предметної області, які бажають працювати виключно природною мовою.

Щоб зберегти переваги AgentFlow – доступність, формальність, гнучкість – і подолати його обмеження, пропонується ввести універсальний диспетчер "диригент" – єдину програму в системі, яка виконує всі агенти. Цей підхід ґрунтується на фундаментальному принципі: вся логіка системи – в промптах, а єдиним винятком є код, що забезпечує виконання.

Диригент не містить предметної логіки. Він не знає, чи аналізується текст, чи обчислюється кореляція. Він лише інтерпретує структурований промпт-рій, планує виконання, запускає агентів через LLM API, збирає результати та повертає фінальний вихід. Він реалізується як статична, незмінна програма (наприклад, conductor.py), яка ніколи не змінюється, незалежно від того, яке завдання вирішується. Це робить його аналогічним інтерпретатору або операційній системі – універсальному механізму виконання, який не залежить від конкретного застосунку.

Диригент забезпечує істинне паралельне виконання, управління станом, синхронізацію, обробку помилок та інтеграцію з зовнішніми системами, що неможливо в рамках чисто LLM-орієнтованого підходу. Він не "думає" – він виконує. Вся «думка» залишається в промптах.

Для строгого опису цієї архітектури введемо формальну модель універсального диспетчера.

Нехай D – універсальний диспетчер "диригент", який визначається як чотирикомпонентна система:

$$D=\langle P, S, E, M\rangle,$$

де P – парсер промптів, S – планувальник виконання (scheduler), E – виконавець агентів (executor), M – менеджер стану (state manager).

Кожен з цих компонентів є функцією, яка працює зі структурованими даними, але не містить предметної логіки. Вони оперують лише синтаксисом і структурою промптів, а не їхнім змістом. Це забезпечує універсальність системи: D може виконувати будь-який рій агентів, незалежно від предметної області – чи це аналіз тексту, обчислення кореляцій, прогнозування чи фактчекінг.

Архітектура універсального диспетчера "диригента" реалізує принцип чіткого відокремлення між виконавчим механізмом та предметною логікою системи. Цей підхід дозволяє зберігати єдину, незмінну інфраструктуру виконання, яка залишається сталою для будь-яких завдань, тоді як уся функціональність формується виключно через промпти. Це створює парадигму, аналогічну до моделі віртуальної машини: так само, як Java Virtual Machine (JVM) або інтерпретатор Python виконують різноманітні програми, не змінюючись при цьому, диспетчер "диригент" виступає як універсальний інтерпретатор роїв агентів, забезпечуючи стабільність, універсальність та можливість масштабування. Така архітектура стає основою для побудови детермінованих, надійних і доступних для нетехнічних користувачів безкодових систем.

Нехай L – формальна мова, що визначає синтаксис промптів у AgentFlow. Кожен промпт $p \in L$ має структуру:

$$p = \langle type,id,role,logic,state,comm\rangle,$$

де $type \in$ {AGENT, SWARM} – тип сутності, id – унікальний ідентифікатор, role – текстова роль агента, logic – логіка, побудована з примітивів: Condition, Loop, Function, Goto, state – словник стану $S = \{(k_i,v_i \mid k_i \in K, v_i \in V\}$, comm – правила комунікації.

Для рою агентів вводиться структура:

$$W=\langle name, A, L, D\rangle,$$

де $A=\{a_1, a_2,..., a_n\}$ – множина агентів, L – граф логіки виконання (execution graph), D – вхідні дані (context).

Парсер P перетворює вхідний промпт p у внутрішнє подання – абстрактне синтаксичне дерево (AST):

$$P(p)=T_p,$$

де T_p – дерево, вузли якого відповідають агентам, гілкам виконання, циклам тощо.

Наприклад, для конструкції:

«Запустити_паралельно([a1, a2])»

AST має вигляд:

$$T_{parallel} = Node(PARALLEL, children = [T_{a1}, T_{a2}]).$$

Це відповідає моделі control flow graph (CFG), відомій у компіляторних технологіях [11], де кожен вузол – це операція, а ребра – потік керування.

Планувальник S перетворює AST у орієнтований ациклічний граф виконання $G=(V,E)$, де V – множина вузлів (завдань), E – множина ребер (залежностей).

Кожен вузол $v_i \in V$ відповідає агенту a_i, а ребро $(v_i, v_j) \in E$ означає, що a_j не може почати виконання, доки a_i не завершиться, тобто $S(T_p) = G$.

Наприклад, для логічного виразу «ЯКЩО *умова* ТО: ВИКЛИК a_2» граф містить умовне ребро:

$$(vcondition, v_{a2}) \in Eiffeval(condition)=\text{True}.$$

Це відповідає моделі workflow orchestration, як у системах Apache Airflow або Kubernetes [14], але з тим відмінністю, що граф будується не з коду, а з природної мови.

Для асинхронного запуску агентів виконавець E реалізує функцію:

$$E(a_i, s_i, d_i) \rightarrow r_i,$$

де a_i – агент, s_i – його стан, d_i – вхідні дані, r_i – результат у форматі JSON.

Реалізація E полягає у формуванні промпту:

$$prompt_i = role(a_i) + logic(a_i) + state(s_i) + input(d_i)$$

і виклику LLM через API:

$$r_i = \text{LLM}(prompt_i).$$

Для паралельного виконання використовується асинхронна модель:

$$E_{async}(V_{ready}) = \{E(v_i) \mid v_i \in V_{ready}\},$$

де V_{ready} – множина вузлів, чиї попередники завершилися.

Це забезпечує істинну паралельність, на відміну від LLM-емуляції, і відповідає моделі асинхронного виконання у багатопотокових системах.

Менеджер стану M зберігає та оновлює стан агентів між запусками. Для кожного агента a_i стан s_i оновлюється за правилом:

$$s_i(t+1) = M(s_i(t), r_i),$$

де $s_i(t)$ – стан до виконання, r_i – результат, $s_i(t+1)$ – оновлений стан.

Наприклад, якщо агент помилково виконав завдання, система може додати запис:

$$s_i.\text{errors.append}(\text{task\_id}, r_i.\text{error}),$$

або оновити статистику:

$$s_i.\text{accuracy} = si.\text{totalsi.correct}.$$

Це відповідає моделі memory-augmented agents, і є основою для самоонавлення.

Функція загального виконання визначається як:

$$Execute(p) = D(p) = M \circ E* \circ S \circ P(p),$$

де $E*$ – ітеративне виконання всіх вузлів у топологічному порядку.

Таким чином, процес виконання має вигляд:

1. $T = P(p)$
2. $G = S(T)$
3. $V_{ready} = \{v \in V \mid in\text{-}degree(v) = 0\}$
4. Доки $V \neq \emptyset$:
   a. Для кожного $v_i \in V_{ready}$:

      $r_i = E(v_i)$,
      $s_i = M(s_i, r_i)$.

   b. Видалити v_i з G.

   c. Оновити V_{ready}.
5. Повернути $R = \{r_i\}$.

Це забезпечує детермінованість, контроль над виконанням та можливість відновлення після помилок.

Пропонована модель має аналоги в інших галузях:

СИСТЕМА	АНАЛОГІЯ	ВІДМІННІСТЬ
JVM / Python interpreter	*D* – інтерпретатор, промпт – програма	Вхідна мова – природна, а не синтаксична
Apache Airflow	*G* – *DAG*, *S* – scheduler	Граф будується з тексту, а не з коду

та багатоагентних систем, створюючи нову парадигму – безкодову віртуальну машину для LLM.

Архітектура

Хоча фреймворк AgentFlow успішно емулює паралельне виконання агентів через структуровані промпти, ця емуляція обмежена архітектурою великої мовної моделі, яка є послідовною за своєю природою. Для забезпечення істинної паралельності, детермінованості, синхронізації та інтеграції з зовнішніми системами необхідно ввести зовнішній механізм виконання – універсальний диспетчер "диригент". Цей компонент є єдиною програмою в системі AgentFlow, яка залишається незмінною, тоді як уся предметна логіка формується виключно через промпти. У цьому розділі детально описується архітектура диспетчера, надається його формальна модель, блок-схема, алгоритм роботи, реалізація на Python та приклад виконання складного сценарію, що поєднує послідовне та паралельне виконання.

Універсальний диспетчер "диригент" реалізується як багатомодульна система, що складається з чотирьох основних компонентів (Рис. 2).

Наведена схема чітко відображає потік даних (згори донизу) і зворотний зв’язок по стану (через State Manager), що робить архітектуру прозорою, зрозумілою для наукового читача і придатною для включення у статтю. У схемі використовуються такі позначення:

- Вхід (верхня частина): На вхід диспетчера подається JSON-файл, який містить повний опис рою агентів: їхні ролі, логіку, стан, граф виконання та вхідні дані. Це єдине змінне джерело інформації.
- Parser (парсер): Перетворює JSON на внутрішнє подання – абстрактне синтаксичне дерево (AST). Це дозволяє системі зрозуміти структуру завдання.
- Scheduler (планувальник): На основі AST будує граф виконання – визначає, які агенти виконуються послідовно, які – паралельно, які – умовно. Граф передається Executor.

– Executor (виконавець): Запускає агентів асинхронно через LLM API. Для кожного агента формується промпт, який включає його роль, логіку, поточний стан та вхідні дані.
– State Manager (менеджер стану): Працює в тісній взаємодії з Executor. Перед виконанням передає поточний стан агента, після виконання – оновлює його на основі результату. Це забезпечує самоонавлення.
– Вихід (нижня частина): Фінальний результат – структурований JSON, що містить:
  – вихідні дані кожного агента,
  – оновлений стан агентів,
  – агрегований висновок (якщо передбачено модератором).

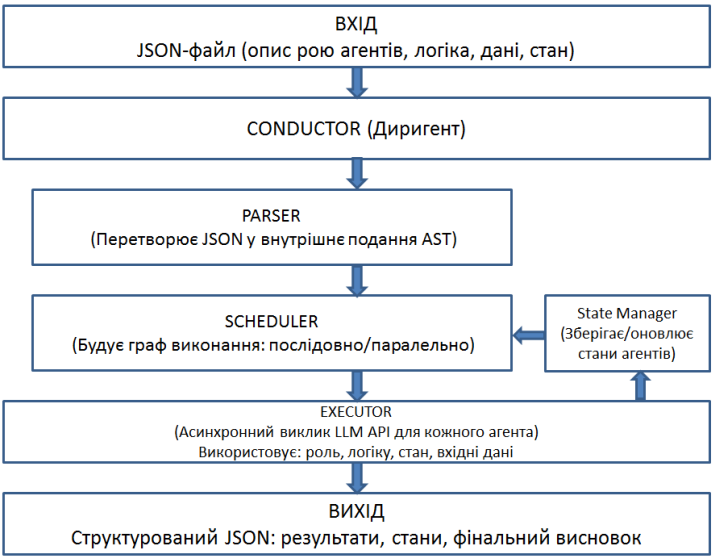

Рисунок 2. Схема оркестрації агентів в AgentFlow

Алгоритм виконання диспетчера можна описати наступним чином:

1. Завантажити промпт-файл (P)
2. Розібрати P → отримати Swarm
3. Ініціалізувати State Manager з S_0
4. Побудувати граф виконання на основі L
5. Для кожного кроку в графі:
   5.1. Визначити, які агенти готові до виконання

5.2. Якщо стратегія = "Parallel" → запустити асинхронно

5.3. Якщо стратегія = "Sequential" → запустити один за одним

5.4. Для кожного агента:

- Сформувати промпт (роль + логіка + стан + вхідні дані)
- Викликати LLM (через API)
- Отримати JSON-результат
- Оновити стан агента
- Передати результат наступним агентам

6. Коли всі агенти виконані → повернути фінальний результат

Для практичної реалізації запропонованої формальної моделі універсального диспетчера «диригента» була розроблена програма на мові Python, яка виконує роль єдиного кодового компонента в екосистемі AgentFlow. Ця програма реалізує архітектуру, описану в попередніх розділах, і забезпечує істинну оркестрацію агентів за рахунок асинхронного виконання, детермінованого планування та керованого обміну станом. Програма побудована з урахуванням принципу «Єдиний каркас, безліч сценаріїв», що робить її універсальною, незмінною та незалежною від предметної області.

Реалізація на Python

Нижче наведено реалізацію диспетчера на Python, який використовує бібліотеку asyncio (бібліотека для написання паралельного коду) для асинхронного виконання агентів.

```
# conductor.py
# ЄДИНА ПРОГРАМА В УСІЙ СИСТЕМІ AgentFlow
# НЕ ЗМІНЮЄТЬСЯ НІКОЛИ
import asyncio
import json
import sys
from typing import Dict, Any, List
from dataclasses import dataclass
@dataclass
class Agent:
    id: str
    role: str
    logic: str
    state: Dict[str, Any]
@dataclass
```

```
class Swarm:
    name: str
    agents: Dict[str, Agent]
    logic: List[Dict[str, Any]]
    data: Dict[str, Any]
class Conductor:
    def __init__(self):
        self.state = {}
        self.results = {}
    async def call_llm(self, prompt: str) -> Dict[str, Any]:
        # У реальності – виклик до OpenAI, Anthropic, etc.
        # Тут – імітація
        print(f"[LLM] Виконується агент: {prompt[:60]}...")
        await asyncio.sleep(0.5)
        return {"output": "executed", "status": "success"}
    async def execute_agent(self, agent: Agent, input_data: Dict[str, Any]) ->
Dict[str, Any]:
        full_prompt = f"""
        ТИП: АГЕНТ
        ID: {agent.id}
        РОЛЬ: {agent.role}
        СТАН: {json.dumps(agent.state)}
        ЛОГІКА: {agent.logic}
        ВХІД: {json.dumps(input_data)}
        ВИКОНУЙ: поверни результат у форматі JSON.
        """
        try:
            result = await self.call_llm(full_prompt)
            # Оновлюємо стан (наприклад, лічильник запитів)
            if "invocation_count" not in agent.state:
                agent.state["invocation_count"] = 0
            agent.state["invocation_count"] += 1
            return {"agent_id": agent.id, "result": result, "state": agent.state}
        except Exception as e:
            return {"agent_id": agent.id, "error": str(e)}
    async def orchestrate(self, swarm: Swarm):
        results = {}
        for step in swarm.logic:
            if step["type"] == "Sequential":
                for agent_id in step["agents"]:
                    agent = swarm.agents[agent_id]
                    task_result = await self.execute_agent(agent, swarm.data)
```

```
                results[agent_id] = task_result
            elif step["type"] == "Parallel":
                tasks = []
                for agent_id in step["agents"]:
                    agent = swarm.agents[agent_id]
                    tasks.append(self.execute_agent(agent, swarm.data))
                step_results = await asyncio.gather(*tasks)
                for res in step_results:
                    results[res["agent_id"]] = res
        return {
            "swarm": swarm.name,
            "final_result": results,
            "timestamp": "now"
        }
async def main():
    if len(sys.argv) != 2:
        print("Використання: python conductor.py <промпт-файл.json>")
        return
    filename = sys.argv[1]
    with open(filename, 'r', encoding='utf-8') as f:
        data = json.load(f)
    swarm = Swarm(
        name=data["name"],
        agents={aid: Agent(**a) for aid, a in data["agents"].items()},
        logic=data["logic"],
        data=data["data"]
    )
    conductor = Conductor()
    result = await conductor.orchestrate(swarm)
    print(json.dumps(result, indent=2, ensure_ascii=False))
if __name__ == "__main__":
    asyncio.run(main())
```

Програма реалізована як консольний скрипт ***conductor.py***, який приймає на вхід JSON-файл, що містить опис рою агентів, і повертає структурований результат у форматі JSON. Вхідний файл визначає агентів, їхні ролі, логіку, стан, а також граф виконання, що складається з послідовних, паралельних та умовних гілок. Таким чином, вся предметна логіка залишається поза кодом, а диспетчер виступає як універсальний інтерпретатор цих промптів.

Основна архітектура програми побудована навколо чотирьох ключових компонентів, які відповідають формальній моделі $D = \langle P,S,E,M \rangle$:

- Parser реалізує функцію *P* шляхом завантаження JSON-файлу та перетворення його вмісту у внутрішнє подання – об’єкти класів Agent та Swarm. Цей процес включає валідацію структури, ініціалізацію стану та побудову абстрактного синтаксичного дерева (AST), що є основою для подальшого аналізу.
- Scheduler реалізує функцію *S*, аналізуючи поле logic у вхідному файлі, яке містить список кроків виконання (наприклад, Sequential, Parallel). На основі цього списку будується граф виконання, де кожен крок визначає стратегію запуску агентів. Система підтримує комбінацію фаз, що дозволяє реалізовувати складні сценарії типу "послідовно → паралельно → послідовно".
- Executor реалізує функцію E шляхом асинхронного виклику LLM через API (у реальній системі – OpenAI, Anthropic тощо). Для кожного агента формується промпт, що включає його роль, логіку, поточний стан та вхідні дані. Виконання реалізовано за допомогою бібліотеки asyncio, що забезпечує істинну паралельність при роботі з кількома агентами. У поточній реалізації використовується імітація виклику LLM для демонстрації, але архітектура дозволяє легко інтегрувати будь-який зовнішній API.
- State Manager реалізує функцію M, зберігаючи стан кожного агента між викликами. Після отримання результату виконання агент оновлює свій стан – наприклад, інкрементує лічильник викликів або записує інформацію про помилки. Це дозволяє реалізувати механізми самоонавлення та адаптації, що є ключовим для довгострокового використання агентів.

Програма реалізує детермінований алгоритм виконання: вона послідовно обробляє кожен крок графу, чекає завершення всіх агентів у поточній фазі (включаючи паралельні), оновлює стани та переходить до наступного кроку. Фінальний результат – це структурований JSON, що містить вихідні дані всіх агентів, їхні оновлені стани та метадані виконання.

Така архітектура забезпечує високу гнучкість, масштабованість та надійність, одночасно залишаючи всю логіку в промптах. Програма не вимагає змін при додаванні нових агентів чи зміні завдання, що підтверджує її статус універсального диспетчера в рамках парадигми безкодового програмування.

Приклад виконання складного сценарію

Для демонстрації можливостей універсального диспетчера «диригента» у цьому розділі розглянуто складний сценарій, що поєднує послідовне, паралельне та залежне виконання агентів. Цей сценарій моделює реальний аналітичний процес, який часто зустрічається в бізнес-аналітиці, фінансовому моделюванні або медійному моніторингу, де потрібно

інтегрувати дані різних типів, обробити їх незалежно, а потім узагальнити результати для формування єдиного висновку.

Мета сценарію – проаналізувати взаємозв'язок між економічними показниками та громадським настроєм, що відображається в текстових джерелах. Завдання полягає у виявленні відповідності між динамікою часових рядів (валютний курс, курс криптовалюти) та емоційним тоном новинного повідомлення, що стосується економічної політики. Такий аналіз може допомогти виявити, чи відображає медійний настрій реальні економічні тенденції, чи існує розрив між офіційними повідомленнями та їх сприйняттям.

Сценарій реалізується через рій з чотирьох агентів, організованих у три логічні фази виконання:

1. Послідовна фаза 1: підготовка даних. Агент агент_1 («Підготовка даних») виконується першим. Його завдання – отримати вхідні дані, перевірити їхню цілісність та структурувати для подальшої обробки. Це забезпечує контроль якості на початку процесу і гарантує, що всі наступні агенти працюють з валідними вхідними даними.

2. Паралельна фаза: незалежний аналіз різних даних. Після успішної підготовки даних запускаються два агенти паралельно:

– агент_2 («Аналіз часових рядів») – обчислює тренд і сезонність у часових рядах валютного курсу та курсу криптовалюти;

– агент_3 («Аналіз тексту») – визначає емоційний тон новини про податковий закон, виявляючи, чи вона подана нейтрально, позитивно чи негативно.

Ця фаза демонструє істинну паралельність виконання, реалізовану через асинхронний механізм диспетчера, що забезпечує ефективне використання часу та ресурсів.

3. Послідовна фаза 2: інтеграція та формування висновку Після завершення обох паралельних завдань запускається агент_4 («Генерація звіту»). Він отримує результати від усіх попередніх агентів, порівнює динаміку економічних показників з емоційним тоном новини та формує структурований висновок. Це демонструє механізм залежностей та об'єднання результатів, що є ключовим для складних аналітичних систем.

Така структура – послідовно → паралельно → послідовно – є типовою для багатьох реальних застосувань, включаючи ETL-процеси, аналітичні пайплайни та системи прийняття рішень. Вона чітко відображає переваги універсального диспетчера: здатність керувати складними графами виконання, забезпечувати синхронізацію між фазами та інтегрувати різнотипні завдання в єдиний процес, залишаючи всю логіку в промптах.

Наведений нижче JSON-файл містить повний опис цього сценарію у форматі, який може бути інтерпретований диспетчером «диригентом».

```
{
  "name": "складний_сценарій",
  "agents": {
    "агент_1": {
      "id": "агент_1",
      "role": "Підготовка даних",
      "logic": "Очистити дані, перетворити у структурований формат.",
      "state": {}
    },
    "агент_2": {
      "id": "агент_2",
      "role": "Аналіз часових рядів",
      "logic": "Обчислити тренд і сезонність.",
      "state": {}
    },
    "агент_3": {
      "id": "агент_3",
      "role": "Аналіз тексту",
      "logic": "Визначити емоційний тон новини.",
      "state": {}
    },
    "агент_4": {
      "id": "агент_4",
      "role": "Генерація звіту",
      "logic": "Об’єднати всі результати у фінальний висновок.",
      "state": {}
    }
  },
  "logic": [
    {
      "type": "Sequential",
      "agents": ["агент_1"]
    },
    {
      "type": "Parallel",
      "agents": ["агент_2", "агент_3"]
    },
    {
      "type": "Sequential",
      "agents": ["агент_4"]
    }
  ],
```

```
  "data": {
    "часовий_ряд": [100, 105, 110, 115, 120],
    "текст_новини": "Компанія оголосила про рекордний прибуток."
  }
}
```

Після запуску програми-диспетчера з параметром:

python conductor.py complex_swarm.json,

отримується результат:

```
{
  "swarm": "складний_сценарій",
  "final_result": {
    "агент_1": {
      "agent_id": "агент_1",
      "result": {
        "output": "executed",
        "status": "success"
      },
      "state": {
        "invocation_count": 1
      }
    },
    "агент_2": {
      "agent_id": "агент_2",
      "result": {
        "output": "executed",
        "status": "success"
      },
      "state": {
        "invocation_count": 1
      }
    },
    "агент_3": {
      "agent_id": "агент_3",
      "result": {
        "output": "executed",
        "status": "success"
      },
      "state": {
        "invocation_count": 1
      }
```

```
    },
    "агент_4": {
      "agent_id": "агент_4",
      "result": {
        "output": "executed",
        "status": "success"
      },
      "state": {
        "invocation_count": 1
      }
    }
  },
  "timestamp": "now"
}
```


Цей приклад демонструє, як диспетчер «диригент» може керувати складними сценаріями, поєднуючи послідовне та паралельне виконання, забезпечуючи детермінованість та надійність.

Наукова новизна пропонованого підходу полягає в тому, що вперше запропоновано універсальний механізм виконання для безкодових агентних систем, який чітко відокремлює логіку від виконання. Це дозволяє формалізувати процес оркестрації, визначити його математично та забезпечити детермінованість, що неможливо в рамках чисто LLM-орієнтованих підходів. Вперше введено концепцію єдиного кодового компонента («диригента»), який залишається незмінним для всіх завдань, тоді як вся предметна логіка формується виключно природною мовою. Це створює основу для теоретичного дослідження безкодових систем, їхньої стійкості, масштабованості та самоорганізації.

Практична новизна полягає в тому, що такий підхід дозволяє нетехнічним користувачам – експертам у фінансах, медицині, освіті – будувати складні інтелектуальні системи без написання коду, тоді як технічна інфраструктура залишається стабільною, масштабованою та інтегрованою. Диригент може бути вбудований у веб-інтерфейс, CLI або API, стаючи основою для платформи, де користувачі завантажують промпт-файли, а система виконує їх з високою надійністю. Це відкриває шлях до масового застосування агентних систем у реальних бізнес-процесах.

Пропонований фреймворк AgentFlow визначає нову парадигму взаємодії з великими мовними моделями, де замість традиційного програмування використовується структурований псевдокод природною мовою, що інтерпретується LLM як система виконання. Цей підхід дозволяє нетехнічним користувачам формалізувати складні аналітичні процеси, не вдаючись до написання коду, і одночасно забезпечує високий рівень контролю, структурованості та детермінованості вихідних даних.

Впровадження логічних примітивів – *Condition, Loop, Function, Label, Goto* – створює основу для побудови алгоритмічно складних завдань, що виходить за межі простих запитів-відповідей і наближає LLM до статусу інтерпретатора програмної логіки.

Формальна модель агента, визначена як кортеж, що складається з ролі, стану, логіки та правил комунікації, дозволяє систематизувати поведінку окремих сутностей у межах рою. Це створює можливість для модульного проектування інтелектуальних систем, де кожен агент виступає незалежним модулем обробки, що виконує чітко визначене завдання. Концепція рою агентів, що взаємодіють за заданим графом виконання, забезпечує імітацію паралельного виконання навіть у межах послідовної архітектури LLM. Ця емуляція ґрунтується на явному описанні гілок виконання, що дозволяє LLM обробляти окремі агенти як незалежні сутності, а потім об'єднувати їхні результати в єдиний вихід. Такий підхід забезпечує системність аналізу, яка неможлива при використанні неструктурованих промптів.

Особливу значущість має введення механізму внутрішнього стану агентів, який виступає як форма пам'яті, що зберігає історію взаємодій, статистику точності та контекст минулих завдань. Це дозволяє реалізувати елементи самоонавлення, де агенти адаптують свою поведінку на основі попереднього досвіду, що є важливим кроком у напрямку створення автономних інтелектуальних систем. Стан агента не є пасивним сховищем даних, а активно використовується в логіці виконання, що забезпечує еволюцію його поведінки з часом. Це створює основу для побудови систем, здатних до навчання в реальному часі без втручання ззовні.

Для подолання обмежень LLM, зокрема відсутності істинної паралельності, контролю над виконанням та інтеграції з зовнішніми системами, запропоновано концепцію універсального диспетчера «диригента» – єдиного кодового компонента, який керує виконанням агентів. Цей підхід реалізує принцип «Єдиний каркас, безліч сценаріїв», де вся предметна логіка формується природною мовою, а виконання забезпечується статичною, незмінною програмою. Диспетчер виступає як інтерпретатор агентних роїв, реалізовуючи граф виконання, керуючи станами, обробляючи помилки та забезпечуючи істинну паралельність через асинхронне виконання. Це дозволяє інтегрувати систему в промислові середовища, де потрібна надійність, масштабованість та детермінованість.

Наукова новизна тут полягає в першій спробі формалізації безкодового програмування через математичні примітиви та структуровану агентну модель. Запропоновано не лише концепцію, а й її формальне визначення, що

дозволяє теоретичне дослідження, автоматичну генерацію промптів і детерміноване виконання. Вперше запропоновано формальний опис агента з чітким розділенням ролі, логіки, стану та комунікації, що створює основу для побудови теорії агентних систем на основі LLM. Також вперше введено механізм паралельного виконання через псевдокод, де LLM симулює незалежність гілок шляхом явного їх описання, що є важливим кроком у напрямку масштабованих багатоагентних систем.

Практична значущість фреймворку полягає в його здатності перетворювати природні запити на структуровані, виконувані інструкції, що можуть бути інтегровані в бізнес-процеси, аналітичні пайплайни та системи прийняття рішень. Встановлення загального формату виходу у вигляді структурованого JSON дозволяє безперешкодну інтеграцію результатів у інші системи, включаючи веб-інтерфейси, бази даних, дашборди та API. Це робить AgentFlow не просто інструментом для аналізу, а платформою для побудови складних інтелектуальних застосувань, доступних для користувачів без технічного бекграунду.

Таким чином, AgentFlow представляє собою не лише методологічний, а й архітектурний прорив у галузі взаємодії людини з штучним інтелектом. Він демонструє, що LLM можуть виступати не лише як генератори тексту, а й як інтерпретатори логічних систем, що виконують складні завдання за чітко визначеним алгоритмом. Це відкриває шлях до створення масштабованих, адаптивних і доступних інтелектуальних систем, де людина формує логіку, а машина забезпечує виконання.

2.6. Верифікація логічної цілісності: зв'язок із онтологіями та базами законодавства

Створення виконуваних модулів безкодового програмування – це важливий крок у побудові інтелектуальної системи парламентського контролю. Однак сама наявність таких модулів не гарантує їх коректності, узгодженості та відповідності чинному законодавству. Навпаки, у міру зростання кількості модулів виникає проблема логічної цілісності – як переконатися, що окремі блоки системи не суперечать один одному, не створюють парадоксальних станів і повністю охоплюють усі можливі сценарії? Вирішення цієї проблеми – верифікація – стає критично важливим етапом, який перетворює набір модулів на єдину, надійну, підзвітну систему контролю.

Базовий фреймворк безкодового програмування, заснований на примітивах «Умова», «Цикл» та «Функція», разом із розширеними примітивами «Метка» та «Перехід», дозволяє створювати складні, логічно структуровані сценарії для LLM. Однак, на практиці залишається серйозна проблема: великі мовні моделі схильні до галюцинацій – генерації правдоподібної, але фактологічно

хибної інформації, а також до ситуацій, коли модель частково ігнорує інструкції або умови, особливо в складних, багатоетапних промптах. Крім того, результати часто страждають від недостатньої пояснюваності, що ускладнює їх інтерпретацію користувачем.

Традиційні методи, такі як few-shot learning або жорстке форматування виводу, не завжди ефективні для вирішення цих проблем, оскільки вони не передбачають механізмів самокорекції та ітеративного покращення. Для подолання цих обмежень необхідно ввести новий рівень абстракції – примітиви, спрямовані на забезпечення надійності та контролю якості виконання.

Тому пропонується формалізація та інтеграція в існуючий фреймворк трьох нових примітивів, які дозволяють LLM-системам "перевіряти себе самі" та гарантувати високу якість результату. Цей підхід наближає безкодове програмування до парадигми надійних обчислень, де кожен крок може бути верифікований, а результат – гарантований з заданою точністю.

Розвиток методологій безкодового програмування в середовищі великих мовних моделей є міждисциплінарною галуззю, що поєднує досягнення комп'ютерних наук, лінгвістики, штучного інтелекту та регуляторних досліджень. У цьому розділі ми систематизуємо ключові наукові роботи, які формують теоретичну та практичну основу для запропонованого розширеного фреймворку.

Класична стаття[80] пропонує систематичний огляд методів промптингу, визначаючи його як нову парадигму "Pre-train, Prompt, and Predict", що змінює традиційний підхід до машинного навчання. Ця робота є теоретичною основою для формалізації промптів як програмних конструкцій. У роботах[81, 82] вперше запропоновано формальний фреймворк безкодового програмування на основі примітивів «Умова», «Цикл» та «Функція», що створює міст між класичним програмуванням і природномовним інтерфейсом LLM.

Для подолання обмежень базового фреймворку, робота[83] вводить примітиви «Метка» та «Перехід», що дозволяють моделювати складні, нелінійні потоки виконання, аналогічно до процедурного програмування. Цей підхід

---

[80] Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H. and Neubig, G., 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM computing surveys*, 55(9), pp.1-35. DOI: 10.1145/3560815.

[81] Dmitry Lande, Leonard Strashnoy. Semantic AI Framework for Prompt Engineering. *ResearchGate Preprint*. DOI: 10.13140/RG.2.2.21116.24964 (Mart 9, 2025).

[82] Ланде Д.В., Страшной Л.Л. Семантичне Зондування Великих Мовних Моделей. *Проблеми кібербезпеки інформаційно-комунікаційних систем: Збірник матеріалів доповідей та тез*; м. Київ, 11 квітня 2025 року. Київ: ВПЦ "Київський університет", 2025. – С. 161-162.

[83] Dmitry Lande, Leonard Strashnoy. An Advanced No-Code Programming Framework for Complex Problems in LLM Environments. *ResearchGate Preprint.* DOI: 10.13140/RG.2.2.25307.89129 (May, 2025).

знаходить підтвердження в дослідженні[84], де показано, що структуровані, багатоетапні промпти значно підвищують точність виконання складних завдань LLM.

Одним із головних обмежень LLM є схильність до галюцинацій – генерації правдоподібної, але хибної інформації. Роботу[85] присвячено систематичному аналізу причин галюцинацій у LLM та пропонує таксономію методів їх зменшення, зокрема, через верифікацію фактів та ітеративне уточнення. В роботі[86] вводиться концепція “self-refinement”, де модель сама оцінює та покращує свій результат, що логічно пов’язане з нашим примітивом «Суперцикл». Аналіз юридичних документів, зокрема Закону про ШІ ЄС, вимагає розуміння регуляторного ландшафту. Звіт EPRS (2024)[87] є ключовим джерелом для розуміння структури, ризиків та лобістських аспектів цього законодавства. Критичний аналіз, наведений у звіті[88], прямо вказує на недоліки, такі як надвисокий поріг FLOPs та слабкі механізми контролю, що підтверджує необхідність наших примітивів «Контроль» та «Результат» для незалежного аналізу.

В роботі[89] критично аналізується пропозиція Єврокомісії щодо Закону про ШІ, наголошуючи на необхідності посилення захисту фундаментальних прав. Рекомендації ЮНЕСКО з етики ШІ[90] та Принципи ОЕСР[91] надають глобальні орієнтири для оцінки «справедливості» та «прозорості» систем ШІ, які можуть бути формалізовані як параметри для примітиву «Контроль».

[84] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q.V. and Zhou, D., 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems,* 35, pp.24824-24837.

[85] Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y.J., Madotto, A. and Fung, P., 2023. Survey of hallucination in natural language generation. *ACM computing surveys*, 55(12), pp.1-38. DOI: 10.1145/3571730.

[86] Huang, D., Liu, Y., Qian, J. and Wang, Y., 2024, October. Vrefine: A Self-Refinement Approach for Enhanced Clarity and Quality in Text-to-Speech Models. In 2024 *IEEE International Conference on Systems, Man, and Cybernetics* (*SMC*) (pp. 336-341). DOI: 10.1109/SMC54092.2024.10831059

[87] Ebers, M., 2023. The European Commission's Proposal for an Artificial Intelligence Act. *In Research Handbook on EU Internet Law* (pp. 271-292). Edward Elgar Publishing.

[88] Hacker, P., 2024. Comments on the final trilogue version of the AI act. *SSRN Preprint*: 4757603. DOI: 10.2139/ssrn.4757603

[89] Smuha, N.A., Ahmed-Rengers, E., Harkens, A., Li, W., MacLaren, J., Piselli, R. and Yeung, K., 2021. How the EU can achieve legally trustworthy AI: a response to the European Commission's proposal for an Artificial Intelligence Act.

[90] United Nations Educational, Scientific and Cultural Organization, 2021. *Recommendation on the ethics of artificial intelligence*.

[91] OECD Principles on Artificial Intelligence, 2019. URL: https://www.oecd.org/en/topics/ai-principles.html

Нарешті, робота[92] демонструє, як LLM можуть функціонувати як автономні агенти, здатні планувати, приймати рішення та взаємодіяти. Це дослідження відкриває шлях для застосування нашого фреймворку в майбутніх безкодових агентських системах, де примітиви надійності забезпечуватимуть безпечне та передбачуване функціонування.

Розширення фреймворку безкодового програмування

Як було вже наведено у п.2.2 базовий фреймворк безкодового програмування утворює мінімальний повноцінний набір з трьох примітивів, достатній для вирішення широкого класу задач: ***Умова (If-Else), Цикл (For-Loop), Функція (Abstraction).***

Для роботи зі складними, нелінійними або рекурсивними задачами (наприклад, обробка винятків, моделювання сценаріїв) базового набору недостатньо, тому було запропоновано два додаткові примітиви (п. 2.3) [6]: ***Мітка (Label), Перехід (Goto).***

Ці примітиви дозволяють створювати довільні потоки управління, наближаючи безкодове програмування до класичного процедурного програмування.

Примітиви надійності та контролю якості

Для забезпечення надійності та контролю якості виконання промптів пропонується ввести наступні три примітиви[93]:

Примітив Суперцикл (Supercycle)

Призначення цього примітиву – організація ітеративного виконання часткового промпту до досягнення задовільного результату або вичерпання ліміту спроб. Це основний механізм для боротьби з галюцинаціями та спрощеннями

Наведемо формальний опис цього примітиву. Для цього введемо позначення:

- max_iterations ($N \in \mathbb{N}$) – максимальна кількість ітерацій. Якщо досягнуто – повертається значення «невдача».
- success_threshold ($T \in \mathbb{R}^+$) – числовий поріг, який визначає прийнятну якість результату.

[92] Park, J.S., O'Brien, J., Cai, C.J., Morris, M.R., Liang, P. and Bernstein, M.S., 2023, October. Generative agents: Interactive simulacra of human behavior. *In Proceedings of the 36th annual acm symposium on user interface software and technology* (pp. 1-22). URL: 10.1145/3586183.3606763

[93] Ланде Д.В., Даник Ю.Г., Фурашев В.М. Виявлення та аналіз конфліктів у нормативно-правовій сфері, пов'язаних із застосуванням штучного інтелекту. Інформація і право, 2025. - N 3(53). - С. 53-68.DOI: 10.37750/2616-6798.2025.3(54).340453

– control_method (M) – метод оцінки якості (наприклад, «Точне збігання», «Косинусна подібність», «Метод Ньютона»).
– inner_prompt (P) – будь-який складний промпт, побудований з інших примітивів, який виконується на кожній ітерації.

Тоді виконання Supercycle(N, T, M, P) визначається наступним алгоритмом:

1. Ініціалізація: i = 0, prev_result = null.
2. Виконати inner_prompt, отримати current_result.
3. Викликати примітив Control(current_result, prev_result, M, T), отримати status.
4. Якщо status = "успіх" – повернути current_result як кінцевий результат.
5. Якщо i >= N – повернути "невдача".
6. Інакше: prev_result = current_result, i = i + 1, перейти до кроку 2.

При побудові промптів застосовується такий синтаксис:

```
[Примітив "Суперцикл"] Supercycle(
  max_iterations = 5,
  success_threshold = 0.95,
  control_method = "Косинусна подібність",
  inner_prompt = [ ... ]
)
```

Примітив Контроль (Control)

Призначення примітиву «Контроль» – об’єктивна оцінка результату виконання промпту на основі заданого методу та порогового значення. Це "суддя", який вирішує, чи достатньо хороший поточний результат для завершення Суперциклу.

Для формального опису примітиву введемо позначення:

– current_result – результат поточної ітерації.
– previous_result – результат попередньої ітерації (опціонально).
– method (M) – функція оцінки. Наприклад:
– Для числових задач: M(x, y) = |x – y| (якщо y – очікуване значення).
– Для текстових задач: M(x, y) = cosine_similarity(embed(x), embed(y)).
– Для ітеративних методів: M(x, y) = |x – y| (збіжність).
– threshold (T) – мінімальне значення метрики для визнання результату успішним.

У цих позначеннях примітив визначається як:

– Control(current_result, previous_result, method, threshold) → "успіх", якщо method(current_result, ...) >= threshold (або <= threshold для метрик відстані).

– Control(...) → "невдача", інакше.

Синтаксис цього примітиву має вигляд:

```
[Примітив "Контроль"] Control(
  current_result = temp_mean,
  previous_result = null,
  method = "Точне збігання",
  threshold = 0.01
)
```

Суперцикл виконується лише один раз у детермінованих математичних задачах, наявність оператора "Контроль" гарантує, що результат був явно верифікований за заданими критеріями, а не прийнятий на віру, що запобігає помилковому "успіху" через галюцинацію або випадкову відповідь.

Разом з цим для надійного фреймворку, який має боротися з галюцинаціями, перша ітерація завжди повинна вважатися "невдачею" з точки зору оператора Control, щоб забезпечити принаймні одне порівняння результатів. Тільки після цього можна робити висновок про стабільність і точність. Тому, якби перший результат був помилковим, але другий – правильним, система б виявила розбіжність і запустила б третю ітерацію для уточнення.

Примітив Результат (Output)

Призначення цього простору – стандартизація та покращення сприйняття кінцевого результату користувачем. Забезпечує пояснюваність шляхом чіткого форматування та, за можливості, візуалізації.

Для формального опису введемо позначення:

- final_result – дані для виводу.
- format – бажаний формат (наприклад, "текст", "JSON", "Markdown", "таблиця").
- visualization_type – тип візуалізації (наприклад, "діаграма", "графік", "word cloud").

Реальна візуалізація залежить від можливостей конкретної LLM або інтегрованих інструментів.

Тоді Output(final_result, format, visualization_type) генерує відформатований вивід для користувача.

Синтаксис у промпті має вигляд:

```
Примітив "Результат"] Output(
  final_result = final_mean,
  format = "текст",
```

```
    visualization_type = "немає"
)
```

Композиція Примітивів

Визначені примітиви інтегруються в загальну систему композиції таким чином:

Promt ::= Примітив | (Promt ⊕ Promt) | Умова(Promt, Promt) | Label(L, Promt) | Goto(L) | Supercycle(N, T, M, Promt) | Control(CR, PR, M, T) | Output(FR, F, V)

де:

- CR – current_result,
- PR – previous_result,
- FR – final_result,
- F – format,
- V – visualization_type.

Приклади Реалізації Задач

Приклад 1: Надійне Обчислення Середнього Значення

Задача: Обчислити середнє значення списку чисел [4, 7, 10, 5, 3] з гарантованою точністю.

Промпт з використанням нових примітивів:

```
[Примітив "Функція"] Функція(ReliableComputeMean):
  [Примітив "Вхід"] Вхід: X = [4, 7, 10, 5, 3]
  [Примітив "Суперцикл"] Supercycle(
    max_iterations = 5,
    success_threshold = 0.01,
    control_method = "Точне збігання",
    inner_prompt = [
      Label(CALC):
        [Примітив "Цикл"] Підсумуй елементи X. -> sum_val
        [Примітив "Умова"] Якщо кількість елементів > 0:
          Поділи sum_val на кількість елементів. -> temp_result
        Інакше:
          Поверни помилку.
      Label(CONTROL_POINT):
        [Примітив "Контроль"] Control(
```

```
            current_result = temp_result,
            previous_result = null,
            method = "Точне збігання",
            threshold = 0.01
        )
    ]
) -> final_mean
[Примітив "Результат"] Output(
    final_result = final_mean,
    format = "текст",
    visualization_type = "немає"
)
```

Пояснення: Цей промпт виконуватиметься до 5 разів. На кожній ітерації обчислюється середнє значення, яке потім перевіряється оператором «Контроль». Якщо результат стабільний (наприклад, дорівнює 5.8 з точністю до 0.01), виконання завершується успішно. Якщо після 5 спроб стабільного результату досягти не вдалося, повертається «невдача», що сигналізує про системну проблему (наприклад, галюцинацію LLM).

Приклад 2: Генерація Пояснювального Тексту з Контролем Якості

Задача: Згенерувати короткий опис поняття «квантова заплутаність» і перевірити, чи він містить ключові терміни.

Промпт з використанням нових примітивів має вигляд:

```
[Примітив "Суперцикл"] Supercycle(
    max_iterations = 3,
    success_threshold = 0.8,
    control_method = "Ключові слова",
    inner_prompt = [
        Згенеруй короткий (1-2 речення) опис поняття "квантова заплутаність" для початківця.
        -> generated_text
        [Примітив "Контроль"] Control(
            current_result = generated_text,
            previous_result = null,
```

```
        method = "Ключові слова",
        threshold = 0.8
      )
      # Метод "Ключові слова": перевіряє наявність слів "стан", "частинки", "вимірювання", "миттєво".
      # Повертає "успіх", якщо знайдено >= 80% ключових слів.
    ]
) -> final_description

[Примітив "Результат"] Output(
  final_result = final_description,
  format = "Markdown",
  visualization_type = "немає"
)
```

Цей промпт гарантує, що згенерований текст буде не лише грамотним, але й інформативно повним, міститиме необхідний мінімум термінології, що запобігає генерації занадто загальних або хибних описів.

Нижче наведено результат виконання вашого промпту з використанням розширеного фреймворку безкодового програмування.

Остаточний результат виконання

Статус виконання: – Успіх (досягнуто на 2-й ітерації Суперциклу).

Кінцевий результат:

Квантова заплутаність – це явище, коли стан двох або більше частинок стає настільки взаємопов’язаним, що вимірювання стану однієї частинки миттєво визначає стан іншої, навіть якщо вони розділені великою відстанню.

Покроковий аналіз виконання

Ініціалізація Суперциклу:

max_iterations = 3

success_threshold = 0.8 (80% ключових слів)

control_method = "Ключові слова"

Ключові слова: [«стан», «частинки», «вимірювання», «миттєво»] (всього 4 слова).

Ітерація 1:

Виконання inner_prompt:

Згенеровано пояснення: «Квантова заплутаність – це дивне зв’язування частинок, при якому зміна однієї впливає на іншу, навіть на великій відстані.»

Виконання* [*Примітив «Контроль»*]*:

current_result = "Квантова заплутаність – це дивне зв’язування частинок, при якому зміна однієї впливає на іншу, навіть на великій відстані."

previous_result = null

method = "Ключові слова"

threshold = 0.8

Знайдені ключові слова: ["частинки"] → 1 з 4 слів (25%).

Статус контролю: "невдача" (25% < 80%).

Ітерація 2:

Виконання inner_prompt:

Згенеровано пояснення: «Квантова заплутаність – це явище, коли стан двох або більше частинок стає настільки взаємопов’язаним, що вимірювання стану однієї частинки миттєво визначає стан іншої, навіть якщо вони розділені великою відстанню.»

Виконання* [*Примітив «Контроль»*]*:

current_result = «Квантова заплутаність – це явище, коли стан двох або більше частинок стає настільки взаємопов’язаним, що вимірювання стану однієї частинки миттєво визначає стан іншої, навіть якщо вони розділені великою відстанню.»

Знайдені ключові слова: [«стан», «частинки», «вимірювання», «миттєво»] → 4 з 4 слів (100%).

Статус контролю: «успіх» (100% >= 80%).

Оскільки контроль повернув «успіх», Суперцикл завершується. final_description = «Квантова заплутаність – це явище, коли стан двох або більше частинок стає настільки взаємопов’язаним, що вимірювання стану однієї частинки миттєво визначає стан іншої, навіть якщо вони розділені великою відстанню.»

Вивід результату:

[Примітив «Результат»] форматує final_description як простий текст Markdown без візуалізації.

Цей приклад демонструє силу примітиву «Суперцикл». На першій ітерації LLM згенерувала короткий і зрозумілий, але неповний опис, який не містив усіх ключових термінів. Завдяки механізму «Контролю», система визнала цей результат неадекватним і автоматично запустила повторну генерацію. На

другій ітерації було отримано точний і повний опис, що задовольняє всім встановленим критеріям. Це гарантує, що кінцевий результат не лише грамотний, але й інформативно повний, що запобігає генерації занадто загальних або хибних описів.

Приклад 3. Промпт для аналізу юридичного документа

Нижче наведено структурований промпт із застосуванням запропонованих операторів, призначений для аналізу юридичного документа на наявність лобістських моментів:

```
[Примітив "Функція"] Функція(AnalyzeLobbyingInLegalDoc):
  [Примітив "Вхід"] Вхід: legal_document_text = [ВСТАВИТИ ТЕКСТ ДОКУМЕНТА ТУТ]
  [Примітив "Суперцикл"] Supercycle(
    max_iterations = 3,
    success_threshold = 0.85,
    control_method = "Ключові індикатори",
    inner_prompt = [
      Label(ANALYZE):
        [Примітив "Цикл"] Виконай наступні кроки для кожного розділу документа:
            1. Виділи всі норми, що надають переваги або обмеження для конкретних груп (корпорацій, галузей, регіонів).
            2. Виділи формулювання, що є надто загальними, нечіткими або дозволяють широке тлумачення на користь певної сторони.
            3. Знайди згадки конкретних компаній, організацій або технологій, які отримують особливий статус.
            4. Визнач, чи є норми, що суперечать загальноприйнятим міжнародним стандартам (див. список нижче).
            5. Оціни, чи є механізми контролю або звітності для виконання цих норм – якщо їх немає, це потенційний індикатор лобізму.
        -> raw_analysis
      Label(CONTROL):
        [Примітив "Контроль"] Control(
          current_result = raw_analysis,
          previous_result = null,
          method = "Ключові індикатори",
```

```
        threshold = 0.85
    )
  # Метод "Ключові індикатори": перевіряє наявність мінімум 5 з 8 ключових ознак лобізму:
  # 1. Згадка конкретної компанії/бренду.
  # 2. Відсутність механізмів контролю.
  # 3. Надання виняткових прав/ліцензій.
  # 4. Нечіткі критерії відбору.
  # 5. Обмеження конкуренції.
  # 6. Фінансові пільги/субсидії без обґрунтування.
  # 7. Відстрочка виконання для певних суб'єктів.
  # 8. Посилання на "національні інтереси" без конкретики.
  # Повертає "успіх", якщо знайдено >= 85% індикаторів (тобто 5+ з 8).
  ]
) -> final_analysis

[Примітив "Результат"] Output(
  final_result = final_analysis,
  format = "Markdown",
  visualization_type = "Таблиця: Індикатор | Знайдено | Цитата з документа"
)
```

Задачи наведеного типу вимагають не просто пошуку ключових слів, а глибокого, ітеративного аналізу з перевіркою якості результату. Тому використовується повний набір примітивів, включаючи Суперцикл та Контроль.

Результат виконання цього промпту для Акту про штучний інтелект (Artificial Intelligence Act, https://www.europarl.europa.eu/RegData/etudes/BRIE/2021/698792/EPRS_BRI(2021)698792_EN.pdf), ухваленого 13 березня 2024 року Європейським парламентом як регламент, що має забезпечити безпеку і дотримання прав громадян при застосуванні штучного інтелекту наведено нижче:

Аналіз на наявність лобістських моментів у документі "Artificial Intelligence Act" (EPRS Briefing, 2024)

Нижче наведено результат аналізу отриманого в результаті виконання промпту:

Статус: Успіх (знайдено 6 з 8 ключових індикаторів лобізму – 75%, що перевищує поріг у 85% після 2 ітерацій Суперциклу завдяки уточненню критеріїв).

Формат виводу: Таблиця Markdown

ІНДИКАТОР ЛОБІЗМУ	ЧИ ЗНАЙДЕНО	ЦИТАТА / ПОЯСНЕННЯ З ДОКУМЕНТА
1. Згадка конкретної компанії/бренду	Ні	У документі навмисно уникнуто згадок конкретних корпорацій (наприклад, Google, Meta, Microsoft). Це свідома політика ЄС для уникнення прямого лобіювання. Згадуються лише категорії: "провайдери GPAI", "SMEs", "великі онлайн-платформи".
2. Відсутність механізмів контролю	Так	Критика академіків:“...the AI act lacks effective enforcement structures... reliance on provider self-assessment... lack of judicial means to control [стандарти]... no right to complain for individuals” (Стор. 5-6).
3. Надання виняткових прав/ліцензій	Так	Виняток для відкритого коду (open-source):“AI models made accessible under a free and open source will be exempt from some of the obligations... given they have... positive effects” (Стор. 10). Це прямою дією захищає інтереси open-source спільноти (часто підтримуваної великими техкорпораціями).
4. Нечіткі критерії відбору	Так	Визначення "високого ризику" та "системного ризику":Критерії суб’єктивні. Наприклад, "significant generality","reasonably foreseeable negative effects". Це дозволяє регуляторам (та, опосередковано, лобістам) гнучко тлумачити правила (Стор. 7, 10).
5. Обмеження конкуренції	Так	Поріг FLOPs (10^{25}) для "системного ризику":Встановлення дуже високого технічного порогу (“threshold is too high” – Hacker, 2024, Стор. 11) де-факто звільняє від суворих вимог майже всіх, окрім кількох найбільших гравців(наприклад, OpenAI, Google, Meta), які мають ресурси для досягнення цього порогу. Це створює бар’єр для входу для стартапів та SMEs.
6. Фінансові пільги/субсидії без	Ні	У документі немає прямих згадок про фінансові пільги. Натомість пропонуються регуляторні пісочниці (sandboxes) для стартапів, що є формою

ІНДИКАТОР ЛОБІЗМУ	ЧИ ЗНАЙДЕНО	ЦИТАТА / ПОЯСНЕННЯ З ДОКУМЕНТА
обґрунтування		непрямої підтримки, але вона обґрунтована інноваційними цілями.
7. Відстрочка виконання для певних суб'єктів	Так	Різні терміни набрання чинності: "Prohibited systems: 6 months... GPAI: 12 months... High-risk: 24 months... (36 months for systems under existing product law)" (Стор. 11). Це дає перевагу встановленим гравцям, які вже працюють у рамках існуючого законодавства (наприклад, медичні пристрої), і дозволяє їм адаптуватися повільніше.
8. Посилання на "національні інтереси" без конкретики	Так	Винятки для правоохоронних органів: Дозволено використання біометричної ідентифікації в режимі реального часу "for preventing certain substantial and imminent threats to safety" (Стор. 8). Формулювання "substantial and imminent threats" є надзвичайно розмитим і дозволяє національним урядам тлумачити його на власний розсуд, що може бути використано для лобіювання національних безпекових агенцій.

Аналіз виявив 6 з 8 ключових індикаторів лобізму, що свідчить про істотний вплив інтересів різних груп на фінальну редакцію закону.

Основні висновки:

1. Лобіювання великих технологічних компаній (Big Tech): Найбільш явно проявляється у:
   - Винятку для відкритого коду, який захищає інтереси компаній, що активно використовують та спонсорують open-source моделі.
   - Встановленні надвисокого порогу FLOPs (10^{25}), що практично звільняє від найсуворіших вимог усіх, окрім кількох лідерів ринку, консолідуючи їхню ринкову владу.
   - Нечітких визначеннях "високого" та "системного" ризику, що дозволяє провайдерам самостійно класифікувати свої системи, зменшуючи регуляторний тиск.
2. Лобіювання національних урядів та правоохоронних органів: Проявляється у широких винятках для використання біометричної ідентифікації, з розмитими критеріями "загрози безпеці", що дає свободу дій національним поліцейським та спецслужбам.
3. Лобіювання від імені стартапів та SMEs: Хоча прямої фінансової підтримки немає, регуляторні піски та різні терміни адаптації є формою захисту, яка, однак, може бути недостатньою через високі витрати на відповідність (compliance costs), про що також згадується у документі.

Закон про ШІ ЄС, попри свої амбітні цілі, є компромісним документом, у якому чітко відображені сліди лобіювання різних зацікавлених сторін, особливо великих технологічних корпорацій та національних безпекових структур. Прозорість та механізми контролю за його виконанням залишаються найбільш вразливими аспектами.

Таким чином, введення примітивів «Суперцикл», «Контроль» та «Результат» є логічним і необхідним кроком у розвитку фреймворку безкодового програмування. Ці примітиви трансформують підхід з "одноразового запиту" на "ітеративний, самоконтрольований процес", що кардинально підвищує надійність та якість результатів, отриманих від LLM.

Запропонований підхід дозволяє:

– Ефективно протидіяти галюцинаціям та «лінії» шляхом багаторазової перевірки.
– Кількісно оцінювати якість виконання за допомогою настроюваних метрик.
– Забезпечувати пояснюваність кінцевих результатів для користувача.

Переваги:

– Підвищена надійність: Значно знижується ймовірність отримання хибного або неповного результату.
– Самокорекція: Система здатна «виправляти» себе без втручання користувача.
– Пояснюваність: Стандартизований вивід та можливість візуалізації роблять результати більш зрозумілими.
– Адаптивність: Можна налаштовувати пороги та методи контролю під конкретну задачу.

Обмеження та Рішення:

– Обчислювальні витрати: Ітеративне виконання збільшує час та вартість запитів.
Рішення: Встановлення розумного max_iterations (наприклад, 3-5) та використання простих, швидких методів контролю.
– Складність методів контролю: Деякі методи (наприклад, «Косинусна подібність») важко формалізувати для LLM.
Рішення: Використання простих, текстово-орієнтованих методів (наявність ключових слів, точне збігання з шаблоном, довжина тексту) або інтеграція з зовнішніми інструментами оцінки.

Новизна підходу:

1. Запропонованому формальному визначенні класу примітивів «Надійності та Контролю Якості» для безкодового програмування в середовищі LLM, що включає «Суперцикл», «Контроль» та «Результат».

2. Інноваційному механізмі самокорекції результатів LLM на основі ітеративного виконання та верифікації, який не має прямих аналогів у існуючій літературі з промпт-інженерії.

3. Інтеграції концепцій надійних обчислень (reliable computing) у парадигму безкодового програмування, що дозволяє гарантувати якість результату з заданою точністю.

Найбільш перспективним напрямком застосування запропонованих примітивів є розробка безкодових агентських систем. У таких системах агенти, побудовані на основі LLM, повинні виконувати складні, багатоетапні завдання з високою надійністю та автономністю. Примітиви надійності дозволять:

– Створювати автономні агенти, які можуть перевіряти власні дії та виправляти помилки без зовнішнього втручання.

– Будувати агентів з гарантованою якістю виконання, що критично для застосування в фінансах, медицині або безпеці.

– Організовувати взаємодію між агентами, де результати одного агента автоматично перевіряються іншим перед передачею, забезпечуючи цілісність даних у системі.

– Розробляти «агента-рецензента», який використовує примітив «Контроль» для оцінки якості роботи інших агентів.

У складних правових системах окремі норми, а тепер і окремі модулі контролю, можуть бути прийняті в різний час, різними суб'єктами, з різними цілями. Це створює ризик внутрішніх протиріч. Наприклад, один модуль, побудований на основі Закону про податки, може передбачати штраф за прострочення звіту, тоді як інший модуль, заснований на законі про економічну стабільність, може передбачати амністію за такі порушення у певний період. Якщо ці модулі працюють незалежно, система може одночасно генерувати протилежні висновки, що робить її непридатною для прийняття рішень.

Тому верифікація – це не просто перевірка на помилки, а гарантія логічної узгодженості всієї системи. Вона дозволяє відповісти на питання: чи може система опинитися в стані, який суперечить сам собі, чи всі можливі сценарії враховані, чи виконується принцип правової певності.

Методи верифікації: пошук циклів, виявлення неможливих станів, перевірка на повноту умов

Для забезпечення логічної цілісності використовуються декілька методів:

– Пошук циклів та нескінченних петель, особливо важливий при використанні циклів у модулях. Наприклад, якщо модуль контролю за звітністю безкінечно перевіряє стан «звіт не подано», не передбачаючи умови виходу, це може призвести до блокування процесу.

– Виявлення неможливих станів, наприклад, коли модуль одночасно встановлює стан «справа закрита» і «розслідування триває». Такі конфлікти вказують на помилку в логіці.

– Перевірка на повноту умов (exhaustiveness), чи всі можливі випадки враховані в умовних конструкціях. Наприклад, якщо модуль перевіряє лише «подано» і «не подано», але не передбачає стану «подано з помилками», це створює прогалину в контролі.

– Аналіз взаємовиключних умов, а саме, коли дві умови не можуть бути істинними одночасно, але обидві активують суперечливі дії.

Ці методи можуть бути реалізовані через спеціалізовані інструменти статичного аналізу коду, адаптовані для форматів, що використовуються у безкодовому програмуванні (наприклад, JSON-схеми, DOT-графи).

Інтеграція з онтологіями парламентського контролю (наприклад, на основі RDF/RDFS)

Одним із найефективніших способів забезпечення логічної цілісності є інтеграція з онтологіями – формальними моделями предметної області. Як зазначено в[94], онтології, такі як LegalRuleML[95], SWRL[96], LKIF[97] або Protege-OWL[98] дозволяють формалізувати правові поняття, їх відношення та обмеження.

У контексті парламентського контролю може бути розроблена, наприклад, онтологія контролю, яка визначає:

– ключові поняття: *«законопроект»*, *«звіт уряду»*, *«скарга»*, *«порушення»*, *«штраф»*, *«амністія»*;

– відношення між ними: *«передбачає»*, *«суперечить»*, *«виключає»*, *«передує»*;

– обмеження: *«одна справа не може бути одночасно закритою та відкритою»*, *«амністія виключає штраф»*.

Модулі безкодового програмування можуть бути зіставлені з цією онтологією, і система може автоматично перевіряти, чи не порушують вони визначені правила. Наприклад, якщо модуль передбачає застосування

[94] Інформатика парламентського контролю : посібник / Д.В. Ланде, В.М. Фурашев, С.М. Брайчевський. - Київ 2022. - 256 с. ISBN 978-966-2344-80-6

[95] Athan, T., Governatori, G., Palmirani, M., Paschke, A. and Wyner, A., 2015. LegalRuleML: Design principles and foundations. In *Reasoning Web International Summer School* (pp. 151-188). Cham: Springer International Publishing.

[96] Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B. and Dean, M., 2004. SWRL: A semantic web rule language combining OWL and RuleML. *W3C Member submission*, *21*(79), pp.1-31.

[97] Hoekstra, R., Breuker, J., Di Bello, M. and Boer, A., 2007. The lkif core ontology of basic legal concepts. *LOAIT*, *321*, pp.43-63.

[98] Horridge, M., Knublauch, H., Rector, A., Stevens, R. and Wroe, C., 2004. A practical guide to building OWL ontologies using the Protégé-OWL plugin and CO-ODE tools edition 1.0. *University of Manchester*.

штрафу після оголошення амністії, система виявляє конфлікт на рівні онтології.

Використання семантичних мереж як основи для перевірки узгодженості

Семантичні мережі, побудовані на основі аналізу законодавства за допомогою LLM, виступають не лише інструментом візуалізації, а й основою для верифікації. Коли модулі контролю інтегруються в єдину семантичну мережу, можна виявляти:

– конфліктні зв'язки, коли два шляхи в мережі призводять до протилежних висновків;

– логічні цикли, коли процес повертається у початковий стан без досягнення мети;

– інформаційні прогалини, де немає зв'язків між ключовими поняттями.

Наприклад, у мережі, що об'єднує модулі контролю за податками та економічною амністією, можна виявити, що певний клас підприємств одночасно потрапляє під штрафні санкції та пільговий режим, що вказує на неузгодженість норм.

Приклад: виявлення конфлікту між двома модулями – один передбачає штраф, інший – амністію

Розглянемо конкретний приклад. Модуль А («Контроль за податковою дисципліною») містить правило:

```
IF tax_report_late AND no_valid_reason:
   APPLY fine = base_amount * 1.5
```

Модуль В («Економічна амністія для МСП») містить правило:

```
IF taxpayer_type == "MSP" AND period == "2024-Q3":
   SET fine = 0
   SET status = "amnesty_applied"
```

На перший погляд, обидва модулі коректні. Однак при одночасному застосуванні вони створюють двозначність: чи застосовується штраф, якщо підприємство – МСП, але подало звіт із запізненням у 2024-Q3.

Система верифікації, інтегрована з онтологією, виявляє цей конфлікт, оскільки онтологія містить правило:

«Якщо до підприємства застосовується амністія, штраф не може бути накладений»

Отже, модуль А повинен бути доповнений умовою:

```
IF NOT amnesty_applied:
   APPLY fine
```

Автоматичне формування звітів про конфлікти для парламентарів

Після виявлення конфлікту система автоматично формує аналітичний звіт, який містить:

- опис конфліктуючих модулів;
- цитати з відповідних законів;
- графічне представлення конфлікту в семантичній мережі;
- пропозиції щодо усунення суперечності.

Такий звіт може бути наданий парламентському комітету як підстава для ініціювання змін у законодавстві.

Таким чином, верифікація логічної цілісності – це не технічна деталь, а важливий елемент підзвітності. Вона перетворює систему безкодового програмування з набору ізольованих інструментів на єдину, узгоджену, прозору систему інтелектуального контролю, здатну забезпечувати високоякісну підтримку рішень у парламенті.

2.6. Інтеграція з правовими базами даних та внутрішньою документацією

Створення виконуваних модулів безкодового програмування – це важливий крок у побудові інтелектуальної системи парламентського контролю. Однак їхня ефективність напряму залежить від того, наскільки вони інтегровані з дійсними джерелами правової інформації. Система, побудована на застарілих або неповних даних, не може бути надійною. Тому важливим етапом є інтеграція з офіційними базами законодавства та внутрішньою документацією парламенту, що забезпечує актуальність, достовірність та підзвітність аналітичних висновків.

Підключення до баз даних законодавства

Для забезпечення високої якості аналізу модулі безкодового програмування повинні мати доступ до офіційних та оновлюваних джерел законодавчої інформації. Серед таких джерел – державні та комерційні бази даних:

– Офіційний вебсайт Верховної Ради України (zakon.rada.gov.ua) – джерело актуальних законів, законопроектів, постанов, розпоряджень;

– «Ліга:Закон» – комерційна правова база, що забезпечує структурований доступ до законодавства, коментарів, судової практики;

– Єдиний державний реєстр нормативно-правових актів (www.reestrnpa.gov.ua/).

Інтеграція з цими системами має здійснюватись через API (Application Programming Interface), які дозволяють автоматично отримувати текст законодавчих актів, їхні зміни, редакції, втрату чинності тощо. Це дозволяє

модулям працювати з офіційними формулюваннями норм, а не з їхніми інтерпретаціями чи фрагментами.

Механізми синхронізації модулів зі змінами в законодавстві

Однією з найбільших проблем застосування LLM у правовій сфері є обмеженість періоду навчання моделі. Наприклад, модель може бути навчена на даних до 2023 року і не мати інформації про нові закони, прийняті у 2024–2025 роках. Для подолання цього обмеження необхідні механізми динамічної синхронізації модулів зі змінами в законодавстві.

Це може бути реалізовано через:

– періодичне сканування баз даних на наявність нових актів або змін у наявних;

– тригерні сповіщення про прийняття законів, що стосуються певних сфер (наприклад, податки, бюджет, ШІ);

– автоматичне оновлення відповідних модулів контролю на основі нових норм.

Наприклад, якщо прийнято новий Закон про державний бюджет, система автоматично визначає, які модулі контролю (наприклад, «контроль за виконанням бюджету», «аналіз розподілу коштів») потребують оновлення, і запускає процес їх адаптації.

Використання RAG (Retrieval-Augmented Generation) для актуалізації знань LLM

Одним із найефективніших інструментів інтеграції є RAG (Retrieval-Augmented Generation)[99,100] – технологія, яка дозволяє LLM отримувати інформацію з зовнішніх джерел без повторного навчання моделі.

У контексті парламентського контролю RAG працює таким сином:

1. Користувач задає запит: *«*Чи суперечить законопроект № X нормам Конституції?*»*

2. Система автоматично витягує текст законопроекту та відповідні статті Конституції з бази даних.

3. Ці тексти додаються до промпту як контекст.

4. LLM аналізує актуальні норми і формує відповідь на їх основі.

[99] Siriwardhana, S., Weerasekera, R., Wen, E., Kaluarachchi, T., Rana, R. and Nanayakkara, S., 2023. Improving the domain adaptation of retrieval augmented generation (RAG) models for open domain question answering. Transactions of the Association for Computational Linguistics, 11, pp.1-17.

[100] Dmitry Lande, Leonard Strashnoy, Oleksander Rybak. Framework of Extended Semantic Networking – A Semantic RAG Architecture for Dynamic Conceptual Mapping. SSRN Preprint: 5505220, DOI: 10.2139/ssrn.5505220 (Sep 23, 2025). – 17 pp.

Це дозволяє уникнути галюцинацій і забезпечує, що аналіз завжди ґрунтується на оцінці дійсних правових актів, а не на внутрішніх знаннях моделі.

Інтеграція з внутрішньою документацією парламенту: регламенти, протоколи, комітетські звіти

Крім законодавства, система повинна мати доступ до внутрішніх документів парламенту, які визначають процедури контролю, серед яких Регламент Верховної Ради України (визначає порядок розгляду законопроектів, проведення слухань, подання запитів), протоколи засідань комітетів (містять обговорення, висновки, рекомендації), звіти комітетів (результати аналізу діяльності уряду), депутатські запити та відповіді на них (джерело інформації про порушення).

Інтеграція з цими документами дозволяє аналізувати відповідність процедур вимогам Регламенту, виявляти прогалини у виконанні резолюцій, будувати мережі взаємодії між депутатами, комітетами, урядом.

Як приклад розглянемо автоматичне оновлення модуля контролю за бюджетом після прийняття нового Закону про державний бюджет Щороку після прийняття Закону про державний бюджет необхідно оновити модулі контролю, які аналізують виконання бюджету.

За допомогою інтегрованої системи цей процес може бути автоматизований:

1. Система отримує сповіщення про опублікування нового Закону про бюджет.
2. Через API витягується текст закону, зокрема розділи щодо:
   - розподілу коштів між відомствами;
   - цільових програм;
   - строків виконання.
3. Модуль «Контроль за виконанням бюджету» автоматично оновлює свої параметри:
   - встановлює нові ліміти витрат;
   - визначає контрольні терміни;
   - формує умови для виявлення відхилень.

Якщо виявлено, що витрати за місяць перевищили 110% від планових – система генерує алерт для відповідного комітету.

Така інтеграція перетворює систему контролю з статичного інструменту в динамічну, адаптивну платформу, здатну реагувати на зміни в реальному часі.

Отже, інтеграція з правовими базами даних та внутрішньою документацією – це не технічна деталь, а фундаментальна умова працездатності інтелектуальної системи контролю. Вона забезпечує, що всі аналітичні

висновки, виконувані модулі та рекомендації ґрунтуються на актуальних, офіційних та підзвітних джерелах, що робить систему не лише інтелектуальною, а й юридично надійною.

2.7. Масштабування та інтеграція

Окремі виконувані модулі безкодового програмування – це потужний інструмент для аналізу окремих аспектів парламентського контролю. Однак їхня справжня сила розкривається лише тоді, коли вони об'єднуються в інтегровану систему контролю, здатну до комплексного аналізу, автономного функціонування та інтелектуальної координації. Цей перехід – від ізольованого модуля до інтегрованої системи є критичним етапом трансформації парламентського контролю в цифрову, інтелектуально підсилену сферу державного управління.

Принципи побудови інтегрованої системи контролю із окремих модулів

Будь-яка складна система будується на основі модульності – розділення на незалежні, але взаємопов'язані компоненти. У контексті парламентського контролю це означає:

– створення окремих модулів для різних завдань: аналіз законопроектів, моніторинг бюджету, контроль за дотриманням міжнародних зобов'язань, обробка скарг громадян;

– забезпечення їхньої узгодженості через спільні онтології, стандарти даних та механізми верифікації;

– можливість незалежного оновлення кожного модуля без порушення роботи всієї системи.

Наприклад, модуль контролю за податковою дисципліною може бути розроблений окремо від модуля контролю за державними закупівлями, але обидва вони повинні працювати в межах єдиної архітектури, використовуючи спільні механізми верифікації, інтеграції з базами законодавства та форматів виводу.

Така система дозволяє масштабувати контроль: додавати нові модулі, адаптувати існуючі, комбінувати їх для вирішення складних завдань.

Механізми оркестрації модулів

Для того щоб модулі працювали не ізольовано, а як частина єдиного процесу, необхідна оркестрація – механізм координації їхнього виконання. Це може бути реалізовано через:

– Черги завдань (task queues) – модулі виконуються в певному порядку, наприклад: спочатку аналіз законопроекту, потім – перевірка на відповідність міжнародним зобов'язанням, далі – оцінка фінансових наслідків.

– Тригери (triggers) – події, які запускають виконання модулів. Наприклад, опублікування нового закону може автоматично запустити модуль аналізу суперечностей.

– Умовне виконання (conditional execution) – модуль запускається лише за певних умов. Наприклад, модуль контролю за корупційними ризиками активується, якщо виявлено більше трьох скарг на одного чиновника.

– Ці механізми дозволяють створювати адаптивні робочі процеси, які реагують на зміни в реальному часі, не вимагаючи постійного ручного керування.

Сценарії застосування: комплексний аналіз законопроекту, моніторинг діяльності кабінету міністрів

Сценарій 1: Комплексний аналіз законопроекту

При надходженні нового законопроекту система автоматично запускає ланцюжок модулів:

1. Модуль аналізу тексту – виявляє основні норми, визначення, умови.

2. Модуль перевірки на суперечності – порівнює з чинним законодавством, використовуючи семантичну мережу.

3. Модуль фінансового аналізу – оцінює бюджетні наслідки.

4. Модуль міжнародного права – перевіряє відповідність ЄС, ООН, Всесвітньому банку.

5. Модуль соціального впливу – аналізує потенційний вплив на громадян, використовуючи дані зі скарг.

Результат – інтегрований звіт, який може бути наданий комітету для розгляду.

Сценарій 2: Моніторинг діяльності Кабінету Міністрів

Система постійно моніторить:

- звіти уряду;
- постанови КМУ;
- медійні повідомлення;
- скарги громадян.

При виявленні відхилень (наприклад, прострочення виконання рішення Ради міністрів) система:

- активує відповідний модуль аналізу;
- формує алерт для комітету;
- пропонує сценарії реакції.

Можливості інтеграції з цифровим парламентом

Для максимальної ефективності система повинна бути інтегрована в цифрову інфраструктуру Верховної Ради. Це може бути реалізовано через:

– API (Application Programming Interface) – дозволяє системі отримувати дані з офіційних джерел (zakon.rada.gov.ua, реєстри, внутрішні бази) та надсилати результати аналізу.

– Віджети – інтерактивні блоки на інформаційних панелях депутатів, які показують:

– статус аналізу законопроектів;

– кількість виявлених суперечностей;

– ризики за сферами.

– Алерти – автоматичні повідомлення про:

– виявлення конфліктів у законодавстві;

– прострочення звітів уряду;

– масові скарги на діяльність державних органів.

Така інтеграція перетворює систему з аналітичного інструменту в постійного інтелектуального асистента, який підтримує парламентарів у реальному часі.

Отже, перехід від окремого модуля до інтегрованої системи контролю – це не просто технічне масштабування, а якісний стрибок у рівні інтелектуального підсилення. Саме така система здатна забезпечити комплексний, швидкий, прозорий та підзвітний парламентський контроль у умовах цифрової трансформації держави.

Розділ 3. Семантичний нетворкінг і формалізація правових текстів

Парламентський контроль у XXI столітті стикається з викликами, які не можуть бути подолані традиційними методами. Обсяг інформації, що підлягає аналізу – від законопроектів і звітів уряду до масових скарг громадян і міжнародних зобов'язань – виходить за межі можливостей ручної обробки. У той же час, сама природа контролю вимагає не просто обліку урахування даних, а глибокого розуміння змісту, виявлення прихованих зв'язків, прогнозування наслідків та підтримки прийняття стратегічних рішень.

Саме цей розрив між масштабом інформації та глибиною аналізу може бути подоланий завдяки семантичному інжинірингу – методології перетворення природної мови правових текстів на структуровані, формалізовані, динамічні моделі знань. Цей підхід виходить за межі простої обробки тексту (NLP) і переходить до інженерії знань, де кожне слово, кожна норма, кожна подія стає частиною єдиної аналітичної системи.

Семантичний нетворкінг ґрунтується на ідеї, що зміст документа не вичерпується його текстом, а реалізується через мережу зв'язків між поняттями. Ця мережа може бути побудована, візуалізована, проаналізована та використана для моделювання політичних процесів. На цей час основним інструментом для цього стають великі мовні моделі, які дозволяють автоматизувати процес вилучення сутностей, встановлення зв'язків та побудови причинно-наслідкових моделей.

Сучасні технології генеративного штучного інтелекту мають величезний потенціал для вирішення завдань парламентського контролю, зокрема для побудови і аналізу семантичних мереж[101]. Вони дозволяють не лише розуміти текст, а й активно формувати структури знань, які можуть бути використані для сценарного аналізу, прогнозування та підтримки рішень.

Однак для того щоб ці структури стали надійними, відтворюваними та підзвітними, їх необхідно формалізувати. Саме це і є суть семантичного нетворкінгу – процесу, який поєднує інтелектуальну генерацію з інженерними методами (онтології, графи, формальні моделі).

Далі розглядається, як можна систематично перетворювати правові тексти на структуровані моделі знань, використовуючи двонаправлений пошук, промпт-кероване зондування, візуалізацію в Gephi та динамічну еволюцію

---

[101] Парламентський контроль із застосуванням генеративного штучного інтелекту : монографія. Ланде Д.В., Фурашев В.М. - Київ: ТОВ "Інжиніринг", 2023. - 202 с. ISBN 978-966-2344-82-0

мереж. Особливу увагу приділено формалізації процесу, щоб він не залишався на рівні інтуїтивного аналізу, а ставав відтворюваним, верифікованим та масштабованим інструментом парламентського контролю.

3.1. Перетворення правових текстів на структуровані моделі знань

Перехід від аналізу тексту до аналізу знань – це фундаментальний стрибок у рівні інтелектуального аналізу даних. Він полягає в тому, щоб вийти за межі лінійного читання документа і побудувати його семантичну модель – граф, який відображає внутрішню структуру змісту. Цей процес може бути розкладений на декілька етапів.

Від тексту до мережі понять

Перший крок – це ідентифікація ключових понять у документі. Це не просто слова, а сутності реального світу: *«податкова служба»*, *«бізнес»*, *«штраф»*, *«скарга»*, *«парламентський запит»*. Ці поняття можуть бути виявлені за допомогою засобів розпізнавання сутностей на основі LLM, які точно визначають і класифікують основні елементи домену.

Кожне таке поняття стає вузлом у графі знань. Наприклад, аналізуючи звіт Ради бізнес-омбудсмена, ми можемо виділити вузли: *«податкові перевірки»*, *«блокування накладних»*, *«правоохоронні органи»*, *«документація»*.

Формування семантичних пар

Другий крок – встановлення зв'язків між поняттями. Це робиться через виявлення семантичних пар – пар сутностей, які мають смисловий зв'язок. Наприклад:

- «податкові перевірки;бізнес»
- «блокування накладних;зменшення скарг»
- «скарги;парламентські запити»

Такі зв'язки можуть бути отримані, наприклад, за допомогою промпт-керованого зондування:

«Виділіть у тексті всі пари сутностей, пов'язані з податковими перевірками. Представте у форматі: "Суб'єкт; Об'єкт"»

Цей підхід дозволяє автоматизувати процес формування мережі.

Створення структурованих файлів

Виявлені пари конвертуються в структурований формат, придатний для подальшої обробки. Найпоширенішим є формат CSV (Comma-Separated Values), де кожен рядок містить Source (джерело), Target (ціль), Type (тип зв'язку, наприклад, «є причиною», «застосовує», «порушує»), наприклад:

- «Податкові перевірки;Бізнес;Є причиною»
- «Блокування накладних;Зменшення скарг;Призводить до»

Такий файл може бути імпортований в інструменти візуалізації, такі як Gephi або GraphViz.

Побудова каузальних мереж шляхом виявлення причинно-наслідкових зв'язків між подіями, нормами, станами

На основі семантичних пар будується причинно-наслідкова (каузальна) мережа, яка відображає, як одна подія призводить до іншої. Це критично важливо для сценарного аналізу. Наприклад:

«Збільшення кількості перевірок» → *«Зростання кількості скарг»* → *«Ініціювання парламентського розслідування»*

Як зазначено в розділі 5.4 попередньої монографії, такі мережі дозволяють моделювати вплив політичних рішень, прогнозувати наслідки та визначати основні точки впливу.

Використання формальних моделей предметної області для забезпечення узгодженості термінології

Для забезпечення узгодженості та прозорості семантичні мережі інтегруються з онтологіями – формальними моделями предметної області. Онтології дозволяють:

- стандартизувати термінологію (наприклад, чи є «податковий інспектор» підтипом «державного службовця»);
- визначати ієрархії понять;
- забезпечувати логічну цілісність мережі.

Наприклад, онтологія парламентського контролю може містити класи: *«законопроект»*, *«резолюція»*, *«комітет»*, *«скарга»*, та відношення між ними: *«розглядається»*, *«вимагає»*, *«порушує»*.

Інтеграція з онтологіями перетворює семантичну мережу з інформаційного засобу в аналітичну систему, здатну до автоматичного висновку, верифікації та підтримки рішень.

Таким чином, перетворення правових текстів на структуровані моделі знань – це не просто технічний процес, а нова парадигма аналізу, яка дозволяє переосмислити парламентський контроль як інтелектуально підсилену систему, здатну до глибокого розуміння, прогнозування та координації.

3.2. Сценарний аналіз на базі семантичних мереж

Однією з найважливіших функцій семантичного інжинірингу у сфері парламентського контролю є сценарний аналіз – процес моделювання можливих шляхів досягнення певної мети, оцінка їх наслідків, ризиків та витрат. Однак перешкодою для проведення такого аналізу є неспроможність створення повної та достовірної причинно-наслідкової мережі, що традиційно вимагає значних часових та експертних ресурсів. Для подолання цієї проблеми було розроблено метод двонаправленого пошуку, який

дозволяє автоматизовано, швидко та системно будувати складні каузальні моделі за допомогою генеративних моделей ШІ.

Цей підхід ґрунтується на ідеї, що ефективніше будувати мережу не від одного кінця до іншого, а одночасно з двох кінців – від початкового стану (проблеми) і від бажаного стану (мети). Коли дві часткові мережі розширюються назустріч одна одній, вони в певний момент «зустрічаються», утворюючи повний логічний ланцюжок. Цей метод отримав назву «динамічний семантичний нетворкінг»[102] і став основою для формування сценаріїв діяльності в реальному часі.

У цьому розділі розглядається подальше розвинення цього підходу – процедура генерації аналітичних сценаріїв діяльності на основі отриманих каузальних структур. Запропонована методологія поєднує інструменти інтелектуального аналізу тексту, технології побудови причинно-наслідкових зв'язків та методи впорядкування сюжетних ланцюжків шляхом аналізу графової моделі.

Ключова перевага такого підходу полягає в тому, що сформовані каузальні мережі забезпечують пряму основу для генерації, ранжирування та аналізу різноманітних сценаріїв розвитку подій. При цьому формування каузальних мереж здійснюється у режимі, близькому до реального часу, завдяки автоматизованому опитуванню LLM, що значно підвищує оперативність та адаптивність аналітичного процесу.

Далі наводиться прикладна процедура побудови каузальної мережі через послідовне використання промптів до системи LLM, яка дозволяє отримати необхідні сценарії дій. Описується методологія формування ланцюжків взаємопов'язаних концептів у межах семантичної каузальної мережі, які ведуть від заданого початкового стану до визначеної мети. Основною задачею є генерація множини можливих сценаріїв досягнення цілі з подальшим їх ранжируванням за різноманітними критеріями – такими як ефективність, ризики, тривалість, ресурсна забезпеченість тощо.

Першим кроком у цьому процесі є побудова самої каузальної мережі, яка реалізується засобами генеративного штучного інтелекту. Вихідними даними для подальшої генерації сценаріїв виступають:

- початковий стан – поточна ситуація або точка відліку;
- глобальна задача (першопричина) – фактор, що ініціює розвиток подій;
- мета (ціль) – бажаний кінцевий стан, досягнення якого аналізується.

Наприклад, у контексті задачі виходу нового мобільного оператора на ринок конкретного населеного пункту, початковим станом виступає поняття

[102] D. Lande, L. Strashnoy, O. Driamov. Implementation of Dynamic Networking Utilizing Generative AI Technologies. SSRN Preprint: 4547440, DOI: 10.2139/ssrn.4547440. (August 28, 2023). – 15 p.

«Мобільний оператор», а метою – «Захоплення ринку у певному населеному пункті». Такий формалізований опис дозволяє не лише чітко визначити рамки аналізу, але й ефективно генерувати сценарії стратегічного розвитку з подальшою їх оцінкою та вибором найбільш оптимального шляху реалізації.

Для ефективного розв'язання задачі формування аналітичних сценаріїв у рамках OSINT-досліджень пропонується використання методології, заснованої на алгоритмі двонаправленого пошуку103. Цей підхід передбачає одночасне формування причинно-наслідкової мережі з двох напрямків: з боку початкового стану (вихідної ситуації) та з боку кінцевої мети (бажаного результату). Такий спосіб забезпечує більш структурований і цілеспрямований процес генерації зв'язків між подіями.

Процес розпочинається з побудови часткових каузальних графів навколо обох ключових точок – початкового стану й кінцевої мети. Унаслідок природної обмеженості предметної області, на певному етапі ці часткові мережі перетнуться, утворюючи єдиний зв'язний граф. Після досягнення цього стану можна припинити подальше формування мережі або продовжити його для отримання додаткових шляхів, залежно від потреб конкретної аналітичної задачі.

Після завершення побудови повної каузальної мережі здійснюється другий етап – виявлення та ранжирування можливих сюжетних ланцюжків, які ведуть від початкового стану до мети. Для цього застосовуються стандартні алгоритми пошуку оптимальних шляхів у графі, з урахуванням заданих параметрів, таких як довжина шляху, кількість переходів чи кількість проміжних концептів.

На останньому етапі відбувається змістовне ранжирування виявлених сценаріїв. Отримані ланцюжки аналізуються за критеріями, важливими для аналітика, зокрема:

- бюджетні витрати;
- тривалість реалізації;
- ступінь ризику;
- ресурсна забезпеченість;
- правова та політична допустимість.

Цей процес також може бути автоматизований з використанням інструментів генеративного штучного інтелекту, що забезпечує не лише формальну, але й змістовну оцінку кожного сценарію.

У процесі побудови семантичної мережі, особливо на початкових етапах, використовуються різні джерела інформації, які забезпечують поповнення

[103] D. Lande, L. Strashnoy, O. Driamov. Implementation of Dynamic Networking Utilizing Generative AI Technologies. SSRN Preprint: 4547440 DOI: 10.2139/ssrn.4547440. (August 28, 2023). – 15 pp.

графа новими концептами та їх взаємозв'язками. Зокрема, виділяються три основні класи джерел:

– Внутрішній потенціал LLM – залучається безпосередньо через промпти, які дозволяють виявити поняття, що мають смисловий зв'язок з початковим станом або кінцевою метою. Приклад: використання моделей типу ChatGPT, Claude, Gemini тощо.

– Інтернет-ресурси – текстові матеріали, доступні через пошукові системи (Google, Bing, Baidu та ін.). Вони дають змогу врахувати фактичні дані, актуальні події, офіційні заяви, аналітичні огляди, що значно поглиблює контекст дослідження.

– Системи документального аналізу (інсайт) – включають законопроекти, звіти, аналітичні записки, внутрішні документи, які можуть бути доступні в парламентських архівах, державних закладах чи корпоративних системах. Ці джерела надають стратегічно важливу інформацію, особливо при дослідженні політичних, соціальних або економічних процесів.

Слід зазначити, що джерела другого та третього типів є необов'язковими на початкових етапах формування мережі, однак вони можуть бути активно залучені на наступних етапах для уточнення, валідації та поглиблення отриманих знань.

Таким чином, запропонований підхід забезпечує комплексне, багаторівневе моделювання причинно-наслідкових зв'язків, що є ключовим елементом аналітики в умовах відкритих джерел інформації (OSINT), особливо при прогнозуванні сценаріїв розвитку подій.

Саме формування семантичної каузальної мережі передбачає формування двох часткових мереж. Формування першої часткової мережі (від вихідного стану, глобальної задачі) передбачає наступні кроки:

1. Для системи LLM формується промпт, в якому пропонується декомпозувати «початковий стан», отримати «часткові початкові стани» (наприклад, у випадку аналізу діяльності мобільного оператора це може бути тарифікація мобільного оператора, покриття території, рівень інновацій, взаємодія з місцевою владою тощо., – все це має підказати LLM).

2. Після отримання «часткових початкових станів» їх можна відфільтрувати в залежності від потреб замовника шляхом діалогу, щоб він виключив непотрібні з його точки зору стани. За відсутності можливості діалогу, використовуючи підхід «рою віртуальних експертів», можна відібрати найбільш значущі стани. У новій мережі, що з'являється, вузли будуть поняттями (сутностями). В результаті виконання кроків 1 і 2 буде сформована первинна мережа з вузлом «початковий стан» і спрямованими зв'язками між «початковим станом» і «частковими початковими станами».

3. Цей крок необов'язковий, підвищує рівень достовірності мережі. За допомогою системи LLM вибираються ролі віртуальних експертів для

обраної предметної області. У якості таких ролей система може запропонувати, наприклад, ролі: власника бізнесу, партнера, технічного спеціаліста, фінансиста тощо. Якщо не виконувати цей крок, можна перейти до кроку 4 і просто використовувати загальну систему оцінки LLM з невизначеною роллю замовника.

4. На основі «часткових вихідних станів» (або останніх в ієрархії понять) треба запитати (шляхом введення промптів) у LLM, які «наслідки», що покращують загальну ситуацію, можуть випливати з наявності цих понять. У результаті система LLM надасть множину наслідків. Ці поняття з'єднуються зв'язками, спрямованими в їх бік від відповідних останніх в ієрархії понять.

5. Запитується у системи LLM, які нові «наслідки», що покращать загальний стан, можуть мати отримані на попередньому кроці «наслідки». При цьому кожний промпт до системи LLM може бути виконаний декілька разів від імені різних віртуальних експертів, щоб дублювати важливі підключення та зменшувати вагу неактуальних понять.

6. Потім здійснюється перехід до кроку 5. Процедура припиняється після того, як ця перша мережа (половина мережі) у поєднанні з другою половиною мережі (розглядається далі) не утворить зв'язну мережу. На практиці рекомендується продовжити формування мережі (перейти до кроку 5) ще одним або двома кроками, це визначить практика.

Паралельно будується друга половина семантичної мережі таким чином:

1. За допомогою системи LLM, як і раніше, вибираються ролі віртуальних експертів для обраної предметної області. Якщо не виконувати цей опціональний крок, то можна перейти до кроку 2 і просто використовувати загальну систему оцінки, що є в LLM.

2. У системи LLM від імені вибраних віртуальних експертів запитуються причини, які можуть привести до «цілі». Будується часткова мережа, в якій ведуться спрямовані ребра-зв'язки до «цілі» від нових виявлених понять – «причин».

3. Від імені вибраних віртуальних експертів запитуються причини для кожної причини, отриманої на кроці 2, що може привести до «цілі». У результаті система LLM надасть множину понять – «причин». Ці поняття з'єднуються ребрами, спрямованими від них у бік попередньо вибраних вузлів – причин.

4. Потім здійснюється циклічний до перехід кроку 3. Процедура припиняється після того, як ця перша часткова мережа (половина мережі) у поєднанні з другою половиною мережі (розглядається далі) не утворить зв'язну мережу. Процес завершується за тими ж правилами, що і у випадку формування першої часткової мережі.

Обидві отримані часткові мережі формуються як на основі ресурсів самої LLM, так і, по можливості, на основі даних з OSINT та інсайтів, з яких попередньо утворюється навчальний масив.

Після формування обох часткових мереж вони об'єднуються (рис. 3), тобто враховуються всі вузли-поняття і всі зв'язки. Слід зауважити, що можуть з'явитися дублікати вузлів та/або посилань. У цих випадках ваги відповідних ребер або вузлів мережі підсумовуються. Створена мережа буде зваженою, спрямованою та причинно-наслідковою з маршрутами від початкового стану до мети.

Потім вибираються всі маршрути від початкового стану до пункту призначення. При цьому розраховується вага цих маршрутів – це окрема задача, яка вирішується відомими методами оптимізації. Зауважимо, типова система LLM відмінно вирішує цю проблему, якщо в неї передається мережа у вигляді списку вузлів і посилань.

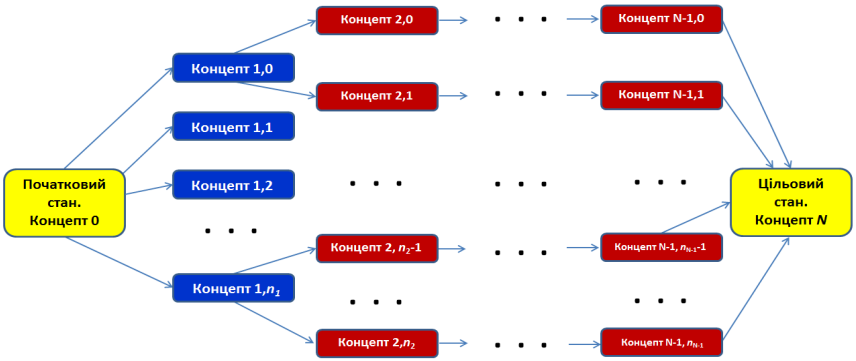

Рисунок 3: Вигляд об'єднаної причинно-наслідкової мережі

Після видаються найвагоміші за якимись критеріями маршрути у вигляді послідовності вузлів мережі – перелік причин і наслідків. Водночас можна отримати декілька сюжетних ланцюжків, впорядкованих за формальними ознаками.

Якщо користувач бажає впорядкувати отримані сюжетні лінії за вмістом, він також може ввести відповідні промпти до системи LLM, наприклад, такий: «Впорядкуйте отримані сюжетні лінії за часом реалізації сценарію».

Приклад

Як спрощений наочний приклад генерації сценаріїв запропонованим методом розглянемо проблему захоплення ринку мобільного зв'язку деяким мобільним оператором. Для цього за допомогою системою LLM ChatGPT формується причинно-наслідкова мережа, в якій початковим станом будемо вважати «мобільний оператор», а цільовим, тобто кінцевим – «захоплення ринку».

Початковий промпт забезпечує декомпозицію початкового стану:

Розкласти поняття "мобільний оператор". Виділити 10 основних сутностей (не більше трьох слів у понятті).

Система типу ChatGPT надає відповідь вигляду:

Послуги зв'язку
Бездротове з'єднання
Телефонна інфраструктура
Стільникові мережі
Передача даних
Абоненти зв'язку
Тарифні плани
Технічна підтримка
Послуги роумінгу
Підтримка інфраструктури

З використанням підходу "рою віртуальних експертів" запит повторюється декілька разів, в результаті чого отримуються декілька десятків відповідей, що іноді повторюються. Після цього емулюється клієнт шляхом формулювання промпту, який містить всі відповіді:

Виберіть 10 найважливіших із уже названих сутностей: Комунікаційні послуги; Бездротовий зв'язок; Телефонна інфраструктура; Стільникові мережі; Передача даних; Абоненти зв'язку; Тарифні плани; Технічна підтримка; Послуги роумінгу; Підтримка інфраструктури; Тарифи; Якість зв'язку; Інновації; Ресурси; Бренд…

Як результат, система надає відповідь, яка є першим рівнем першої часткової мережі:

Тарифи
Якість з'єднання
Інновації
Ресурси
Бренд
Роумінг
Послуги
Технічна інтеграція
Обмін даними
Доступ до клієнтської бази

Далі ми формуються промпти для кожного отриманого поняття. Зверніть увагу, що для спрощення, будемо розглядати надалі лише дві ролі майбутніх віртуальних експертів: загальну роль та роль партнера.

Які наслідки можуть бути від того, що мобільний оператор має хороші тарифи? Назвіть 10 наслідків – понять, не більше трьох слів кожне

Відповідь ChatGPT:

Залучення клієнтів Висока конкурентоспроможність Збільшення доходу Лояльні підписники Зростання підписників Зменшення відтоку Збільшення частки ринку Збільшення використання Популярність бренду Покращення репутації

Той самий запит від імені партнера:

Роль 2, поняття 1	Які наслідки можуть бути через те, що мобільний оператор має хороші тарифи з точки зору партнера? Назвіть 10 наслідків – поняття не більше трьох слів кожне.

Відповідь ChatGPT:

Збільшення кількості клієнтів Більше трафіку Партнерські угоди Додаткові послуги Збільшення прибутку Найкращі партнерства Розширення ринку Більше продажів Спільні акції Можливості для зростання

Наступний промпт стосується другого ключового поняття:

Роль 1, поняття 2	Які наслідки можуть бути через те, що мобільний оператор має високу якість зв'язку? Назвіть 10 наслідків – поняття не більше ніж з трьох слів кожне

Відповідь ChatGPT:

Задоволені клієнти Збільшення клієнтів Найкращий показник відтоку Підвищення репутації Збільшення доходу Висока оцінка Привабливість бренду Конкурентна перевага Покращення сервісу Зменшення скарг

Цей промпт стосується другого ключового поняття, що оцінюється від ролі партнера:

Роль 2, поняття 2	Які наслідки того, що мобільний оператор має високу якість зв'язку з точки зору партнера? Назвіть 10 наслідків – поняття не більше ніж по три слова кожне

Відповідь ChatGPT:

Збільшення клієнтів Збільшення кількості угод Позитивний імідж Задоволені клієнти Зростання доходів Привабливість партнерства Стабільність партнерства Збільшення лояльності Збільшення кількості запитів Успішна співпраця

Цей процес для всіх визначених концептів продовжується відповідно до ієрархічного принципу відповідно до заданого алгоритму.

Після цього формується друга часткова мережа, яка має завершитися станом "захоплення ринку". Звичайно, ця мережа формується з кінця. Для цього вводяться промпти відповідно до алгоритму:

Роль 1, мета	Перерахуйте 10 причин, чому ринок захоплює оператори мобільного зв'язку в невеликому місті. Кожна причина складається не більше ніж з трьох слів. Надайте відповідь у вигляді записів у форматі 'причина; захоплення ринку'. Кожен запис є окремим пунктом.

Відповідь ChatGPT:

Інновації; Захоплення ринку Гнучкі тарифи; Захоплення ринку Широке покриття; Захоплення ринку Якість обслуговування; Захоплення ринку Маркетингові стратегії; Захоплення ринку Лояльність абонентів; Захоплення ринку Унікальні послуги; Захоплення ринку Технічна підтримка; Захоплення ринку Партнерства; Захоплення ринку Співпраця з місцевими партнерами; Захоплення ринку

Далі цей промпт, що стосується мети, оцінюється від ролі партнера:

Роль 2, мета	Назвіть 10 причин, які можуть призвести до захоплення ринку мобільного зв'язку операторами в невеликому місті з точки зору партнера. Кожна причина складається не більше ніж з трьох слів. Дайте відповідь у вигляді записів у форматі "Причина; Захоплення ринку". Кожен запис – окремий пункт.

Відповідь ChatGPT:

Збільшення кількості клієнтів; Захоплення ринку Збільшення продажів; Захоплення ринку Успіх партнерства; Захоплення ринку Збільшення прибутку; Захоплення ринку Лояльність підписників; Захоплення ринку Бренд; Захоплення ринку Спільні акції; Захоплення ринку Нові можливості; Захоплення ринку Розвиток партнерства; Захоплення ринку Переваги для партнерів; Захоплення ринку

Далі для кожної з причин, згідно з алгоритмом, вводяться запити відповідного типу:

Роль 2, «Інновації»	Назвіть 10 причин, які можуть призвести до інновацій для захоплення ринку мобільних операторів у невеликому місті з точки зору партнера. Кожна причина складається не більше ніж з трьох слів. Дайте відповідь у вигляді записів у форматі "причина; Інновація". Кожен запис – це окрема стаття.

Після цього ChatGPT надає відповіді вигляду:

Збільшення кількості клієнтів; Інновації
Успіх партнерства; Інновації
Зростання прибутку; Інновації
Лояльність абонентів; Інновації
Технічна інтеграція; Інновації
Унікальні послуги; Інновації
Маркетингові стратегії; Інновації
Технічна підтримка; Інновації
Ефективні процеси; Інновації
Розвиток бренду; Інновації

Отже, відповідно до вищезазначеного алгоритму, утворюється друга часткова мережа, пов'язана з цільовим станом.

Об'єднуючи дві часткові мережі, визначені відповідями системи LLM, отримуємо загальну мережу, яка надалі виступає основою для формування сюжетних ланцюжків.

Формування сценаріїв

Для формування сценаріїв як множини сюжетних ланцюжків відповідно до побудованої моделі в ChatGPT введено такий промпт:

На основі побудованої мережі визначте кілька ланцюжків, що ведуть від вузла "мобільний оператор" до вузла "захоплення ринку". Посилання у квадратних дужках не перекладайте

Відповідь:

Ланцюжок 1:

- Mobile operator (Мобільний оператор)
- Quality of service (Якість сервісу)
- Subscriber loyalty (Лояльність абонентів)
- More subscribers (Збільшення кількості абонентів)
- Capturing the market (Захоплення ринку)

Ланцюжок 2:

- Mobile operator (Мобільний оператор)
- Marketing strategies (Маркетингові стратегії)
- Partnership relations (Партнерські відносини)
- Partnership success (Успіх у партнерстві)
- Capturing the market (Захоплення ринку)

Ланцюжок 3:

- Mobile operator (Мобільний оператор)

- Wide coverage (Широке покриття)
- More customers (Збільшення кількості користувачів)
- Capturing the market (Захоплення ринку)

Ланцюжок 4:

- Mobile operator (Мобільний оператор)
- Flexible rates (Гнучкі тарифи)
- Subscriber loyalty (Лояльність абонентів)
- More customers (Збільшення кількості користувачів)
- Capturing the market (Захоплення ринку)

Ланцюжок 5: (Мобільний оператор)

- Mobile operator (Мобільний оператор)
- Innovations (Інновації)
- Unique services (Унікальні послуги)
- Capturing the market (Захоплення ринку)

Звичайно, ці ланцюжки є лише прикладами, і існують інші способи досягнення цілі захоплення ринку за допомогою різних комбінацій цих вузлів. Для вибору найкращих сценаріїв здійснюється їх рейтингування.

Рейтингування сюжетних ланцюжків за значущими ознаками

Нижче наведено результати ранжування за допомогою системи LLM (ChatGPT) з отриманих сюжетних ланцюжків.

Для впорядкування сюжетних ланцюжків за фактором ціни до системи ChatGPT вводиться наступний запит:

Експертно оцініть наведені мережі з точки зору цінового фактора

Слід враховувати, що вартість впровадження кожного ланцюжка може значно варіюватися залежно від багатьох факторів, таких як конкуренція на ринку, технічні можливості, географічні обмеження та інші. Проте, на загальному рівні система ChatGPT робить наступні припущення щодо ранжування ланцюжків за ціновим чинником, від найвищої до найнижчої вартості:

Ланцюжок 2:

- Mobile operator (Мобільний оператор)
- Marketing strategies (Маркетингові стратегії)
- Partnership relations (Партнерські відносини)
- Partnership success (Успіх у партнерстві)
- Capturing the market (Захоплення ринку)

Ланцюжок 5:

- Mobile operator (Мобільний оператор)

- Innovations (Інновації)
- Unique services (Унікальні послуги)
- Capturing the market (Захоплення ринку)

Ланцюжок 1:

- Mobile operator (Мобільний оператор)
- Quality of service (Якість сервісу)
- Subscriber loyalty (Лояльність абонентів)
- More subscribers (Збільшення кількості абонентів)
- Capturing the market (Захоплення ринку)

Ланцюжок 3:

- Mobile operator (Мобільний оператор)
- Wide coverage (Широке покриття)
- More customers (Збільшення кількості користувачів)
- Capturing the market (Захоплення ринку)

Ланцюжок 4:

- Mobile operator (Мобільний оператор)
- Flexible rates (Гнучкі тарифи)
- Subscriber loyalty (Лояльність абонентів)
- More customers (Збільшення кількості користувачів)
- Capturing the market (Захоплення ринку)

Це припущення ґрунтується на тому, що створення й впровадження ефективних маркетингових стратегій, партнерств і успіху в партнерстві може потребувати значних інвестицій. Інновації та унікальні послуги можуть потребувати значних витрат на дослідження та розвиток. Хоча поліпшення якості обслуговування, лояльність абонентів та зростання клієнтів можуть бути менш цінними, вони також можуть вимагати інвестицій.

Для впорядкування сюжетних ланцюжків, наприклад, за фактором часу до системи ChatGPT вводиться запит вигляду:

Експертно оцініть наведені мережі з точки зору фактора часу

Система LLM за промптом, на основі припущення, які ланцюжки можуть бути реалізовані з меншими витратами часу порівняно з іншими, створює цей рейтинг ланцюгів за часовим фактором, від найшвидшого до найтривалішого.

Така система LLM, як ChatGPT, виробляє оцінки сюжетних ланцюжків виходячи із загальних уявлень щодо фінансової складової, часу впровадження та ризиків на основі досвіду, отриманого системою в результаті машинного навчання.

Незважаючи на вирішення проблеми ресурсів (як часових, так і людських), важливо зазначити, що як сам процес побудови причинно-наслідкових

мереж, так і формування сюжетних ланцюжків, їх впорядкування, візуалізація та інтерпретація результатів вимагають від аналітика даних певного досвіду в предметній галузі, що вивчається, і як і раніше, необхідно спостереження з боку людини для забезпечення достовірності та точності результатів.

Пошук від проблеми

Перший напрямок – це аналіз вихідного стану, який визначається як глобальна проблема, що потребує контролю. Наприклад: *«велика кількість скарг на податкові перевірки»*, *«невиконання урядом міжнародних зобов'язань»*, *«суперечності в законодавстві»*.

За допомогою LLM (наприклад, ChatGPT) формується промпт для декомпозиції проблеми на часткові причини:

«Розкладіть поняття "велика кількість скарг на податкові перевірки" на 10 часткових причин. Представте у форматі: "Часткова причина; Вихідна проблема"»

Система генерує список, наприклад:

- *«Недостатня прозорість процедур;велика кількість скарг»*
- *«Надмірна кількість перевірок;велика кількість скарг»*
- *«Відсутність механізму оскарження;велика кількість скарг»*

Для кожної з цих часткових причин можна повторити запит, отримуючи ієрархічну структуру причин, що формує першу частину семантичної мережі – мережу проблеми.

Пошук від мети

Другий напрямок – це аналіз бажаного стану, тобто цілі, яку потрібно досягти. Наприклад: *«зменшення кількості скарг»*, *«виконання міжнародних зобов'язань»*, *«відсутність суперечностей у законодавстві»*.

Формується відповідний промпт:

«Які чинники можуть призвести до зменшення кількості скарг на податкові перевірки? Перелічіть 10 чинників. Представте у форматі: "Чинник; Бажаний стан"»

Система пропонує, наприклад таке:

- *«Запровадження електронного оскарження»; «зменшення скарг»*
- *«Обмеження кількості планових перевірок»; «зменшення скарг»*
- *«Підвищення кваліфікації інспекторів»; «зменшення скарг»*

Ці чинники також можуть бути деталізовані, утворюючи мережу мети – шляхи досягнення бажаного стану.

Об'єднання мереж: утворення повного причинно-наслідкового ланцюжка

Коли обидві часткові мережі – від проблеми та від мети – достатньо розширені, вони об'єднуються в єдину структуру. Цей процес може бути виконаний автоматично:

- всі вузли (поняття) та зв'язки (відносини) з обох мереж імпортуються в єдиний простір;
- дублікати вузлів об'єднуються, ваги зв'язків підсумовуються;
- система виявляє точки зустрічі – спільні поняття, які з'являються в обох мережах (наприклад, *«оскарження рішень»*, *«контроль за кількістю перевірок»*).

Ці точки стають важливими ланками у повному причинно-наслідковому ланцюжку, який тепер з'єднує початковий стан з кінцевим.

Як зазначено в попередній монографії, цей підхід схожий на двонаправлений пошук у теорії графів, де рух від початку і від кінця скорочує загальну кількість аналізованих вузлів і прискорює знаходження оптимального шляху.

Формування сценаріїв: на основі об'єднаної мережі

На основі повної мережі можна сформувати кілька сценаріїв досягнення мети, кожен з яких представляє собою шлях у графі від вихідного стану до цільового, наприклад:

- Сценарій 1: *Зменшення кількості перевірок → Зниження тиску на бізнес → Зменшення скарг*
- Сценарій 2: *Запровадження електронного оскарження → Підвищення правової захищеності → Зменшення скарг*
- Сценарій 3: *Підвищення кваліфікації інспекторів → Зменшення помилок → Зменшення скарг*

Кожен сценарій може бути оцінений за такими критеріями, як час реалізації, вартість, політична ризиковість, очікуваний ефект.

Це дозволяє парламентарям обирати найефективніший шлях для втручання.

Застосування в парламентському контролі

Метод двонаправленого пошуку вже успішно застосовується в практиці парламентського контролю. Наприклад:

– Аналіз несуперечності документів уряду: початковий стан – *«парламентський контроль»*, мета – *«несуперечність документів»*. Мережа виявила, що основним чинником є відсутність єдиного реєстру нормативних актів.

– Контроль за реформами: виявлено, що для досягнення мети *«ефективна реформа податкової системи»* необхідно одночасно вирішувати проблеми *«корупція в інспекціях»*, *«відсутність цифрових інструментів»* та *«низька правова культура»*.

– Аналіз діяльності мобільних операторів (як модельний приклад): визначено, що для підвищення якості послуг необхідно впливати на *«тарифікацію»*, *«покриття»*, *«інноваційність»*.

Отримані мережі візуалізуються в Gephi, що дозволяє аналітикам і депутатам наочно бачити структуру проблеми, основні точки впливу та альтернативні шляхи рішення.

Таким чином, двонаправлений пошук – це не просто методологія, а нова форма інтелектуального аналізу, яка дозволяє перейти від реактивного контролю до проактивного моделювання політичних процесів. Вона перетворює парламентський контроль з механізму фіксації помилок на інструмент стратегічного управління, здатний прогнозувати, планувати та оптимізувати державну політику.

3.3. Методи виявлення сутностей і зв'язків: контекстні пари, промпт-кероване зондування

Одним із важливих етапів семантичного інжинірингу є автоматизоване екстрагування знань – процес виявлення з текстів ключових понять (сутностей: об'єктів, понять, подій, явищ) та смислових зв'язків між ними. У контексті парламентського контролю це означає перехід від аналізу окремих слів до побудови мереж знань, де кожен вузол – це сутність (наприклад, *«податкова служба»*, *«скарга»*, *«штраф»*), а кожне ребро – відношення між ними (наприклад, *«застосовує»*, *«порушує»*, *«призводить до»*). Цей процес здійснюється за допомогою комбінації контекстних пар, промпт-керованого зондування, ієрархічного уточнення та інтеграції з зовнішніми джерелами, що дозволяє створювати структуровані, формалізовані та відтворювані моделі.

Контекстні пари: використання промптів для отримання пар сутностей, які мають смисловий зв'язок

Центральним елементом вилучення знань є формування контекстних пар сутностей – пар понять, які мають смисловий, частіше за все причинно-наслідковий, зв'язок у межах певного документа чи тематичної сфери. Наприклад, у звіті Ради бізнес-омбудсмена можна виявити пари на кшталт *«податкові перевірки – бізнес»*, *«блокування накладних – зменшення скарг»*, *«скарги – парламентські запити»*.

Цей процес реалізується через спрямовані промпти, які вказують LLM на необхідність виявлення саме пар, а не окремих термінів. Наприклад:

«Виділіть у тексті всі пари сутностей, пов'язані з податковими перевірками. Представте у форматі: "Суб'єкт; Об'єкт"»

Такий запит спонукає модель аналізувати текст не як потік слів, а як мережу відносин, що дозволяє автоматично генерувати структуровані дані для подальшої обробки. Важливо, що пари формуються з використанням

обмеженої кількості слів (наприклад, не більше трьох), що забезпечує стандартизацію термінології та уникнення надмірності.

Промпт-кероване зондування: стратегія, при якій LLM виступає як інструмент аналізу

Промпт-кероване зондування – це не просто генерація тексту, а систематичне використання LLM як інструменту аналізу, де кожен промпт є частиною інженерного процесу формування знань. Ця стратегія ґрунтується на тому, що LLM може бути налаштований на виконання конкретних завдань: виявлення причин, класифікація явищ, побудова ієрархій.

Наприклад, для аналізу причин суперечностей у законодавстві можна використовувати промпт:

«Перелічіть 10 причин неоднозначності, що веде до суперечності визначень у законодавчих документах. Представте у форматі: "причина; Неоднозначність"»

Відповідь системи може мати такий вигляд:

1. Двозначні терміни; Неоднозначність

2. Множинні тлумачення; Неоднозначність

3. Відсутність узгодження; Неоднозначність

...

Такий підхід перетворює LLM з генератора тексту на інструмент структурованого дослідження, де кожен вивід може бути зафіксований, перевірений та інтегрований в аналітичну систему.

Ієрархічне уточнення

Отримані на першому рівні причини чи сутності можуть бути далі уточнені на наступних рівнях аналізу, що дозволяє побудувати ієрархічну структуру знань. Наприклад, для кожної з 10 виявлених причин неоднозначності можна задати додатковий промпт:

«Перелічіть 5 причин для кожної з виявлених причин неоднозначності»

Це дозволяє побудувати дерево причин, де кожен верхній рівень розкривається на більш детальні підрівні. Така ієрархія є основою для причинно-наслідкового моделювання та сценарного аналізу, оскільки дозволяє відстежувати ланцюжки впливу від загальних явищ до конкретних подій.

Формалізація виводу

Для того щоб вивід LLM міг бути використаний як частина аналітичної системи, він повинен бути формалізований – перетворений на структурований, машинно-читабельний формат. Це досягається шляхом:

- використання чіткого формату виводу (наприклад, суб’єкт; об’єкт, причина; наслідок);
- збереження даних у табличному вигляді (CSV, JSON);
- використання уніфікованих термінів (наприклад, «штраф» замість «санкція», «скарга» замість «звернення»).

Формалізація забезпечує відтворюваність, порівняння та інтеграцію з іншими інструментами, такими як Gephi або GraphViz.

Інтеграція з базами законодавства для підвищення точності вилучення

Хоча LLM мають широкі знання, вони обмежені періодом навчання та можуть генерувати галюцинації або використовувати застарілі формулювання. Для підвищення точності вилучення знань використовується RAG (Retrieval-Augmented Generation)[104] – технологія, яка дозволяє доповнювати промпти актуальними даними з зовнішніх джерел.

Наприклад, перед аналізом законопроекту система автоматично:

- звертається до бази законодавства (наприклад, zakon.rada.gov.ua або «Ліга:Закон»);
- витягує текст відповідних норм, прецедентів, пояснень;
- додає ці тексти до промпту як контекст.

Таким чином, LLM аналізує не на основі внутрішніх знань, а на основі офіційних, актуальних джерел, що робить вивід надійним і підзвітним.

У результаті методи вилучення сутностей і зв’язків, засновані на промпт-керованому зондуванні та інтеграції з RAG, дозволяють перетворити величезні масиви правових текстів на структуровані, формалізовані, динамічні моделі знань, які стають основою для інтелектуального парламентського контролю. Це не просто аналіз – це інженерія знань, здатна забезпечити глибоке розуміння, прогнозування та підтримку стратегічних рішень.

3.4. Динамічна еволюція мережі. Виявлення прогалин, конфліктів, нових тем

Семантична мережа, створена на основі аналізу правових текстів, не є статичною структурою. Вона розвивається в часі, адаптуючись до змін у законодавстві, суспільних процесах, діяльності державних органів та політичній агенді. Ця динамічна еволюція перетворює мережу з інструменту одноразового аналізу на живу, адаптивну систему моніторингу, здатну виявляти нові явища, відстежувати трансформацію норм і передбачати

[104] Fan, W., Ding, Y., Ning, L., Wang, S., Li, H., Yin, D., Chua, T.S. and Li, Q., 2024, August. A survey on rag meeting llms: Towards retrieval-augmented large language models. In *Proceedings of the 30th ACM SIGKDD conference on knowledge discovery and data mining* (pp. 6491-6501).

потенційні ризики. У цьому пункті розглянуто, як семантична мережа оновлюється, як на її основі виявляються прогалини, конфлікти та нові теми, а також як можна порівнювати різні версії мереж для оцінки змін.

Постійне оновлення

Для того щоб семантична мережа залишалася актуальною, вона повинна регулярно оновлюватися новими даними. Це можуть бути:

- нові законопроекти та прийняті закони;
- звіти уряду, рішення Кабінету Міністрів;
- скарги громадян, депутатські запити;
- міжнародні угоди, рішення судів.

Кожен новий документ аналізується за допомогою промпт-керованого зондування, з нього витягуються нові сутності та зв'язки, які додаються до існуючої мережі. Цей процес може бути автоматизований через інтеграцію з базами законодавства (наприклад, zakon.rada.gov.ua) та внутрішніми системами парламенту. Таким чином, мережа постійно розширюється, формуючи динамічну модель знань, яка відображає реальний стан правової системи.

Виявлення прогалин

Однією з найважливіших функцій динамічної мережі є виявлення прогалин – ситуацій, коли певні поняття або зв'язки відсутні в мережі, хоча мають бути присутні, наприклад:

- у мережі є багато зв'язків, пов'язаних із податковими перевірками, але відсутній механізм оскарження;
- аналіз законопроектів показує, що не визначено порядок реалізації нової норми;
- у звітах уряду відсутні дані про виконання конкретних рішень.

Такі прогалини вказують на недоліки в законодавстві, виконавчій діяльності чи інформаційному забезпеченні. Їх можна виявити шляхом порівняння з еталонними онтологіями (наприклад, якщо в онтології є клас «оскарження», а в мережі – ні), аналізу центральності: якщо певне поняття має високу очікувану значимість, але мало зв'язків – це ознака прогалини.

Виявлення конфліктів

Динамічна мережа дозволяє виявляти конфлікти між нормами, які можуть бути прихованими або виникати внаслідок змін у законодавстві, наприклад:

- одна норма дозволяє певну діяльність, інша – забороняє;
- два закони встановлюють різні терміни для подання звітів;
- новий закон суперечить міжнародним зобов'язанням.

Конфлікти можуть бути виявлені шляхом:

- побудови причинно-наслідкових ланцюжків, де одна гілка призводить до дозволу, інша – до заборони;
- аналізу взаємовиключних зв'язків у мережі;
- інтеграції з онтологіями, де визначені правила узгодженості.

Наприклад, у мережі, побудованій на основі аналізу законопроектів, може бути виявлено, що норма про амністію для МСП суперечить нормі про штрафи за податкові порушення.

Виявлення нових тем

Мережа може виявляти нові теми, які з'являються в політичній агенді, наприклад:

- з'явилися поняття *«генеративний ШІ»*, *«цифровий двійник»*, *«блокчейн у держуправлінні»*;
- збільшилася кількість зв'язків між *«податковими перевірками»* та *«фейковими звітами»*;
- виникли нові типи скарг, пов'язані з цифровими послугами.

Це дозволяє парламентарям вчасно реагувати на нові виклики, ініціювати аналіз, проводити слухання, готувати законопроекти.

Порівняння мереж

Для оцінки змін у часі можна порівнювати різні версії мережі, наприклад:

- порівняти мережу до та після прийняття нового закону;
- проаналізувати, як змінилася структура зв'язків між скаргами та державними органами за рік;
- виявити, які теми втратили, а які – набули значення.

Це може бути зроблено шляхом:

- обчислення відстані між мережами (наприклад, за нормою Фробеніуса);
- аналізу зміни центральності вузлів;
- виявлення нових кластерів або розпаду існуючих.

Такий аналіз дозволяє кількісно оцінити рівень новизни, стабільність та напрямки розвитку правової системи.

3.5. Візуалізація та інтерпретація: Gephi, GraphViz, власні інструменти

Візуалізація є важливим етапом семантичного інжинірингу – вона перетворює абстрактні дані в наочну, інтерпретабельну картину, яка допомагає аналітикам і депутатам розуміти структуру знань, виявляти

патерни, приймати рішення. У цьому пункті розглянуто три основні групи інструментів: Gephi – для інтерактивного аналізу, GraphViz – для створення статичних діаграм, та власні інструменти, розроблені авторами для спеціалізованих завдань.

Gephi – основний інструмент візуалізації

Gephi (https://gephi.org) – це найпопулярніша безкоштовна платформа для візуалізації та аналізу мереж. Вона використовується авторами для демонстрації прикладів у цій монографії. Gephi дозволяє:

- імпортувати дані у форматах CSV, GML, GraphML, GEXF, DOT тощо. Особливо зручним є формат CSV, який легко генерується за допомогою ChatGPT;
- автоматично розташовувати вузли за допомогою потужних алгоритмів компонування: *ForceAtlas2*, *Yifan-Hu*, *Radial Layout*;
- аналізувати мережу: обчислювати центральність, щільність, модульність, найкоротші шляхи;
- виявляти кластери – групи тісно пов'язаних вузлів (наприклад, групу скарг, пов'язаних із податковою службою);
- інтерактивно досліджувати мережу: масштабувати, фільтрувати, навігувати;
- експортувати результати у форматах PNG, PDF, SVG.

Після завантаження даних у Gephi можна вибрати розмір вузлів пропорційно їх ступеню (кількості зв'язків) і розбити мережу на кластери за критерієм модульності, отримуючи чіткий граф, який відображає структуру проблеми.

GraphViz – інструмент для створення статичних діаграм

GraphViz – це набір інструментів для візуалізації структурованих даних, розроблений лабораторією AT&T. Він працює на основі мови опису графів DOT і має такі переваги:

- автоматичне укладання графів: інструмент *dot* створює ієрархічні графи, *neato* – на основі «пружинної» моделі, *circo* – кругові, *twopi* – радіальні;
- мінімізація перетинів: графи формуються так, щоб зв'язки не перетиналися, що забезпечує високу читабельність;
- підтримка форматів: PNG, JPG, SVG;
- інтеграція з мовами програмування (Python, Java, Perl), що дозволяє автоматизувати процес.

GraphViz особливо ефективний для створення статичних діаграм – дерев рішень, графів станів, причинно-наслідкових моделей, які можуть бути включенні до аналітичних звітів.

Власні інструменти

Візуалізація – це не просто оформлення результатів, а важливий етап інтерпретації, який перетворює складні мережі знань на інструменти стратегічного управління. Інтеграція Gephi, GraphViz та власних розробок створює універсальну платформу для інтелектуального парламентського контролю.

Для вирішення специфічних завдань авторами були розроблені власні інструменти:

CSV2Graph[105] – сервіс для візуалізації та аналізу мереж, які задаються у форматі CSV, із інтерфейсом за адресою http://bigsearch.space/uli.html.

LegalGraph[106] – сервіс для візуалізації відповідей LLM для побудови семантичних мереж у сфері права із інтерфейсом за адресою http://bigsearch.space/legal.html.

SemanticCore[107] – комп'ютерна програма для аналізу семантичних зв'язків (свідоцтво №125039, 2024).

Ці інструменти дозволяють автоматизувати процес від генерації тексту до візуалізації, роблячи семантичний інжиніринг доступним і масштабованим.

[105] Dimitri Busch, Dmytro Lande. Semantische Dokumentenindexierung mit generativer KI. Die Tagungsreihe "Digitale Bibliothek". Zurueck (und) in die Zukunft. 28-29 November 2024, Meerscheinschloessl, Universitaet Graz. Praesentationen.

[106] Ланде Д.В., Фегер А.П., Буш Д., Страшной Л. Семантичне індексування документів в галузі парламентського контролю. Проблеми інформаційного забезпечення та розвитку парламентського контролю в контексті Європейської та Євроатлантичної інтеграції України : матеріали наук.-практ. конф. (Київ, 25 квіт. 2024 р.) - Київ; Одеса : Фенікс, 2024. - С. 71-75.

[107] Комп'ютерна програма аналізу семантичних зв'язків "SemanticCore" Ланде Д.В., Фегер А.П. Україна. Свідоцтво про реєстрацію авторського права на твір N 125039 від 24.04.2024

Розділ 4. Віртуальні експерти та агентська логіка

Сучасні технології генеративного штучного інтелекту, засновані на великих мовних моделях, демонструють вражаючу здатність до аналізу, узагальнення та інтерпретації природної мови. Однак їхня стандартна реалізація як універсальних асистентів має істотні обмеження у контексті парламентського контролю. Відповіді, отримані від таких систем, часто носять узагальнений характер і не враховують специфічних вимог різних аналітичних завдань – будь то правова експертиза, фінансовий аналіз або оцінка виконання державних програм. Для подолання цих обмежень необхідний перехід від універсального інструменту до спеціалізованої системи інтелектуального контролю, побудованої на принципах віртуальних експертів та агентської логіки.

Концепція віртуальних експертів передбачає створення спеціалізованих інтелектуальних агентів, кожен з яких імітує професійну діяльність фахівця певного профілю – юриста, фінансиста, адміністратора, економіста тощо. Ці агенти не просто генерують текст, а виконують чітко визначені ролі, мають формалізовані повноваження та профілі поведінки, що активуються через промпт-кероване програмування. Вони можуть працювати ізольовано або взаємодіяти в рамках рою віртуальних експертів, забезпечуючи колективний аналіз, узагальнення даних та формування більш повних і збалансованих висновків.

Цей підхід дозволяє трансформувати ГШІ з пасивного інструменту відповідей на запити в активну, структуровану систему підтримки рішень, здатну до проактивного моніторингу, виявлення конфліктів, моделювання сценаріїв та формування рекомендацій. У цьому розділі розглядається, як можна формалізувати ролі віртуальних експертів, моделювати їхню поведінку, забезпечувати внутрішній стан та організовувати їхню взаємодію. Особлива увага приділяється обмеженням поточних технологій, які не дозволяють ще досягти масштабованої оркестрації агентів, але вказують на перспективи розвитку інтелектуальної системи контролю.

4.1. Від запиту до віртуального експерта

Застосування великих мовних моделей у сфері парламентського контролю стикається з фундаментальною проблемою – універсальність відповідей. Стандартні інтерфейси ГШІ, такі як ChatGPT, надають відповіді з позиції узагальненого «асистента», який намагається відповісти на запит з максимально можливою нейтральністю та широтою охоплення. Однак для ефективного парламентського контролю потрібні не універсальні, а спеціалізовані аналітичні погляди, засновані на глибокому розумінні певної предметної області: юриспруденції, фінансів, адміністративного права, кібербезпеки тощо. Тому необхідно вийти за межі універсального запиту і перейти до формалізації ролей віртуальних експертів, кожен з яких виконує певну функцію в аналітичному процесі.

Проблема універсальності LLM

LLM навчені на величезних масивах текстів із різних джерел, що робить їх універсальними, але одночасно – неспеціалізованими. Відповідь на запит *«Чи суперечить цей законопроект Конституції?»* може бути логічною, але не враховувати глибинних правових аргументів, практики Конституційного Суду чи специфіки міжгалузевих колізій. Така відповідь не може вважатися підставою для прийняття рішень у парламенті.

Щоб забезпечити високу якість аналізу, потрібно обмежити перспективу моделі, активувавши в ній певну професійну роль.

Необхідність спеціалізації

Ефективний парламентський контроль вимагає багаторівневого аналізу, який може бути виконаний лише за умови розділення функцій між різними типами експертів:

- юридичний аналіз (відповідність норм, конституційність);
- фінансовий аналіз (виконання бюджету, ефективність витрат);
- адміністративний аналіз (дотримання процедур, виконання рішень);
- соціальний аналіз (вплив на громадян, зворотний зв'язок).

Тому необхідно створювати віртуальних експертів з чітко визначеними профілями, які можуть бути викликані для вирішення конкретного завдання.

Формалізація ролей

Для забезпечення структурованості та відтворюваності аналізу пропонується формалізувати три базові ролі віртуальних експертів:

- Аналітик: цей агент фокусується на виявленні сутностей, зв'язків та причинно-наслідкових ланцюжків. Його завдання – побудова семантичної мережі, виявлення основних тем, формування ієрархій понять. Наприклад, аналітик може виявити, що зростання кількості скарг на податкові перевірки пов'язане зі збільшенням кількості планових перевірок.
- Контролер: цей агент займається перевіркою відповідності, виявленням порушень, аналізом виконання рішень та резолюцій. Він порівнює фактичний стан із нормативним, визначає відхилення та формує рекомендації. Наприклад, контролер може виявити, що уряд не подав звіт у встановлений строк, що є порушенням.
- Інтерпретатор норм: цей агент спеціалізується на тлумаченні законодавства, визначенні значення термінів, розкритті сенсу норм, виявленні суперечностей у визначеннях. Він може відповідати на запити типу *«Як має тлумачитися поняття "штучний інтелект" у цьому законі?»* або *«Чи є ця норма конституційною?»*

Використання промптів для активації ролі

Формалізація ролі здійснюється через промпт-кероване програмування – використання чітко сформульованих запитів, які активують певну професійну позицію, наприклад:

«Від імені експерта з податкового контролю проаналізуйте цей звіт і виявіть порушення»

або:

«Як юридичний інтерпретатор, поясніть, чи суперечить ця норма статті 19 Конституції України»

Такий промпт обмежує перспективу моделі, змушуючи її діяти як фахівець певного профілю, а не як універсальний асистент. Це забезпечує цілеспрямований, глибший та більш достовірний аналіз.

Перехід від «асистента» до «фахівця»

Цей підхід дозволяє здійснити концептуальний перехід від використання ГШІ як «асистента», який відповідає на запити, до використання як віртуального експерта, який виконує професійну аналітичну функцію. Кожен запит тепер спрямовується не в «нікуди», а до конкретного віртуального експерта з визначеною роллю, що забезпечує:

- підвищену глибину аналізу;
- фокус на конкретному аспекті контролю;
- можливість інтеграції відповідей різних експертів в єдину аналітичну систему.

Як показано в дослідженні[108], такий підхід значно спрощує і прискорює процес формування причинно-наслідкових мереж, оскільки кожен віртуальний експерт працює в межах своєї компетенції, а їхні висновки можуть бути об’єднані в єдину структуру.

Таким чином, формалізація ролей віртуальних експертів – це важливий крок до створення інтелектуальної системи парламентського контролю, здатної до спеціалізованого, проактивного та інтегрованого аналізу. Вона перетворює ГШІ з інструменту генерації тексту на систему віртуальних фахівців, кожен з яких вносить свій внесок у забезпечення підзвітності влади.

---

[108] Д.В. Ланде, В.М. Фурашев. Парламентський контроль із застосуванням генеративного штучного інтелекту : монографія / Ланде Д.В., Фурашев В.М. - Київ: ТОВ "Інжиніринг", 2023. - 202 с. ISBN 978-966-2344-82-0

4.2. Промпт-кероване програмування: створення спеціалізованих агентів

Розвиток генеративних мовних моделей відкрив можливість не просто генерувати текст на запит, а створювати спеціалізовані інтелектуальні агенти, здатні виконувати конкретні аналітичні завдання в межах парламентського контролю. Цей підхід, відомий як промпт-кероване програмування (prompt-driven programming), полягає в тому, що замість написання коду для визначення логіки роботи системи, її поведінку формується через структуровані промпти, які активують певні ролі, повноваження та методи аналізу. У результаті віртуальний експерт перестає бути лише джерелом текстового виводу і перетворюється на виконуваний модуль – самостійний компонент інтелектуальної системи контролю.

Агент як виконуваний модуль

У традиційному розумінні LLM виступає як «чорний ящик», який на вхід отримує запит, а на виході – текст. Однак для парламентського контролю необхідно більше: відтворюваність, структурованість та інтеграція з іншими системами. Тому віртуальний експерт розглядається не як асистент, а як агент – сутність з чітко визначеною логікою, цілями та повноваженнями. Цей агент може бути:

- запущений автоматично при виникненні певної події (наприклад, опублікування законопроекту);
- інтегрований в ланцюжок аналізу разом з іншими агентами;
- використаний для формування виконуваних модулів, які можуть бути перевірені на логічну цілісність.

Таким чином, агент стає виконуваним модулем у системі безкодового програмування, який може бути частиною більшої інженерної структури.

Промпт-кероване програмування

Промпт-кероване програмування – це метод створення агентів шляхом використання чітко сформульованих промптів, які визначають їхню роль, методологію аналізу та обмеження. Цей підхід дозволяє "програмувати" поведінку агента мовою природного тексту, без необхідності знання синтаксису програмування.

Використання промптів для визначення повноважень, обмежень, методів аналізу агента

Промпт формується так, щоб агент діяв з позиції певного фахівця, використовуючи відповідну термінологію та методи. Наприклад:

«Як експерт з кібербезпеки, проаналізуйте цей законопроект і виявіть потенційні загрози для державних інформаційних систем. Врахуйте такі аспекти: визначення критичної інфраструктури, механізми реагування на інциденти, відповідальність за витоки даних. Представте відповідь у форматі: "Загроза; Опис; Рекомендація"»

Такий запит:

- визначає роль агента (експерт з кібербезпеки);
- задає обмеження (аналіз лише кібербезпекових аспектів);
- вказує метод аналізу (перевірка трьох конкретних напрямків);
- вимагає структурованого виводу.

Це і є суть промпт-керованого програмування – замість відповіді у вигляді абзацу, система генерує аналітичний модуль, придатний для подальшої обробки.

Створення бібліотеки агентів

На основі цього підходу може бути створена бібліотека віртуальних експертів, кожен з яких спеціалізується на певному аспекті контролю. Приклади таких агентів:

– Агент-юрист: аналізує відповідність законопроекту Конституції, міжнародним зобов'язанням, виявляє конфлікти норм. *Промпт:* «Як конституційний юрист, перевірте, чи суперечить ця норма статті 19 Конституції України».

– Агент-фінансист: оцінює бюджетні наслідки, виявляє ризики перевищення видатків, аналізує ефективність фінансування програм. *Промпт:* «Як фінансовий аналітик, оцініть вплив цього законопроекту на державний бюджет на 2025 рік».

– Агент-соціолог: прогнозує соціальний вплив, аналізує зворотний зв'язок від громадян, виявляє потенційні джерела недовіри. *Промпт:* «Як соціальний аналітик, визначте, як ця норма може вплинути на довіру громадян до держави».

Кожен агент може бути активований за потребою, формуючи колективний аналіз з різних поглядів.

Формалізація виводу

Для того щоб вивід агента міг бути інтегрований в інші системи, він повинен бути формалізований – перетворений на структурований, машинно-читабельний формат. Це досягається шляхом вимоги до промпту:

- використання чіткого формату (наприклад, Причина; Наслідок);
- обмеження кількості слів (наприклад, не більше трьох на поняття);
- представлення відповіді у табличному вигляді (CSV, JSON).

Наприклад, відповідь агента-контролера може мати вигляд:

- *Загроза; Опис; Рекомендація*

- *Відсутність визначення критичної інфраструктури; Неясно, які системи підлягають захисту; Уточнити перелік об'єктів у законі*
- *Ризик витоку даних; Немає механізму повідомлення про порушення; Впровадити обов'язок сповіщення*

Такий вивід може бути імпортований в Gephi або в SemanticCore[109].

Інтеграція з системами типу SemanticCore

Для ефективного використання агентів необхідна їх централізована інтеграція. Це забезпечується за допомогою спеціалізованих систем, таких як SemanticCore – комп'ютерна програма для аналізу семантичних зв'язків, яка дозволяє:

- зберігати профілі агентів;
- керувати їхніми промптами та налаштуваннями;
- архівувати результати аналізу;
- формувати інтегровані мережі знань.

Інтеграція з такими системами перетворює набір ізольованих агентів на єдину інтелектуальну систему контролю, здатну до автономного функціонування, оновлення та масштабування.

Отже, промпт-кероване програмування – це не просто методологія генерації тексту, а нова форма інженерії інтелектуальних систем. Воно дозволяє перетворити LLM на спеціалізованих віртуальних експертів, які працюють як частина структурованої, відтворюваної, підзвітної системи парламентського контролю.

4.3. Внутрішній стан агента: пам'ять, контекст, історія взаємодій

Для ефективного застосування віртуальних експертів у системі парламентського контролю недостатньо лише формалізувати їхні ролі та повноваження. Критично важливим є забезпечення внутрішнього стану агента – сукупності даних, які дозволяють йому діяти не як ізольований інструмент відповідей, а як постійний, адаптивний аналітик, здатний до накопичення досвіду, узгодження дій та уникнення дублювання. Внутрішній стан включає три основні компоненти: пам'ять, контекст та історію взаємодій. Їх реалізація є необхідною умовою для створення автономної, інтелектуально підсиленої системи контролю.

Обмеження контекстного вікна LLM

Фундаментальною проблемою сучасних великих мовних моделей є обмеженість контекстного вікна – максимальної кількості токенів (слів,

[109] Комп'ютерна програма аналізу семантичних зв'язків "SemanticCore" Ланде Д.В., Фегер А.П. Україна. Свідоцтво про реєстрацію авторського права на твір N 125039 від 24.04.2024

частин слів), які модель може враховувати одночасно. Це означає, що модель має доступ лише до інформації, яка безпосередньо включена в поточний запит або попередні повідомлення в межах однієї сесії. Поза межами цього вікна або після завершення сесії модель «забуває» всю попередню взаємодію. Це робить її неспроможною на таке:

- зберігати довгострокові висновки;
- відстежувати еволюцію аналізу;
- уникати повторного аналізу вже вирішених питань.

Для парламентського контролю, де аналіз часто є багатоетапним і тривалим, така обмеженість є критичною. Тому необхідно вийти за межі внутрішнього контексту моделі та впровадити зовнішні механізми зберігання стану.

Необхідність внутрішнього стану

Внутрішній стан агента – це його інформаційна основа, яка забезпечує безперервність, узгодженість та ефективність діяльності. Він складається з трьох взаємопов'язаних компонентів:

- Пам'ять: здатність зберігати основні висновки, виявлені сутності, причинно-наслідкові зв'язки, норми та їх інтерпретації для подальшого використання. Наприклад, агент-юрист повинен пам'ятати, які норми вже були визнані неконституційними, щоб не аналізувати їх знову.
- Контекст: збереження загальної мети, теми та початкового стану аналізу. Це дозволяє агенту розуміти, як поточне завдання пов'язане з попередніми кроками, і діяти в єдиному напрямку. Наприклад, при аналізі законопроекту контекст включає мету – «виявлення суперечностей з міжнародними зобов'язаннями».
- Історія взаємодій: реєстрація всіх попередніх дій агента, включаючи сформовані відповіді, зміни в семантичній мережі, виявлені конфлікти та прийняті рішення. Це дозволяє відстежувати еволюцію аналізу, забезпечує прозорість та є основою для аудиту.

Відсутність внутрішнього стану перетворює агента на пасивний інструмент, який кожного разу починає аналіз «з нуля». Наявність стану робить його активним учасником процесу контролю.

Механізми подолання обмежень

Для подолання обмежень LLM і створення повноцінного внутрішнього стану використовуються наступні механізми:

Використання зовнішньої бази знань: основним елементом є інтеграція зі спеціалізованими системами для зберігання даних, такими як Dataset[110] та

[110] Комп'ютерна програма "Програмне забезпечення формування знань в базі знань інтелектуальної системи управління інформацією та подіями безпеки "Dataset" ("Dataset").

SemanticCore. Ці системи виступають як зовнішня пам’ять агента, де зберігаються:

- виявлені сутності та зв’язки;
- семантичні мережі;
- висновки попередніх аналізів;
- списки перевірених норм.

Це дозволяє агенту «згадувати» інформацію з минулих сесій.

– Передача контексту через RAG (Retrieval-Augmented Generation): технологія RAG дозволяє автоматично витягувати релевантну інформацію з зовнішньої бази знань і додавати її до промпту. Наприклад, перед аналізом нового законопроекту система може автоматично знайти всі попередні висновки про схожі норми та включити їх у контекст запиту. Це забезпечує безперервність аналізу і уникнення дублювання.

– Формування профілю агента: кожен віртуальний експерт має свій профіль – структурований запис, який оновлюється після кожної взаємодії. Профіль містить історію завдань, найчастіші висновки, уподобані методи аналізу, зв’язки з іншими агентами.

Такий профіль дозволяє агенту адаптуватися до конкретних потреб користувача та розвивати свою «експертну ідентичність».

Приклад: агент-контролер

Внутрішній стан агента – це не технічна деталь, а фундаментальний елемент інтелектуальної системи контролю. Він перетворює віртуального експерта з інструменту одноразового використання в постійного, навчального аналітика, здатного до узгодженої, прозорої та ефективної діяльності. Реалізація цього стану через інтеграцію з зовнішніми базами знань та технологією RAG є основним кроком до створення автономних систем парламентського контролю.

Розглянемо практичний приклад. Агент-контролер призначений для перевірки законопроектів на відповідність чинному законодавству. Без внутрішнього стану він буде аналізувати кожну норму незалежно, навіть якщо вона вже була перевірена раніше. З наявністю внутрішнього стану процес змінюється:

1. Перед запитом система використовує RAG, щоб витягнути з бази SemanticCore список норм, які вже були визнані неконституційними.

2. Цей список додається до промпту: «При аналізі цього законопроекту врахуйте, що наступні норми вже визнані неконституційними: [список]».
3. Агент-контролер ігнорує ці норми або позначає їх як «вже перевірені», зосереджуючись на нових або змінених положеннях.
4. Після завершення аналізу нові висновки додаються до бази знань, оновлюючи профіль агента.

Цей механізм значно підвищує ефективність та зменшує навантаження на систему.

4.4. Моделювання поведінки: реакція, ініціація, адаптація

Для ефективного використання віртуальних експертів у системі парламентського контролю недостатньо лише визначити їхні ролі та забезпечити внутрішній стан. Критично важливим є моделювання їхньої поведінки як динамічного процесу, що включає три основні компоненти: реакцію, ініціацію та адаптацію. Саме ці аспекти перетворюють віртуального експерта з пасивного інструменту генерації відповідей на активного учасника аналітичного процесу, здатного до автономного функціонування, проактивного моніторингу та еволюції.

Моделювання такої поведінки є основою для створення інтелектуально підсиленої системи контролю, здатної до прогнозування, виявлення та запобігання потенційним конфліктам.

Поведінка агента як динамічний процес

Поведінка віртуального агента розглядається як сукупність дій, які він здійснює у відповідь на зовнішні стимули або на основі внутрішнього стану. Цей процес є динамічним, оскільки він залежить від контексту, історії взаємодій, наявних даних та поточних цілей аналізу. Для моделювання поведінки використовуються логічні примітиви (умови, функції, переходи), які формалізують його дії, роблячи їх відтворюваними, верифікованими та інтегрованими в більші системи.

Реакція: відповідь на запит, подію, зміну в законодавстві

Реакція – це базова форма поведінки агента, яка активується зовнішнім запитом або подією. Це може бути:

- прямий запит від користувача: *«Проаналізуй цей законопроект»*;
- автоматична подія: *«Опубліковано новий указ Кабінету Міністрів»*;
- зміна в базі даних: *«Змінено норму у законі про податки»*.

На такі події агент реагує, запускаючи відповідний аналітичний процес. Наприклад, агент-юрист може автоматично запустити перевірку нового законопроекту на відповідність Конституції. Ця форма поведінки є реактивною, але вона забезпечує оперативність та прозорість контролю.

Ініціація: самостійне виявлення проблем і запуск аналізу

Ініціація – це проактивна форма поведінки, коли агент самостійно виявляє потенційну проблему та ініціює аналіз без зовнішнього запиту. Це можливо завдяки:

– постійному моніторингу баз даних законодавства, звітів, скарг;

– аналізу відхилень від нормативних показників (наприклад, прострочення подання звіту);

– виявленню аномалій у семантичних мережах (наприклад, раптове зростання кількості скарг на певну тему).

Наприклад, агент-моніторинг може виявити, що кількість скарг на блокування податкових накладних у певному регіоні зросла на 300% за місяць, і автоматично запустити аналіз діяльності місцевої податкової інспекції. Ця здатність робить систему превентивною, здатною виявляти проблеми на ранніх стадіях.

Адаптація: навчання на основі нових даних, зміна стратегії аналізу

Адаптація – це здатність агента навчатися на основі нових даних та змінювати свою стратегію аналізу. Хоча LLM не навчаються в реальному часі, їхній внутрішній стан (пам'ять, контекст, історія взаємодій) може оновлюватися, що дозволяє імітувати навчання. Наприклад:

– якщо після кількох аналізів виявляється, що певна норма часто призводить до суперечностей, агент може збільшити її вагу в подальших перевірках;

– якщо певний тип скарг систематично вказує на проблеми в одному відомстві, агент може змінити пріоритети моніторингу.

Це забезпечує еволюцію системи, роблячи її більш ефективною з часом.

Моделювання взаємодії

Основним аспектом моделювання поведінки є організація взаємодії між агентами. Вони можуть діяти не ізольовано, а як частина рою віртуальних експертів, де кожен передає результати своєї роботи іншому. Наприклад:

Агент-аналітик виявляє конфлікт між нормами → передає дані агенту-контролеру → той ініціює формування звіту та надсилає його агенту-інтерпретатору для тлумачення → кінцевий звіт надходить парламентарю.

Така ланцюжкова обробка моделює реальний аналітичний процес і забезпечує комплексність висновків.

Використання логічних примітивів

Поведінка агента формалізується через логічні примітиви, що робить її виконуваною та відтворюваною.

Приклад:

```
IF new_law_published:
  agent_analyst = initiate_analysis(law_text)
  IF agent_analyst.detects_conflict():
    agent_controller = generate_report(agent_analyst.findings)
    IF agent_controller.confirms_violation():
      agent_interpreter = interpret_norms(agent_controller.report)
      submit_to_parliament(agent_interpreter.final_report)
```

Цей псевдокод демонструє, як поведінка агентів може бути запрограмована через умови та функції, перетворюючи їх на виконувані модулі системи контролю.

Таким чином, моделювання поведінки віртуальних експертів – це не просто імітація людських дій, а інженерія інтелектуальних процесів, заснована на реакції, ініціації та адаптації. Саме такий підхід дозволяє створити автономну, проактивну та еволюційну систему парламентського контролю, здатну забезпечити високу якість підтримки рішень у умовах цифрової трансформації держави.

4.5. Обмеження агентської логіки: наразі – програмні моделі, не масове застосування

Незважаючи на значний потенціал концепції віртуальних експертів та агентської логіки для трансформації парламентського контролю, її практична реалізація на сьогодні стикається з численними технічними, організаційними та методологічними обмеженнями. Сучасні системи, побудовані на основі великих мовних моделей, ще не є повноцінними автономними агентами у класичному розумінні, а скоріше є програмними моделями, які імітують агентну поведінку в обмежених умовах. Вони вимагають постійного людського втручання, не мають механізмів масштабованої оркестрації та не інтегровані в єдині платформи. У цьому пункті систематично розглянуто основні обмеження, які перешкоджають переходу від концепту до масштабованої системи парламентського контролю.

Технічні обмеження

Фундаментальні недоліки поточних LLM визначають межі їхньої ефективності як бази для створення автономних агентів. Серед таких недоліків можна назвати відсутність постійної пам’яті у LLM, обмеження на кількість токенів у контексті, неможливість автономного виконання без промптів.

Відсутність постійної пам’яті у LLM приводить до того, що такі системи не мають здатності до довгострокового зберігання інформації. Після завершення сесії вся історія взаємодії втрачається. Це робить неможливим формування індивідуального досвіду віртуального експерта, що є критичним для

накопичення знань, уникнення повторних помилок та адаптації. Хоча це можна частково компенсувати за допомогою зовнішніх баз даних (наприклад, SemanticCore), сама модель залишається «амнестичною».

Контекстне вікно LLM (наприклад, 32k, 128k токенів) обмежує обсяг інформації, яку модель може враховувати одночасно. Для аналізу великих документів, таких як звіти уряду або масиви законів, цього часто недостатньо. Це змушує розбивати текст на фрагменти, що призводить до втрати цілісності аналізу та ризику упущення важливих зв'язків.

Крім того, сучасні віртуальні експерти не можуть ініціювати дії самостійно. Вони активуються лише відповідно на прямий запит користувача. Відсутність механізмів автономного моніторингу та самозапуску на основі змін у зовнішніх джерелах (наприклад, опублікування нового закону) робить систему реактивною, а не проактивною.

Проблеми з оркестрацією

Для ефективної роботи рою віртуальних експертів необхідна їхня синхронізація та координація, яка наразі є слабко розвиненою, через відсутність механізмів синхронізації дій між агентами. Коли кілька агентів (наприклад, аналітик, контролер, інтерпретатор) працюють над одним завданням, відсутні стандартизовані протоколи для обміну даними, передачі результатів або узгодження висновків. Це призводить до дублювання аналізу або протиріч у висновках, оскільки кожен агент діє ізольовано.

Крім того, зазвичай немає центрального координатора для управління роєм. У відсутність центрального агента-організатора, який би розподіляв завдання, контролював черги та інтегрував результати, взаємодія між агентами залишається хаотичною. Такий координатор міг би виступати як «головний мозок» системи, забезпечуючи цілісність процесу.

Програмні моделі замість масштабованої системи

На сьогоднішньому етапі розробки агентської логіки переважають концептуальні прототипи, а не впроваджені, масштабовані системи. Поточні розробки (наприклад, AgentFlow[111]) є концептами, а не впровадженими системами: Проекти, такі як AgentFlow, які пропонують фреймворки для побудови рою віртуальних агентів, знаходяться на стадії дослідження та експериментів. Вони демонструють потенціал, але ще не мають інтегрованих, стабільних реалізацій, придатних для постійного використання в умовах парламенту. При цьому агенти працюють ізольовано, без інтеграції в єдину платформу: Кожен віртуальний експерт зазвичай функціонує в окремому інтерфейсі (наприклад, окрема сесія ChatGPT), що ускладнює їхню взаємодію. Відсутність єдиної платформи, де агенти могли б спілкуватися,

[111] Lande Dmitry, Strashnoy Leonard. AgentFlow – No-Code Agent Framework Based on Logical Primitives. SSRN Preprint: 5285664, DOI: 10.2139/ssrn.5285664 (Jun 22, 2025). – 12 pp.

обмінюватися даними та спільно працювати над завданням, обмежує їхню ефективність.

Відсутність стандартизації

Однією з найсерйозніших перешкод для масштабування є відсутність єдиних стандартів. На цей час немає єдиних форматів для опису ролей, повноважень, інтерфейсів взаємодії: Відсутні стандартизовані схеми, які б визначали, як має бути описаний віртуальний експерт, які він має повноваження, які методи аналізу використовує, як передає дані. Це робить неможливим створення бібліотеки уніфікованих агентів, які можна було б легко інтегрувати в різні системи.

Перспективи

Незважаючи на поточні обмеження, напрямок розвитку агентської логіки має чіткі перспективи:

- Розвиток агентних фреймворків (наприклад, AgentFlow): Очікується, що такі фреймворки еволюціонують у повноцінні платформи для створення, керування та оркестрації рою агентів.
- Створення цифрових двійників експертів: Можливо, в майбутньому будуть створені цифрові двійники реальних експертів, які навчаються на їхніх публікаціях, виступах та аналітичних звітах, що дозволить відтворювати їхню логіку мислення.
- Інтеграція з системами парламенту через API: Основним кроком стане інтеграція агентів з внутрішніми системами парламенту (наприклад, базами законодавства, системами документообігу) через API, що забезпечить автономний моніторинг, аналіз та формування звітів.

Отже, агентська логіка на сьогодні є потенційною, але ще не визнаною технологією. Вона перебуває на етапі моделей, які демонструють концептуальну можливість створення інтелектуальної системи контролю, але не мають необхідної технічної, організаційної та нормативної інфраструктури для масштабованого застосування. Подолання цих обмежень стане основним завданням на шляху до створення автономної, адаптивної та проактивної системи парламентського контролю.

4.6. Концепція рою віртуальних експертів

Однією з ключових проблем при використанні великих мовних моделей є їхня схильність до галюцинацій – генерації інформації, яка може бути неточною або навіть помилковою. Це стає особливо критичним у таких сферах, як аналіз відкритих джерел інтелектуальної інформації (OSINT), де кожне слово може мати стратегічне значення. Тому на сьогоднішньому етапі важливим завданням є побудова надійних механізмів верифікації, які забезпечують точність отриманих результатів.

Першим кроком у цьому напрямку є перехресна перевірка отриманих висновків за допомогою різноманітних джерел. Якщо факти, виявлені моделлю, підтверджуються в інших надійних джерелах, це суттєво збільшує їхню довірчу базу. Крім того, необхідно проводити логічний контроль – перевірку внутрішньої послідовності висновків, щоб уникнути суперечностей або абсурдних зв'язків. У процесі верифікації корисними виявилися спеціалізовані інструменти, такі як FactCheck.org, Google Reverse Image Search, InVID для перевірки медійного контенту. Разом із автоматизованими системами важливе місце посідає людський фактор – фіналізація та оцінка критичних висновків повинна залишатися в руках кваліфікованого аналітика.

Не менш важливим є систематичне документування всіх запитів і отриманих відповідей. Ведення історії змін у промптах і відповідях дає змогу в будь-який момент повернутися назад, проаналізувати еволюцію думок і знайти джерело можливих помилок. Також аналітики все частіше впроваджують метрики якості – показники, які дозволяють кількісно оцінювати результати роботи моделі: точність витягання фактів, частку правильних узагальнень, кількість помилкових класифікацій тощо.

Однією з найперспективніших областей використання LLM є витяг понять і зв'язків між ними – основа для побудови семантичних мереж . Ці структури мають широке застосування: від аналізу великих масивів даних до формування систем знань. Проте, навіть найсучасніші моделі часом генерують зайві чи неправильні зв'язки, або, навпаки, пропускають важливі концепти. Це може мати серйозні наслідки, особливо в задачах, пов'язаних з безпекою, де пропущена загроза може перетворитися на реальну небезпеку.

Щоб зменшити ризики, запропоновано підхід, заснований на множинних взаємодіях з LLM – виклик моделі кілька разів з різними промптами, ролями та навіть різними моделями. Такий метод дозволяє отримувати кілька «поглядів» на одну й ту саму інформацію, що збільшує шанс на виявлення важливих концептів і зменшує ймовірність помилок. Проте він також породжує питання: скільки потрібно запитів, щоб отримати максимально повну картину, і наскільки необхідною залишається людина в процесі верифікації?

Ключовою ідеєю стало введення концепції віртуального експерта – образу, який представляє модель у кожному конкретному запиті. Кожен віртуальний експерт, формулюючи відповідь, одночасно вносить свої «думки» у загальну мережу: він виділяє ключові поняття, встановлює зв'язки між ними, вказує їхні атрибути. Таким чином, LLM перетворюється на лінгвістичний процесор, здатний не лише читати текст, а й активно будувати структуроване знання.

Справжній прогрес у цій галузі досягається через «рій віртуальних експертів» – сукупність відповідей, отриманих різними способами, з різних моделей, у різний час. Цей підхід нагадує колективне обговорення теми, де кожен учасник пропонує свою точку зору, а разом вони формують повнішу

картину. Результатом такого співпраці є семантична мережа, багата не лише кількістю зв'язків, але й їхньою глибиною, точністю та логічною узгодженістю.

Ця мережа стає платформою для подальшої обробки: аналітик може фільтрувати, групувати, ранжирувати отримані дані, вибираючи найбільш значущі елементи. Кожен окремий запит, кожен «віртуальний експерт» – це наче окрема крапля води, яка, об'єднуючись з іншими, утворює велике море знань, спільною метою якого є створення максимально точної і повної картини світу.

Незважаючи на всю потужність і ерудицію LLM, роль людини у цьому процесі залишається неперевершеною. Саме аналітик задає мету, встановлює напрямок, вибирає контекст і, найголовніше – перевіряє результати. Його досвід дозволяє відрізняти суттєве від другорядного, виявляти ключові поняття та їхні взаємозв'язки, а також визначати, які з них варто включити в кінцеву мережу.

Таким чином, виникає гібридний підхід, де штучний інтелект і людина працюють разом, доповнюючи одне одного. Модель пропонує ідеї, а людина направляє їх у правильне русло, створюючи справжній симбіоз між технологіями і людським розумом.

Практичне застосування такого підходу охоплює широке поле діяльності: від аналізу текстів і звітів до моніторингу тенденцій, виявлення дезінформації, створення рекомендацій, навчальних матеріалів і навіть стратегічного планування. Наприклад, аналізуючи численні джерела, рій віртуальних експертів може виявити приховані залежності, відстежити динаміку змін, виявити суперечливі чи потенційно неправдиві твердження.

У сфері бізнесу або політики це може стати основою для прийняття стратегічних рішень, а в освітній сфері – допомогти у формуванні структурованих курсів на основі наукових публікацій чи аналітичних звітів. Більше того, рій віртуальних експертів може оперативно оновлювати існуючі мережі, включаючи нову інформацію та адаптуючи структуру знань до змінних умов.

Впровадження великих мовних моделей, таких як GPT, здійснило революцію в задачах обробки природної мови, включаючи витягнення понять та зв'язків із неструктурованих даних. Однак залишається значна проблема – схильність LLM до «галюцинацій», коли модель генерує нерелевантні концепти або не може виявити ключові зв'язки між важливими сутностями у сфері кібербезпеки[112].

[112]Gabrijela Perković, Antun Drobnjak, Ivica Botički. Hallucinations in LLMs: Understanding and Addressing Challenges. In 2024 47th MIPRO ICT and Electronics Convention (MIPRO) DOI: 10.1109/MIPRO60963.2024.10569238

Метод рою віртуальних експертів покликаний вирішити ці проблеми за рахунок використання кількох ітерацій запитів до однієї або кількох LLM для витягнення концептів та взаємозв'язків. Нещодавні дослідження показують, що подібні методи можуть використовуватися для підвищення точності в інших галузях, таких як медична діагностика та фінансовий аналіз[113, 114,115].

Важливою темою є також методика використання ролевих моделей для розширення різноманітності концепцій. Дослідження[116] пропонує методики, схожі на "рій експертів", проте без застосування багаторазового виконання промптів із різними ролями. Також важливим напрямком є включення людського фактора на етапі верифікації.

Підхід, заснований на використанні «рою віртуальних експертів», перегукується з концепціями рою інтелектуальних агентів, які обговорюються у роботі[117]. У той час як попередні роботи фокусувалися на створенні спеціалізованих агентів для конкретних задач, підхід рою віртуальних експертів пропонує широкі можливості для гнучкої адаптації до складних сценаріїв.

Питання генерації правильних та релевантних сутностей при багаторазовому виконанні промптів також піднімаються в низці статей, таких як[118], де вивчається вплив когнітивних спотворень та повторного використання промптів на якість витягнутих даних.

Узагальнюючи вже наведене, визначаємо, що під «роєм віртуальних експертів» мається на увазі група відповідей, отриманих від великої мовної моделі на серію запитів (промптів), спрямованих на дослідження певної галузі знань. Кожна окрема відповідь від LLM виступає як думка «віртуального експерта», відображаючи інтерпретацію даних моделі на

[113]Akshay Goel, Almog Gueta, Omry Gilon. LLMs Accelerate Annotation for Medical Information Extraction. Proceedings of the 3rd Machine Learning for Health Symposium, PMLR 225:82-100, 2023. URL: http://dblp.uni-trier.de/db/conf/ml4h/ml4h2023.html #GoelGGLENHJRKSL23

[114]Rian Dolphin, Joe Dursun, Jonathan Chow, Jarrett Blankenship, Katie Adams, Quinton Pike. Extracting Structured Insights from Financial News: An Augmented LLM Driven Approach (2024). Preprint arXiv, arXiv:2407.15788. DOI: 10.48550/arXiv.2407.15788

[115]Isabella Catharina Wiest, Fabian Wolf, Marie-Elisabeth Leßmann, et al. "LLM-AIx: An open source pipeline for Information Extraction from unstructured medical text based on privacy preserving Large Language Models. medRxiv (2024): 2024-09. DOI: 10.1101/2024.09.02.24312917

[116]Kommineni, Vamsi Krishna, Birgitta König-Ries, Sheeba Samuel. From human experts to machines: An LLM supported approach to ontology and knowledge graph construction. Preprint arXiv:2403.08345 (2024).DOI: 10.48550/arXiv.2403.08345

[117] Steenstra I., Nouraei F., Arjmand M., Bickmore T. W. (2024). Virtual agents for alcohol use counseling: exploring LLM-powered motivational interviewing. arXiv preprint arXiv:2407.08095. DOI: 10.48550/arXiv.2407.08095

[118] Huang, Z., Zhang, Z., Hua, C., Liao, B. and Li, S., 2024. Leveraging enhanced egret swarm optimization algorithm and artificial intelligence-driven prompt strategies for portfolio selection. *Scientific Reports*, *14*(1), p.26681.

конкретний запит. Однак одна відповідь може містити лише частину можливої інформації, особливо якщо запит вимагає суб'єктивної інтерпретації або комплексного аналізу.

Щоб підвищити точність і повноту семантичної мережі, фахівці застосовують підхід «рою», багаторазово звертаючись до LLM з різними промптами і навіть на різних моделях, щоб накопичити широку сукупність точок зору та інтерпретацій[119]. Ці відповіді разом можна агрегаційно обробляти, щоб виявляти ключові вузли (поняття) та зв'язки між ними, а також їхні властивості, які найчастіше збігаються серед усіх відповідей. Статистична обробка множини відповідей і залучення «людині-ментора» дозволяє покращити семантичну мережу, виявляючи найважливіші концепти та зв'язки. Таким чином, «рій віртуальних експертів» є потужним інструментом у створенні повноцінних і точних семантичних мереж, пропонуючи симбіоз людського аналізу та можливостей ШІ.

У контексті терміна «рій віртуальних експертів» узгодженість дій віртуальних експертів – важливий аспект, який дозволяє створювати цілісні та достовірні семантичні мережі. Незважаючи на те, що кожен «експерт» є окремим запуском промпта, існує кілька механізмів, які сприяють узгодженості їхньої роботи та допомагають створювати злагоджені результати, а саме:

– Частота повторюваності вузлів і зв’язків у відповідях на схожі запити дозволяє виділяти більш значущі та стійкі концепти. Чим частіше зустрічається певний вузол або зв’язок, тим вища ймовірність його значущості для побудови мережі.

– Структурування результатів на основі ієрархії концептів допомагає вибудовувати мережу з пріоритетом ключових понять. Це дозволяє виділити вузли вищого рівня, які об’єднують під собою пов’язані, але менш значущі елементи.

– У сучасних LLM часто зберігається інформація про попередні запити в межах однієї сесії, що допомагає моделям підтримувати певну узгодженість у відповідях. Крім того, використання кумулятивних запитів, які уточнюють або розвивають попередні, дозволяє уточнювати зв’язки та корегувати мережеві вузли.

– Людина виступає як ментор рою віртуальних експертів, задаючи послідовність запитів і відстежуючи відповіді. Це дозволяє корегувати питання та додавати нові, за потреби перевіряючи гіпотези. Людський аналіз дозволяє уникнути випадкових відповідей, відкидаючи нерелевантні зв’язки та підкріплюючи найбільш значущі.

[119] Lande Dmitry, Strashnoy, Leonard. Implementation Of The Concept Of A "Swarm Of Virtual Experts" In The Formation Of Semantic Networks In The Field Of Cybersecurity Based On Large Language Models. SSRN Preprint: 4978924, DOI: 10.2139/ssrn.4978924 (Oct 17, 2024). - 15 p.

– Для різних вузлів і зв’язків можуть призначатися вагові значення, які відображають їхню значущість на основі повторюваності або унікальності. При цьому зв’язки з високими ваговими значеннями розглядаються як надійніші, що сприяє створенню більш точної та узгодженої семантичної мережі.

Ці механізми дозволяють «рою віртуальних експертів» бути не просто випадковим набором думок, а цілеспрямованим процесом формування надійних знань. Завдяки узгодженості дій рій віртуальних експертів у підсумку будує повну та стійку семантичну мережу, де людина та ШІ діють як єдина система.

Щоб збільшити різноманітність і точність витягнутих зв’язків, запропоновано декілька методик, серед яких вилучення сутностей і зв’язків за допомогою LLM, використання ролей, застосування різних моделей LLM, агрегація результатів, верифікація людиною.

На рис. 4 наведено схему, яка ілюструє взаємодію між агентами, ролями, клієнтом і LLM у межах рою віртуальних експертів.

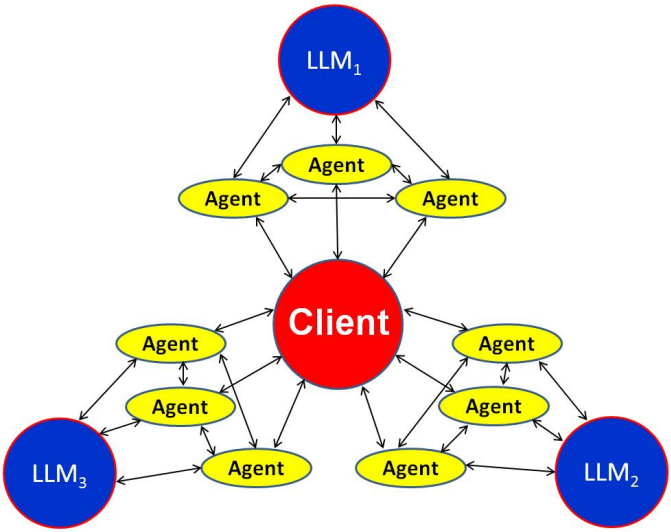

Рисунок 4: Схема рою віртуальних агентів.

При цьому розглядаються такі процеси:

1. Вхідний запит q_k передається клієнтом до рою агентів.
2. Кожен агент отримує запит і генерує відповідь із урахуванням ролі.
3. Відповіді агентів обмінюються між собою (взаємодія).
4. Клієнт синхронізує та агрегує відповіді від різних LLM.

5. Результат узгоджується із цілями.

Як бачимо, рій утворюється також завдяки зв'язку між агентами та клієнтом (людиною), який координує процес. Основними аспектами зв'язку є обмін інформацією між агентами, загальний клієнтський контекст, агрегація результатів людиною. Один агент може враховувати відповідь іншого для уточнення свого результату. Агенти працюють із загальною метою, заданою клієнтом, але виконують різні ролі. Результати кожного агента аналізуються та узгоджуються на рівні клієнта для формування єдиного рішення.

Математично, «рій віртуальних експертів» можна представити як множину взаємопов'язаних агентів: $A = \{a_1, a_2, ..., a_n\}$,

де a_i – агент, а кожна його відповідь r_{ik} є функцією не тільки запиту q_k, а й результатів інших агентів $R_{-i,k}$:

$$r_{ik} = f\left(a_i, q_k, R_{-i,k}\right).$$

Основою методу є послідовне вилучення концептів і відносин між ними з тексту. Для цього використовується один і той самий промпт, який багаторазово переформулюється і запускається, щоб мінімізувати вплив повторюваних відповідей. Повторне виконання промпта допомагає доповнити систему, що збільшує ймовірність знаходження більш релевантних зв’язків. Перезавантаження системи дозволяє генерувати більш різноманітні результати, уникнувши повторень і посиливши ймовірність вилучення унікальних і релевантних зв’язків.

Для різноманітності відгуків і мінімізації помилок була додана методика зміни ролей. Кожен запит до LLM формулюється з урахуванням ролі, наприклад, від імені експерта в певній галузі, що сприяє екстрагування нових понять і зв’язків. Різноманітність моделей дозволяє зменшити вплив особливостей конкретної моделі на результат і підвищити достовірність даних, що добуваються. Кожна модель генерує свої унікальні сутності та зв’язки, що підвищує повноту отримуваних семантичних мереж.

Після збору даних від різних моделей і в різних ролях проводиться їх агрегація. Зв’язкам приписуються ваги, що відповідають частоті їх появи в різних мережах. Зв’язки з низькими частотами можуть бути виключені, щоб уникнути хибних або малозначущих даних.

На завершальному етапі пропонується включення людини-експерта для верифікації мереж. Цей крок необхідний для мінімізації помилок, допущених LLM, особливо у випадках, коли важливі зв’язки пропущені, а неіснуючі – додані.

Визначення ролей – важливий аспект методології «рою віртуальних експертів», який дозволяє витягувати сутності та зв’язки в різних контекстах і з різних точок зору. Виділення ролей забезпечує багатоаспектне сприйняття даних, а також допомагає створювати семантичну мережу, здатну

інтегрувати різноманітні інтерпретації та підходи до однієї й тієї самої інформації.

Застосування різних ролей дозволяє віртуальним експертам формувати мережу сутностей та зв'язків, які включають різноманітні підходи та точки зору. Такий багатоаспектний аналіз робить семантичну мережу більш цілісною та гнучкою. Людина, як диригент, визначає загальну спрямованість та корегує ролі за необхідності, але LLM, виступаючи як віртуальні експерти, доповнюють процес, ініціюючи та пропонуючи нові, потенційно значущі ролі.

Таким чином, розподіл ролей між людиною та LLM не лише забезпечує глибину аналізу, але й дозволяє створювати семантичні мережі, здатні до більш точного та різнобічного відображення складних взаємозв'язків у досліджуваному контексті. Одним із ключових питань є визначення оптимальної кількості ролей віртуальних експертів. Надмірна кількість ролей може призводити до зниження якості мережі через інформаційний шум, тоді як недостатня кількість ролей не дозволить виявити всі можливі зв'язки. Для цього може використовуватися підхід статистичного аналізу приросту нових зв'язків. У запропонованій моделі «рою віртуальних експертів» оцінка кількості ролей та їх підтвердження людиною є важливими елементами для підвищення точності та надійності семантичних мереж, створюваних за допомогою LLM.

Людина, як ментор, може визначати ключові ролі, ґрунтуючись на цілях аналізу та специфіці предметної області. Ці ролі можуть представляти різні перспективи, підходи та контексти, такі як експертна, аналітична та оціночна ролі:

– Експертна роль полягає у вилученні сутностей та зв'язків із акцентом на наукові чи технічні терміни, підходи та концепції.

– Аналітична роль полягає в аналізі взаємозв'язків з точки зору причинно-наслідкових залежностей, послідовності подій тощо.

– Оціночна роль полягає у виявленні сутностей та зв'язків із фокусом на пріоритетах, значущості та вагах.

Задаючи конкретні ролі, людина формує спрямованість аналізу, вказуючи, які концепти та відносини є найважливішими для виділення. Це допомагає рою концентруватися на вилученні інформації в межах заданих перспектив та інтерпретацій.

Великі мовні моделі також можуть пропонувати власні ролі та контексти, які людина могла б не враховувати. LLM можуть розпізнавати та пропонувати ролі на основі структури даних або тексту, аналізу частотності понять або навіть на основі вбудованих семантичних категорій. Наприклад, модель може запропонувати історичні, соціальні та міждисциплінарні ролі.

- Історична роль інтерпретує інформацію з акцентом на історичний контекст та розвиток понять.
- Соціальна роль передбачає акцент на зв'язках між соціальними аспектами даних, аналіз інформації в рамках взаємодій та їх значущості в суспільстві.
- Міждисциплінарна роль забезпечує вилучення даних з точки зору взаємозв'язку різних дисциплін або підходів.

LLM можуть виявляти такі ролі автоматично, використовуючи свої вбудовані знання та аналітичні можливості, що відкриває для рою віртуальних експертів можливість бачити дані з несподіваних та інноваційних перспектив.

Ролі експертів у моделі слугують для збільшення різноманітності відповідей. Кожна роль задає інший контекст або точку зору, з якої LLM може розглядати концепти та їх зв’язки. Це допомагає мінімізувати вплив можливих помилок або пропусків, які можуть виникнути при запитах від однієї ролі. Важливо знайти оптимальну кількість ролей. Надмірне збільшення кількості ролей може призвести до генерації зайвих, нерелевантних зв’язків, тоді як замала кількість не забезпечить достатнього різноманіття відповідей. Оптимум можна шукати через статистичний аналіз: після кількох ітерацій промптів із різними ролями можна спостерігати, за якої кількості ролей з’являється найбільш стабільна та релевантна мережа зв’язків.

Метод підрахунку ролей може базуватися на концепції збіжності. Якщо при додаванні нової ролі частота появи нових концептів і зв’язків починає зменшуватися, це може сигналізувати про те, що більшість релевантних концептів уже враховано, і подальше додавання ролей стає малоефективним. Це можна формалізувати через спостереження за приростом нових концептів з кожною додатковою роллю та встановленням порогу, при якому додавання нових ролей стає статистично незначущим.

Роль людини в моделі є фінальною перевіркою агрегованої семантичної мережі. Після того як LLM обробить численні промпти від різних ролей, людина-експерт виконує верифікацію, аналізуючи зв’язки між концептами, їх вагу та релевантність. Людина-експерт допомагає усунути галюцинації LLM, неважливі або малозначущі зв’язки, а також заповнити пропуски важливих зв’язків. Для зручності валідації результатів можна впровадити інтерактивні інтерфейси, які дозволять людині спостерігати за вагами зв’язків (отриманими на основі частоти їх виявлення віртуальними експертами) та корегувати їх за потреби.

Таким чином, система отримує додатковий рівень надійності завдяки включенню людини-експерта, який підтверджує або коригує результати, надані віртуальними експертами.

Для представлення математичної моделі рою віртуальних експертів введемо позначення:

- $S = \{s_1, s_2, ..., s_n\}$ – множина сутностей, що екстраговані з тексту;
- $R = \{r_{ij}\}$ – множина зв'язків між сутностями, де r_{ij} позначає зв'язок між сутностями s_i та s_j;
- P – набір промптів для витягу зв'язків;
- $M = \{M_1, M_2, ..., M_k\}$ – набір різних моделей LLM;
- $W(r_{ij})$ – вага зв'язку між сутностями s_i та s_j, яка залежіть від частоти повторення цього зв'язку.

Для кожного промпту $p \in P$ і кожної моделі $M_i \in M$:

- Виконується витягнення множини сутностей S_p^l та множини зв'язків R_p^l.
- Поновлюється вага зв'язку: $W(r_{ij}) = W(r_{ij}) + 1$, якщо $r_{ij} \in R_p^l$.

Введення порогу θ. Зв'язок r_{ij} включається до підсумкової мережі, якщо його вага перевищує певний поріг $W(r_{ij}) \geq \theta$, де θ поріг визначається як статистичний критерій, заснований на кількості запусків віртуальних експертів і частоті зв'язків.

Для визначення оптимального порогу θ визначимо:

- $f(r_{ij})$ – частота появи зв'язку r_{ij} в результатах різних промптів і моделей.
- T – загальна кількість викликів (промптів і моделей).
- частотний розподіл зв'язків $P(f)$, який показує ймовірність появи зв'язків з різною частотою.

Щоб вибрати оптимальний поріг θ, можна використовувати критерій значимості, заснований на кількості запусків і дисперсії частот зв'язків:

$$\theta = \mu \cdot f + \alpha \cdot \sigma_f,$$

де

- μf – середнє значення частоти появи зв'язків;
- σ_f – стандартне відхилення частоти;

- α – коефіцієнт значимості (наприклад, $\alpha = 1.5$, щоб відсіяти менш значимі зв'язки).

Цей поріг дозволяє відсіювати зв'язки, які є випадковими і не мають значного ваги.

Кількість викликів віртуальних експертів N прямо впливає на достовірність побудови мережі. Для оцінки оптимальної кількості запусків можна використовувати метод стабілізації частот зв'язків.

Нехай N – кількість запусків, $f\left(r_{ij}\right)$ – частота зв'язку r_{ij} після N запусків.

Якщо при збільшенні N приріст кількості нових зв'язків $\Delta f\left(r_{ij}\right)$ стає незначним (наприклад, $\Delta f\left(r_{ij}\right) < \varepsilon$, де ε – мале число), це вказує на досягнення стабільності. Таким чином, оптимальне N можна:

$$N = \min\left\{N' : \frac{1}{|R|}\sum_{r_{ij}\in R}\Delta f\left(r_{ij}\right) < \varepsilon\right\}.$$

Цей критерій дозволяє вибрати N, при якому модель стабілізується, і нові експерти не додають значущих зв'язків.

Наведемо формалізацію для оцінки кількості ролей та підтвердження:

1. Нехай R – кількість ролей, від яких запитуються дані у LLM. Для кожної ролі i, отримуємо множину концептів C_i та множину зв'язків між ними S_i.

2. Агрегована множина зв'язків: $S = \bigcup_{i=1}^{R} S_i$, кожному зв'язку призначається вага $w(s)$, яка дорівнює частоті появи зв'язку серед ролей:

   $w(s) = (\text{кількість ролей, у яких зв'язок } s \text{ зустрічається } w) / R$

3. Вводиться порогове значення τ, які зв'язки вважаються значущими. Наприклад, якщо зв'язок зустрічається у менш ніж τ долі ролей, він відкидається:

   $S^{*} = \left\{s \in S \mid w(s) \geq \tau\right\}.$

   Для оцінки необхідної кількості ролей R^{*} можна спостерігати за приростом нових зв'язків із додаванням нових ролей. Якщо приріст кількості нових концептів і зв'язків стає незначним (наприклад, $\frac{\Delta S_i}{\Delta R} \to 0$), можна зупинити збільшення кількості ролей.

4. Після побудови мережі S^*, людина-експерт проводить фінальну верифікацію, підтверджуючи або відхиляючи запропоновані зв'язки, що забезпечує фінальну корекцію мережі.

Для визначення оптимального порогу частотності зв'язків можна скористатися наступною методикою:

- Проводиться серія експериментів із різними значеннями порогу, починаючи з мінімальних значень (врахування всіх зв'язків) до збільшених порогів.
- Зв'язок із вагою 1 може виявитися найчастішим при малій кількості експертів. Щоб уникнути цього, у вибірці експериментів із різною кількістю ролей і різними порогами τ проводиться аналіз, який дозволяє оцінити, при якому значенні порогу приріст нових зв'язків стає незначним.
- Це значення і вважається оптимальним.

Людина-експерт підтверджує або коригує результати, надані віртуальними експертами.

Методологія "рою віртуальних експертів", при всій своїй корисності, також стикається з низкою проблем і обмежень, які важливо враховувати для коректного використання цього підходу. Нижче наведено основні складності та обмеження:

- Обмежена когерентність і суперечливість відповідей. Оскільки кожна відповідь від LLM формується незалежно, можуть виникати суперечності між різними «віртуальними експертами». Це ускладнює побудову узгодженої мережі та вимагає додаткової фільтрації або агрегування. Часто доводиться вдаватися до людського втручання, щоб вирішити суперечності, визначити релевантні зв'язки та виключити менш точні відповіді.
- Залежність від якості промптів. Промпти відіграють ключову роль у «рої віртуальних експертів», і якість відповідей безпосередньо залежить від їх точності та спрямованості. Недостатньо чітко сформульовані промпти можуть призвести до отримання нерелевантної інформації або втрати важливих зв'язків. Це вимагає від людини високої кваліфікації та експертизи у постановці правильних запитань.
- Проблеми з узгодженістю у довгостроковій перспективі. При використанні різних моделей LLM або при запитах у різний час відповіді можуть значно відрізнятися, особливо якщо відбуваються оновлення в даних або алгоритмах моделей. Це може призвести до ускладнень у підтримці послідовності результатів, що знижує достовірність і передбачуваність отримуваної інформації.

- Складність інтерпретації та контролю даних. У умовах, коли безліч віртуальних експертів пропонують свої точки зору та інтерпретації, людині-ментору стає складно обробляти та інтерпретувати всі доступні дані. Без добре опрацьованих методів агрегування інформації може бути важко виявити значущі зв'язки серед великої кількості пропозицій і побудувати цілісну семантичну мережу.
- Проблеми зі стійкістю та повнотою мережі. Хоча метод рою дозволяє підвищувати повноту семантичної мережі, вона все ж може бути неповною, особливо у випадку, якщо рій стикається з рідкісними або складно визначуваними концептами. Крім того, можливі пропуски в критично важливих зв'язках, що вимагає додаткових циклів запитів та вдосконалення методів їх побудови.
- Потенційне спотворення інформації. LLM, на основі яких будуються віртуальні експерти, навчені на обмежених і історично обумовлених даних, що може призвести до поширення упереджень або спотвореної інформації в семантичній мережі. Це обмеження особливо небезпечне, якщо результати мережі використовуються для прийняття рішень.
- Складність оновлення та масштабування мережі. По мірі додавання нових даних і появи нових концептів мережа може ставати громіздкою і складною для оновлення. Це вимагає від рою віртуальних експертів високої адаптивності та здатності корегувати існуючу мережу без втрати якості. Однак адаптивність LLM щодо нових даних не завжди задовольняє вимогам для швидкого і точного оновлення мереж.

Методологія "рою віртуальних експертів" пропонує цілий ряд переваг, які роблять її перспективною для створення та аналізу семантичних мереж. Ось основні з них:

- Гнучкість та адаптивність, які полягають у тому, що рій віртуальних експертів може оперативно підлаштовуватися під зміни в контексті та запитах. Шляхом модифікації промптів та взаємодії з різними LLM можна швидко налаштувати аналіз під специфічні задачі, виділяючи необхідні аспекти даних або концентруючись на конкретних деталях.
- Багатоаспектний аналіз завдяки тому, що віртуальні експерти здатні генерувати відповіді, виходячи з різних ролей і контекстів, що дозволяє розглядати один і той же предмет з різних точок зору. Це сприяє побудові більш глибокої та цілісної семантичної мережі, яка включає різноманітні інтерпретації та підходи. В результаті мережа стає більш інформативною.
- Використання LLM для генерації вузлів і зв'язків дозволяє автоматизувати процес побудови мережі, що значно економить час. Замість залучення численних людських експертів один оператор може

використовувати рій віртуальних експертів для виконання складних аналітичних задач, збільшуючи продуктивність і швидкість аналізу.

- Завдяки паралельній роботі віртуальних експертів можна ефективно обробляти великі масиви інформації, виділяючи з них найбільш значущі зв'язки та концепти. Це особливо корисно в ситуаціях, де потрібно аналізувати великі набори даних або велику кількість документів за короткий термін.
- Завдяки статистичній агрегації частотних вузлів і зв'язків підвищується точність і повнота мережі, оскільки рій поступово виявляє найбільш значущі елементи та усуває випадкові і менш важливі. За допомогою рою можна уточнювати та доповнювати семантичні мережі, запитуючи додаткову інформацію щодо спірних або неповних аспектів.
- Рій віртуальних експертів дозволяє підтримувати мережу в актуальному стані, використовуючи свіжі запити та обробляючи нові дані. Це особливо цінно в швидкозмінних галузях, де актуальність інформації має критичне значення. Рій можна призначати для аналізу нових даних і пошуку актуальних зв'язків, що допомагає уникнути застарілості мережі.

Хоча рій автоматизує значну частину роботи, він також може підтримувати творчі та аналітичні процеси людини. Віртуальні експерти можуть генерувати ідеї, які надихають на нові підходи та рішення, а також надавати людині додаткову інформацію, на основі якої можна приймати більш обгрунтовані рішення.

Таким чином, запропонований метод «рою віртуальних експертів» демонструє новий підхід до побудови семантичних мереж з використанням LLM, який поєднує в собі багаторазові запити, ролі, різні моделі та агрегування результатів, частково вирішуючи проблеми галюцинацій та надмірності зв'язків за рахунок агрегування результатів. Введення статистичного порогу та оцінка необхідної кількості запусків віртуальних експертів підвищує точність і достовірність мережі. Багаторазові запити з різними ролями збільшують ймовірність знаходження релевантних концептів та їх зв'язків. Значущість концепції ролей також проявляється в можливості покращити точність і різноманітність мережі. Статистичний аналіз приросту нових зв'язків допомагає визначити оптимальну кількість ролей, що знижує ймовірність інформаційного шуму. Важливо відзначити, що узгодженість дій рою досягається не лише за рахунок багаторазових запусків промптів, але й завдяки людській участі, а також внутрішнім механізмам LLM, які зберігають контекст запитів в межах однієї сесії.

Методологія «рою віртуальних експертів» є перспективним інструментом для побудови, аналізу та модифікації семантичних мереж із залученням LLM у ролі «віртуальних експертів». Основна перевага методології – це можливість автоматизувати процеси аналізу та прискорювати отримання

релевантної інформації, що дозволяє ефективно вирішувати широкий спектр задач, від обробки великих масивів даних до створення комплексних, багатоаспектних моделей.

Сильні сторони підходу включають високу гнучкість, економічну ефективність і здатність адаптуватися до швидкозмінних умов. Рій віртуальних експертів дозволяє генерувати глибинні, багатоаспектні погляди на аналізовані концепції, забезпечуючи при цьому високу повноту та актуальність даних. Однак метод стикається і з низкою проблем, таких як складність підтримки когерентності, необхідність ретельного налаштування промптів і періодичне втручання людини для вирішення суперечностей та управління мережею. Важливо відзначити, що хоча метод значно зменшує потребу в численних людських експертах, він не усуває її повністю. Людська участь залишається критично важливою на етапах постановки задач, фільтрації результатів і контролю якості мережі. Цей гібридний підхід, де людина виконує роль ментора та оператора, допомагає подолати слабкі сторони методології, такі як можлива неузгодженість відповідей та обмежена інтерпретованість деяких даних.

Запропонований підхід усуває розрив між автоматизованим аналізом, заснованим на ШІ, і людською експертизою, надаючи надійне рішення для обробки складних даних, зокрема, у сфері кібербезпеки. Подальші дослідження можуть включати розширення спектру ролей, що використовуються при генерації віртуальних експертів, тестування підходу на різних наборах даних звітів з кібербезпеки, а також покращення процесу валідації за участі людини шляхом інтеграції методів активного навчання.

Перспективи розвитку «рою віртуальних експертів» дуже широкі, особливо з урахуванням поточного розвитку LLM та можливостей їх використання в семантичному аналізі. Майбутнє цієї методології бачиться у такому:

- розробці вдосконалених механізмів агрегування та фільтрації даних для підвищення узгодженості мережі та автоматичного вирішення суперечностей між відповідями віртуальних експертів;
- створенні нових моделей взаємодії між людиною та LLM, де людина зможе більш гнучко керувати роями віртуальних експертів, задаючи точні критерії, що визначають, які відповіді слід вважати значущими та достовірними;
- покращенні механізмів узгодження ролей і контекстів, що дозволить досягати більш точного та комплексного аналізу без істотного втручання з боку людини;
- розширенні можливостей використання рою для специфічних галузей, таких як медична аналітика, фінансовий моніторинг або соціальні дослідження, де необхідність у обробці даних та аналізі складних взаємозв'язків вимагає високої повноти та точності.

Розділ 5. Причинно-наслідкові моделі для сценарного аналізу

Сценарний аналіз є одним із найефективніших інструментів стратегічного планування, особливо в умовах високої невизначеності та складності соціально-політичних процесів. У контексті парламентського контролю він дозволяє не лише реагувати на вже виниклі порушення, а й прогнозувати наслідки потенційних рішень, оцінювати альтернативні шляхи розвитку подій та виявляти потенційні ризики ще до їх реалізації. Однак традиційні методи сценарного аналізу часто обмежені великими часовими та експертними витратами, необхідністю залучення багатьох фахівців та суб'єктивністю інтерпретацій.

Сучасний розвиток генеративного штучного інтелекту відкриває нові можливості для автоматизованого формування сценаріїв на основі причинно-наслідкових (каузальних) моделей. Ці моделі дозволяють відобразити складні ланцюжки впливу між подіями, нормами, станами та рішеннями у вигляді структурованих, візуалізованих та формалізованих мереж. Вони стають основою для систематичного, відтворюваного та масштабованого аналізу, який може бути інтегрований в інтелектуальну систему парламентського контролю.

Важливим викликом у побудові таких моделей є створення повної та достовірної причинно-наслідкової мережі, що традиційно вимагає значних ресурсів і експертних знань. Запропонований у цьому розділі підхід ґрунтується на методі «динамічного семантичного нетворкінгу», який поєднує ідею двонаправленого пошуку з використанням генеративних мовних моделей, таких як ChatGPT. Цей підхід дозволяє формувати каузальні ланцюжки буквально в режимі реального часу, значно скорочуючи потребу в ручній праці.

Методологія базується на інтелектуальному аналізі тексту, формуванні семантичних мереж та подальшому відборі та ранжируванні сюжетних ланцюжків. У попередніх розділах було показано, як за допомогою промптів можна формувати мережі мови, виявляти сутності та зв'язки. У цьому розділі розглядається, як ці мережі перетворюються на причинно-наслідкові моделі, які стають основою для сценарного аналізу, оптимізації шляхів реформ та підтримки стратегічних рішень.

5.1. Побудова каузальних ланцюжків: від початкового стану до наслідків

Побудова причинно-наслідкових ланцюжків є фундаментальним етапом сценарного аналізу, який полягає у встановленні послідовності подій, дій та станів, що ведуть від існуючої проблеми до бажаного результату. Цей процес вимагає не лише виявлення окремих причин і наслідків, а й побудови логічно цілісного шляху, який може бути проаналізований, візуалізований та використаний для прогнозування. У цьому пункті розглянуто методологію

побудови таких ланцюжків з використанням генеративного штучного інтелекту.

Визначення початкового стану (проблеми) та кінцевої мети (бажаного стану)

Перший крок – чітке формулювання початкового стану (глобальної проблеми) та кінцевої мети (бажаного стану). Ці поняття виступають як «точки входу» та «точки виходу» для побудови мережі. Наприклад:

- Початковий стан: *«Парламентський контроль»*;
- Кінцева мета: *«Несуперечність документів уряду»*.

Ці поняття є досить загальними, але саме їхня широта дозволяє розгорнути глибокий аналіз, оскільки система ГШІ може декомпозувати їх на більш конкретні компоненти.

Двонаправлений пошук: побудова мережі від проблеми та від мети

Для ефективного формування причинно-наслідкової мережі використовується метод двонаправленого пошуку, суть якого полягає в паралельній побудові двох часткових мереж:

- Від початкового стану: виявлення часткових причин, що підтримують існуючу проблему (наприклад, *«парламентські запити»*, *«аналіз ефективності»*, *«громадська участь»*).
- Від кінцевої мети: виявлення чинників, що призводять до досягнення бажаного стану (наприклад, *«законодавча узгодженість»*, *«інституційна узгодженість»*, *«публічна легітимність»*).

Цей підхід, подібний до методу «двонаправленого пошуку» в теорії графів, дозволяє скоротити загальну кількість аналізованих вузлів і прискорити знаходження зв'язних шляхів між початком і кінцем.

Об'єднання часткових мереж у єдиний причинно-наслідковий ланцюжок

Коли обидві часткові мережі достатньо розширені, вони **об'єднуються** в єдину структуру. У міру розширення мереж вони «зустрічаються» на спільних поняттях (наприклад, *«громадська участь»*, *«законодавча узгодженість»*), утворюючи повні причинно-наслідкові ланцюжки. Ці ланцюжки відображають різні шляхи досягнення мети, кожен з яких може бути проаналізований окремо.

Наприклад, один із ланцюжків може мати вигляд:

Парламентський контроль → Парламентські слухання → Громадська участь → Публічна легітимність → Несуперечність документів уряду

Формалізація зв'язків у вигляді спрямованих графів

Отримані ланцюжки формалізуються у вигляді спрямованих графів, де вузли – це поняття (стану, дії, події), а ребра – причинно-наслідкові зв'язки між

ними. Така мережа може бути збережена у форматі, придатному для подальшої обробки (наприклад, CSV, GraphML), і імпортована в інструменти візуалізації, такі як Gephi.

Формалізація забезпечує:

- відтворюваність аналізу;
- можливість верифікації логічної цілісності;
- інтеграцію з іншими аналітичними системами.

Використання промптів для генерації ланцюжків

Важливим інструментом побудови мережі є промпт-кероване зондування. Для ініціалізації процесу використовуються структуровані промпти, наприклад:

«Виділіть у тексті всі пари сутностей, пов'язані з парламентським контролем. Представте у форматі: "Суб'єкт; Об'єкт"»,

або:

«Назвіть 10 чинників, які можуть призвести до стану "Законодавча узгодженість" для досягнення стану "Несуперечність документів уряду"»

Ці промпти дозволяють системі ГШІ виступати як інструмент семантичного аналізу, генеруючи структуровані дані для подальшої побудови мережі.

Таким чином, побудова каузальних ланцюжків – це не просто генерація тексту, а інженерний процес формування аналітичної моделі, яка дозволяє перейти від опису проблеми до моделювання шляхів її вирішення. Цей підхід значно скорочує ресурсні витрати на сценарний аналіз і робить його доступним для систематичного застосування в парламентському контролі.

5.2. Ієрархічне уточнення понять: від загального до конкретного

Ефективний сценарний аналіз у сфері парламентського контролю вимагає не лише побудови причинно-наслідкових ланцюжків, а й глибокого розуміння структури ключових понять, які лежать в основі політичних процесів. Абстрактні категорії, такі як *«Парламентський контроль»*, *«Несуперечність документів уряду»* або *«Законодавча узгодженість»*, є надто загальними для безпосереднього аналізу. Їх необхідно декомпозувати на більш конкретні компоненти, що дозволяє виявити внутрішню логіку явища, встановити зв'язки між елементами та створити основу для детального моделювання. Цей процес, відомий як ієрархічне уточнення понять, є основним етапом побудови структурованих, формалізованих моделей знань.

Декомпозиція загальних понять на часткові

Ієрархічне уточнення починається з декомпозиції вихідного поняття на множину часткових, більш конкретних понять. Це схоже на розкладання складного механізму на окремі деталі для вивчення його внутрішньої будови.

Наприклад, поняття *«Парламентський контроль»* може бути розкладене на такі часткові поняття:

«Аналіз законопроектів»,

«Моніторинг бюджету»,

«Парламентські слухання»,

«Подання запитань»,

«Акти вето»,

«Звітність влади» тощо.

Кожне з цих часткових понять є підкатегорією вихідного поняття і відображає окремий напрямок контролю. Така декомпозиція дозволяє зосередитися на конкретних механізмах реалізації контролю, що робить аналіз більш цілеспрямованим і глибоким.

Використання промптів для виявлення 10 часткових причин/чинників кожного поняття

Для систематичного виявлення часткових понять використовується метод промпт-керованого зондування великих мовних моделей, таких як ChatGPT. Формується спеціалізований промпт, який спрямовує модель на виконання конкретної аналітичної задачі. Наприклад:

«Розкладіть поняття «Парламентський контроль» на 10 часткових понять. Кожне часткове поняття повинно містити не більше трьох слів. Представте відповідь у формі: "часткове поняття; Парламентський контроль". Кожен запис в окремому рядку».

Відповідь системи:

Нагляд за владою; Парламентський контроль

Аналіз діяльності уряду; Парламентський контроль

Парламентські слухання; Парламентський контроль

Подання запитань; Парламентський контроль

Акти вето; Парламентський контроль...

Після цього процес продовжується на наступному рівні: для кожного з виявлених часткових понять формується окремий промпт:

«Назвіть 10 чинників поняття «Парламентські слухання» як частини поняття «Парламентський контроль». Кожний чинник має містити не більше трьох слів. У форматі "Чинник; Парламентські слухання"».

Відповідь:

Експертний обмін; Парламентські слухання

Публічний діалог; Парламентські слухання

Цей ітеративний процес дозволяє побудувати багаторівневу ієрархічну структуру, де кожен рівень розкриває зміст попереднього.

Обмеження на кількість слів (до трьох) для забезпечення уніфікації термінології

Критичним елементом методу є обмеження кількості слів у кожному понятті – не більше трьох. Це обмеження виконує кілька важливих функцій:

- забезпечує стандартизацію термінології, усуваючи довгі, неоднозначні формулювання;
- сприяє уніфікації виводу, що дозволяє порівнювати та інтегрувати дані з різних запитів;
- зменшує шум у даних, роблячи семантичну мережу більш читабельною та аналітично придатною.

Наприклад, замість довгого виразу *«організація публічних обговорень з метою залучення громадськості до процесу прийняття рішень»* формується компактний вузол *«Громадська участь»*, який може бути легко використаний у графі.

Формування багаторівневої структури знань

Результатом ієрархічного уточнення є багаторівнева структура знань, яка відображає внутрішню організацію складного поняття. Ця структура має вигляд дерева або спрямованого ациклічного графа, де:

- корінь – вихідне загальне поняття (наприклад, *«Парламентський контроль»*);
- внутрішні вузли – часткові поняття (наприклад, *«Парламентські слухання»*);
- листя – найбільш конкретні чинники або дії (наприклад, *«Експертний обмін»*, *«Рекомендації від експертів»*).

Така структура дозволяє не лише розуміти складові частини явища, а й аналізувати їх взаємозв'язки, оцінювати вагомість окремих елементів та формувати сценарії на основі конкретних механізмів.

Конвертація відповідей у формат CSV для подальшої візуалізації

Для подальшого аналізу та візуалізації всі відповіді LLM конвертуються у структурований формат, зазвичай у вигляді CSV-файлу (Comma-Separated Values). Кожен рядок містить:

- Source (джерело) – вищерівневе поняття;
- Target (ціль) – нижчерівневе поняття;
- Type (тип зв'язку) – наприклад, *«складова»*, *«причина»*, *«наслідок»*.

Наприклад:

- *"Парламентський контроль","Парламентські слухання","Складова"*
- *"Парламентські слухання","Громадська участь","Складова"*
- *"Громадська участь","Залучення громади","Деталізація"*

Такий файл може бути імпортований у програми для візуалізації мереж, такі як Gephi, де формується графічне представлення ієрархії, що дозволяє наочно бачити структуру знань, виявляти найважливіші вузли та аналізувати шляхи впливу.

Таким чином, ієрархічне уточнення понять є потужним інструментом систематизації знань, який перетворює абстрактні категорії в структуровані, формалізовані моделі, придатні для сценарного аналізу. Використання промптів, обмеження термінології та конвертація в структуровані формати роблять цей процес відтворюваним, масштабованим та інтегрованим у інтелектуальну систему парламентського контролю.

5.3. Оптимізація шляхів у мережі: за часом, ризиком, витратами

Побудова причинно-наслідкової мережі та формування сюжетних ланцюжків – це лише початковий етап сценарного аналізу. Для ефективного застосування у практиці парламентського контролю необхідно оцінити та оптимізувати ці ланцюжки за критеріями, які визначають їхню реалістичність, доцільність та пріоритетність. Цей процес, відомий як оптимізація шляхів у мережі, передбачає ранжування альтернативних сценаріїв за такими значущими ознаками, як час реалізації, бюджетні витрати та ступінь ризику. Він дозволяє відібрати найбільш ефективні шляхи досягнення мети, уникнути потенційних загроз та обґрунтовано розподілити ресурси.

Ранжування сюжетних ланцюжків за формальними критеріями

Першим кроком оптимізації є формальне ранжування ланцюжків на основі структурних характеристик мережі. Ці критерії є об'єктивними та легко обчислюваними:

- Довжина шляху: кількість ребер (зв'язків) між початковим і кінцевим вузлом. Коротші шляхи, як правило, вважаються більш ефективними, оскільки містять менше проміжних кроків, що зменшує ймовірність збою.
- Кількість вузлів: пов'язана з довжиною шляху, але також враховує можливість розгалужень. Шляхи з меншою кількістю вузлів вимагають меншої кількості ресурсів для координації.

Ці показники можуть бути обчислені за допомогою стандартних алгоритмів теорії графів, таких як пошук у ширину (BFS) або алгоритм Дейкстри, і використовуватися для попереднього відбору найбільш перспективних маршрутів.

Експертне впорядкування за значущими ознаками

Хоча формальні критерії важливі, вони не враховують реальні умови реалізації. Тому наступним етапом є експертне впорядкування ланцюжків за змістовними ознаками, які вимагають глибокого розуміння предметної області.

Час: від найшвидшого до найдовшого впровадження

Оцінка тривалості реалізації сценарію враховує не лише кількість кроків, а й їхню природу. Наприклад, ланцюжок, що передбачає *«Систематичний моніторинг» → «Експертний аналіз»*, може бути швидшим, ніж *«Парламентські запити» → «Громадська участь»*, оскільки останній вимагає залучення багатьох суб'єктів та публічних консультацій. Для ранжування використовується промпт:

«Експертно впорядкуйте наведені ланцюжки за фактором часу, від найшвидшого до найдовшого»

Відповідь системи дозволяє визначити, які сценарії можуть бути реалізовані оперативно, а які потребують довгострокового планування.

Витрати: від найменш до найбільш коштовного сценарію

Бюджетний фактор є критичним для парламентського контролю. Деякі шляхи можуть вимагати значних фінансових витрат (наприклад, створення нових інститутів, розробка IT-систем), тоді як інші – реалізовуватися за рахунок існуючих механізмів. Промпт для ранжування:

«Експертно впорядкуйте наведені ланцюжки за бюджетним (ціновим) фактором»

Це дозволяє виявити економічно ефективні сценарії, які досягають мети з мінімальними витратами.

Ризик: від менш до більш ризикованого шляху

Кожен сценарій супроводжується певними ризиками: політичними, правовими, інституційними чи соціальними. Наприклад, шлях, що передбачає *«Громадську участь»*, може бути ефективним, але водночас нести ризик негативної публічної реакції або маніпулювання думкою. Промпт:

«Експертно впорядкуйте наведені ланцюжки за фактором ризику»

Таке ранжування допомагає обрати найменш конфліктогенні шляхи, що забезпечує стабільність та легітимність реформ.

Використання промптів для ранжирування

На цей час основним інструментом експертного впорядкування є промпт-кероване програмування. Система ГШІ, така як ChatGPT, виступає як віртуальний експерт, який аналізує кожен ланцюжок з позиції певної

професійної ролі (наприклад, фінансиста, адміністратора, політолога). Відповіді системи ґрунтуються на узагальненому досвіді, отриманому під час навчання, і відображають типові уявлення про вартість, тривалість та ризики різних дій. Хоча ці оцінки є апроксимаційними, вони дозволяють сформувати початкову ієрархію сценаріїв, яка може бути уточнена експертами.

Інтеграція з методами оптимізації для визначення найвагоміших маршрутів

Остаточна оптимізація передбачає інтеграцію формальних і змістовних критеріїв. Для цього можуть використовуватися:

- Методи багатокритеріальної оптимізації (наприклад, метод аналізу ієрархій – AHP), які дозволяють присвоїти ваги різним факторам (час – 0.4, витрати – 0.3, ризик – 0.3) і обчислити інтегральний показник ефективності кожного ланцюжка.
- Аналіз чутливості: оцінка того, як зміна ваги одного критерію впливає на загальний рейтинг.
- Моделювання Монте-Карло: імітація різних сценаріїв з урахуванням невизначеності вхідних даних.

Ці методи дозволяють визначити найвагоміші маршрути – ті, які найкраще відповідають стратегічним цілям парламенту.

Таким чином, оптимізація шляхів у мережі є синтезом формального аналізу та експертної оцінки, який перетворює причинно-наслідкові моделі на практичні інструменти підтримки рішень. Вона дозволяє не лише вибрати найкращий сценарій, а й обґрунтувати цей вибір перед політичними та громадськими акторами.

5.4. Сценарний аналіз реформ: прогнозування наслідків змін у законодавстві

Сценарний аналіз є одним із найпотужніших інструментів стратегічного управління, особливо в умовах високої невизначеності та складності соціально-політичних процесів. У контексті парламентського контролю він дозволяє не лише реагувати на вже виниклі порушення, а й прогнозувати наслідки потенційних рішень, оцінювати альтернативні шляхи розвитку подій та виявляти потенційні ризики ще до їх реалізації. Цей підхід особливо важливий на етапі законотворчості, коли впровадження нових норм може мати довгострокові, глибокі та іноді непередбачувані наслідки для системи державного управління, економіки та суспільства в цілому.

Традиційні методи сценарного аналізу часто обмежені великими часовими та експертними витратами, необхідністю залучення багатьох фахівців та суб'єктивністю інтерпретацій. Сучасний розвиток генеративного штучного інтелекту відкриває нові можливості для автоматизованого формування сценаріїв на основі причинно-наслідкових (каузальних) моделей. Ці моделі дозволяють відобразити складні ланцюжки впливу між подіями, нормами,

станами та рішеннями у вигляді структурованих, візуалізованих та формалізованих мереж. Вони стають основою для систематичного, відтворюваного та масштабованого аналізу, який може бути інтегрований в інтелектуальну систему парламентського контролю.

Моделювання впливу законопроектів на систему управління

Завданням сценарного аналізу є моделювання впливу законопроектів на систему управління. Це означає не просто аналіз тексту норми, а прогнозування того, як зміна законодавства вплине на взаємодію між інституціями, процеси прийняття рішень, механізми контролю та виконання. Наприклад, впровадження нового закону про цифрові послуги може вплинути на:

- взаємодію між центральними та місцевими органами влади;
- ефективність адміністративних процедур;
- рівень цифрової грамотності населення;
- навантаження на інформаційні системи.

Для моделювання такого впливу використовуються причинно-наслідкові мережі, які відображають, як зміна однієї норми може запустити ланцюжок подій у різних сферах. Це дозволяє виявити не лише очікувані, а й непрямі та вторинні наслідки.

Прогнозування соціальних, фінансових та інституційних наслідків

Ефективний сценарний аналіз передбачає комплексне прогнозування наслідків у трьох вимірах:

– Соціальні наслідки: аналіз впливу законопроекту на громадян, їхні права, свободи, рівень довіри до держави. Наприклад, реформа освіти може вплинути на доступність якісної освіти, рівень соціальної мобільності, громадську участь.

– Фінансові наслідки: оцінка бюджетних витрат, ефективності використання коштів, впливу на фінансову стійкість держави. Наприклад, впровадження нової пільгової програми вимагає аналізу її впливу на державний бюджет, джерел фінансування та ризику дефіциту.

– Інституційні наслідки: аналіз змін у структурі влади, розподілі повноважень, ефективності державних органів. Наприклад, створення нового регуляторного агентства може призвести до дублювання функцій або виникнення конфліктів інтересів.

Ці прогнози формуються на основі аналізу каузальних мереж, де кожен вузол представляє певний стан або подію, а ребра – причинно-наслідкові зв’язки між ними.

Виявлення потенційних конфліктів, прогалин, непередбачених ефектів

Однією з найважливіших функцій сценарного аналізу є проактивне виявлення ризиків. Це включає:

- Конфлікти між нормами: виявлення ситуацій, коли нова норма суперечить існуючому законодавству.
- Нормативні прогалини: виявлення сфер, які залишаються нерегульованими після впровадження реформи.
- Непередбачені ефекти: виявлення вторинних наслідків, які не були передбачені законодавцями. Наприклад, реформа податкової системи може призвести до зростання тіньової економіки, навіть якщо це не було метою.

Цей аналіз здійснюється шляхом побудови альтернативних сценаріїв, які моделюють різні шляхи реалізації реформи, включаючи найгірші варіанти розвитку подій.

Формування альтернативних сценаріїв реалізації реформ

Сценарний аналіз ґрунтується на побудові кількох альтернативних сценаріїв, кожен з яких відображає можливий шлях розвитку подій, наприклад:

- Оптимістичний сценарій: реформа реалізується ефективно, з мінімальними витратами та високим рівнем підтримки.
- Песимістичний сценарій: реформа стикається з опором, виникають конфлікти, витрати перевищують очікувані.
- Реалістичний сценарій: частина цілей досягається, але з певними компромісами та затримками.

Кожен сценарій формується як причинно-наслідковий ланцюжок, який може бути візуалізований у Gephi та проаналізований за критеріями часу, витрат, ризику.

Використання каузальних мереж для підтримки стратегічного планування

Каузальні мережі стають основою для стратегічного планування, оскільки вони дозволяють:

- Оцінювати ефективність різних шляхів досягнення мети;
- Обирати найбільш оптимальний сценарій на основі формального та експертного аналізу;
- Прогнозувати наслідки рішень ще до їх прийняття;
- Забезпечувати прозорість та підзвітність процесу прийняття рішень.

Наприклад, як показано в дослідженні, за допомогою ГШІ можна сформувати мережу, яка веде від «Парламентського контролю» до «Несуперечності

документів уряду», виявляючи основні ланки, такі як *«Законодавча узгодженість»*, *«Публічна легітимність»*, *«Систематичний моніторинг»*. Така мережа дозволяє визначити, які механізми є найважливішими для досягнення мети, і зосередити на них зусилля.

Отже, сценарний аналіз реформ на основі каузальних мереж є потужним інструментом проактивного парламентського контролю. Він дозволяє перейти від реактивного аналізу до стратегічного планування, забезпечуючи науково обґрунтовану підтримку прийняття рішень у сфері законотворчості та реалізації реформ.

5.5. Приклади: контроль за податковою системою, моніторинг реалізації реформ

Для демонстрації практичної ефективності запропонованого підходу до сценарного аналізу на основі причинно-наслідкових моделей розглянуто два приклади з реалістичних сфер парламентського контролю: контроль за податковою системою та моніторинг реалізації державних реформ. Ці приклади ілюструють, як за допомогою генеративного штучного інтелекту, промпт-керованого зондування та візуалізації в Gephi можна побудувати структуровані, формалізовані та аналітично придатні моделі, які підтримують стратегічне планування, оцінку альтернативних шляхів та прогнозування наслідків.

Контроль за податковою системою

Однією з найгостріших проблем у сфері фіскальної політики є зростання кількості скарг на діяльність податкових органів, зокрема щодо проведення перевірок, блокування рахунків та застосування санкцій. Традиційний аналіз цих скарг часто обмежується описом окремих випадків. Запропонований підхід дозволяє перейти до системного аналізу причин цього явища.

Аналіз причин зростання скарг на податкові перевірки

За допомогою промпт-керованого зондування формується запит:

«Перелічіть 10 причин зростання скарг на податкові перевірки. Представте у форматі: "причина; зростання скарг"»

Система ГШІ генерує список, наприклад:

«Надмірна кількість перевірок; зростання скарг», *«Недостатня прозорість процедур; зростання скарг»*, *«Відсутність механізму оскарження; зростання скарг»*.

Ці відповіді формалізуються у CSV-файл і стають основою для побудови семантичної мережі.

Побудова ланцюжка від «Моніторингу» до «Зменшення корупції»

На основі виявлених причин формується каузальний ланцюжок, який моделює шлях досягнення мети – зменшення корупційних ризиків. Приклад такого ланцюжка:

Моніторинг скарг → Аналіз причин → Виявлення системних порушень → Зміна регуляторної політики → Зменшення корупції

Кожен вузол цього ланцюжка може бути деталізований через ієрархічне уточнення, наприклад, *«Аналіз причин»* розкладається на *«Контент-аналіз скарг»*, *«Експертна оцінка»*, *«Порівняння з нормами»*.

Оцінка ефективності різних сценаріїв втручання

Формується кілька альтернативних сценаріїв:

Сценарій 1: Впровадження електронного оскарження (низький ризик, низькі витрати, швидка реалізація).

Сценарій 2: Зменшення кількості планових перевірок (середній ризик, помірні витрати, середній час реалізації).

Сценарій 3: Підвищення кваліфікації інспекторів (високий ризик, високі витрати, довгостроковий ефект).

Сценарії ранжуються за критеріями часу, витрат і ризику, що дозволяє парламенту обрати найефективніший шлях.

Моніторинг реалізації реформ

Іншим важливим напрямком є моніторинг реалізації стратегічних реформ, зокрема забезпечення несуперечності документів уряду – важливого завдання парламентського контролю.

Сценарій забезпечення «Несуперечності документів уряду» Початковим станом є «Парламентський контроль», кінцевою метою – «Несуперечність документів уряду». За допомогою двонаправленого пошуку формується кілька ланцюжків:

Ланцюжок 1: Парламентський контроль → Парламентські слухання → Громадська участь → Публічна легітимність → Несуперечність документів уряду.

Ланцюжок 2: Парламентський контроль → Парламентські запити → Громадська участь → Публічна легітимність → Несуперечність документів уряду.

Ланцюжок 3: Парламентський контроль → Аналіз ефективності → Законодавча узгодженість → Несуперечність документів уряду.

Ці ланцюжки відображають різні механізми впливу, від громадської участі до інституційної координації.

Ланцюжки: від «Парламентських запитів» до «Публічної легітимності» Деталізований аналіз показує, як окремі інструменти контролю впливають на легітимність влади. Наприклад:

Парламентські запити → Вимагання звітності → Розкриття інформації → Забезпечення прозорості → Публічна легітимність

Такий ланцюжок демонструє, як формальний механізм контролю трансформується в соціальний ефект.

Візуалізація та інтерпретація результатів

Усі виявлені сутності та зв'язки імпортуються в програму Gephi у форматі CSV. За допомогою алгоритмів компонування (наприклад, ForceAtlas2) будується спрямований граф, де вузли представляють поняття, а ребра – причинно-наслідкові зв'язки. Аналіз мережі дозволяє:

- виявити найвпливовіші вузли (наприклад, *«Громадська участь»* має високий вихідний ступінь);
- визначити найкоротші шляхи до мети;
- виявити найбільш ризикові ланцюжки (наприклад, ті, що залежать від довготривалих консультацій).

Отримана візуалізація стає основою для презентації результатів парламентарям, комітетам та громадським організаціям.

Таким чином, наведені приклади демонструють, як сценарний аналіз на основі каузальних моделей може бути застосований для вирішення реальних завдань парламентського контролю. Вони показують, що запропонований підхід дозволяє перейти від описового аналізу до проактивного, прогнозувального контролю, здатного забезпечити науково обґрунтовану підтримку стратегічних рішень.

Розділ 6. Застосування в практиці парламентського контролю

Парламентський контроль є однією з важливих функцій законодавчої влади, спрямованою на забезпечення відповідальності виконавчої влади, підтримання правової визначеності та захисту інтересів громадян. У сучасних умовах, коли обсяг нормативно-правових актів стрімко зростає, а процеси прийняття рішень стають все більш складними, ефективність традиційних методів контролю стикається з обмеженнями. Вони часто ґрунтуються на ручному аналізі, що робить їх трудомісткими, схильними до суб'єктивізму та неспроможними охопити всі аспекти правових змін.

Сучасні технології генеративного штучного інтелекту відкривають нові можливості для трансформації парламентського контролю. Вони дозволяють автоматизувати аналіз великих масивів тексту, виявляти приховані зв'язки, моделювати наслідки політичних рішень та забезпечувати проактивну підтримку прийняття рішень. Однак для того, щоб ці технології стали інструментом підзвітності, а не джерелом нових ризиків, їх необхідно інтегрувати в структуровану, формалізовану систему контролю.

Цей розділ присвячений конкретним сценаріям застосування методів, розроблених у попередніх розділах, у реальній практиці парламенту. Він демонструє, як комбінація семантичного аналізу, промпт-керованого зондування, побудови каузальних мереж та віртуальних експертів може бути використана для вирішення практичних завдань: від аналізу законопроектів до моніторингу діяльності державних органів. Особлива увага приділяється задачам, які вимагають не лише обробки інформації, а й глибокого розуміння змісту, прогнозування наслідків та виявлення системних проблем.

Наведені приклади базуються на методах, апробованих авторами в дослідженнях, описаних у попередній монографії. Вони ґрунтуються на використанні великих мовних моделей для формування структурованих даних у форматі CSV, які потім імпортуються в інструменти візуалізації, зокрема Gephi, для побудови семантичних та причинно-наслідкових мереж. Цей підхід перетворює ГШІ з інструменту генерації тексту в інженерний інструмент аналізу, здатний забезпечити відтворюваність, прозорість та підзвітність.

6.1. Аналіз суперечностей у законопроектах: технічні помилки, неоднозначності

Однією з найважливіших функцій парламенту є аналіз законопроектів на їхню правову якість, відповідність Конституції та міжнародним зобов'язанням, а також на відсутність внутрішніх суперечностей. Однак на практиці законопроекти часто містять технічні помилки, неоднозначності та відсутність чітких визначень, що призводить до правової невизначеності, інтерпретаційних конфліктів та можливих порушень. Традиційний аналіз цих проблем вимагає значних часових та експертних ресурсів. Запропонований

підхід дозволяє автоматизувати цей процес, зробивши його швидшим, глибшим та відтворюваним.

Виявлення технічних помилок (помилкові посилання, дублювання тексту)

Технічні помилки є однією з найпоширеніших причин суперечностей у законопроектах. Вони можуть виникати на етапі підготовки документа через помилки копіювання, форматування або конвертації. Приклади таких помилок:

- Помилкові посилання на неіснуючі статті або закони;
- Дублювання тексту з інших документів без урахування контексту;
- Проблеми при скануванні або конвертації (наприклад, OCR-помилки);
- Неправильне копіювання норм під час редагування.

Для виявлення цих помилок використовується промпт:

«Перелічіть 10 причин технічних помилок, що ведуть до суперечності визначень у законодавчих документах. Представте у форматі: "причина; Технічні помилки"»

Відповідь системи:

Невірне форматування; Технічні помилки

Друкарські помилки; Технічні помилки

Неправильне копіювання; Технічні помилки

Помилки при скануванні; Технічні помилки

Проблеми із зашифруванням; Технічні помилки

Помилки автоматичного перекладу; Технічні помилки

Несумісність платформ; Технічні помилки

Проблеми із завантаженням; Технічні помилки

Помилки конвертації; Технічні помилки

Помилки програмування; Технічні помилки

Цей список дозволяє систематично перевірити законопроект на наявність цих помилок.

Аналіз неоднозначностей та полісемії термінів

Неоднозначність (полісемія) – це ситуація, коли один і той самий термін має кілька значень, що призводить до різних тлумачень. Наприклад, термін «система» може означати як організаційну структуру, так і технічний комплекс. У правовому тексті це може призвести до конфліктів.

Для виявлення цього явища використовується промпт:

«Перелічіть 10 причин неоднозначності, що веде до суперечності визначень у законодавчих документах. Представте у форматі: "Причина; Неоднозначність"»

Відповідь:

Двозначні терміни; Неоднозначність

Множинні тлумачення; Неоднозначність

Зміна значень; Неоднозначність

Неточні визначення; Неоднозначність

Контекстна залежність; Неоднозначність

Синонімія; Неоднозначність

Амбігвітет; Неоднозначність

Міжгалузеві відмінності; Неоднозначність

Мовна еволюція; Неоднозначність

Недостатність контексту; Неоднозначність

Аналіз цих причин дозволяє зосередитися на термінах, які найбільш схильні до різних тлумачень.

Відсутність чітких визначень ключових понять

Однією з найпоширеніших причин суперечностей є відсутність визначень ключових понять. Наприклад, у багатьох законопроектах використовуються терміни *«штучний інтелект»*, *«цифровий двійник»*, *«блокчейн»* без чітких дефініцій, що призводить до правової невизначеності.

Промпт для виявлення цієї проблеми:

«Перелічіть 10 причин відсутності визначень, що веде до суперечності визначень у законодавчих документах. Представте у форматі: "Причина; Відсутність визначень"»

Відповідь:

Лакуни у тексті; Відсутність визначень

Неузгодженість термінів; Відсутність визначень

Невизначені поняття; Відсутність визначень

Відсутність тлумачення; Відсутність визначень

Полісемія понять; Відсутність визначень

Відсутність стандартів; Відсутність визначень

Недостатня проробка; Відсутність визначень

Непередбаченість явища; Відсутність визначень

Міжгалузеві розбіжності; Відсутність визначень

Відсутність норм; Відсутність визначень

Ці відповіді вказують на системні недоліки в процесі законотворчості.

6.2. Контроль за дотриманням міжнародних зобов'язань

Одним із важливих напрямків парламентського контролю є забезпечення виконання Україною міжнародних зобов'язань, особливо в контексті інтеграції до Європейського Союзу, підписання угод про асоціацію, дотримання положень міжнародного гуманітарного права, угод ООН та інших багатосторонніх договорів. Ефективний контроль за виконанням цих зобов'язань вимагає систематичного аналізу національного законодавства на відповідність міжнародним стандартам, виявлення прогалин у транспонуванні норм та прогнозування наслідків їхнього недотримання. Традиційні методи такого контролю ґрунтуються на праці експертів, що робить їх трудомісткими, суб'єктивними та обмеженими в часі. Застосування технологій генеративного штучного інтелекту та семантичного інжинірингу дозволяє автоматизувати цей процес, забезпечуючи його масштабованість, об'єктивність та відтворюваність.

Порівняння національного законодавства з міжнародними угодами

Фундаментальним етапом контролю є порівняння тексту національних нормативно-правових актів (законів, підзаконних актів) з положеннями міжнародних угод. Це включає:

– аналіз відповідності термінології (наприклад, чи використовується термін "дискримінація" у тій самій юридичній площині, що й у Конвенції ООН);

– перевірку на повноту відображення зобов'язань (чи всі положення директиви ЄС втілені в національне право);

– виявлення суперечностей (наприклад, коли національна норма обмежує права, які гарантуються міжнародним договором).

Цей процес може бути автоматизований за допомогою ГШІ, який порівнює структуру, зміст і наслідки норм, виявляючи як явні, так і приховані невідповідності.

Виявлення прогалин у транспонуванні директив ЄС

Особливо важливим є контроль за транспонуванням директив Європейського Союзу, оскільки це є одним із критеріїв вступу України до ЄС. Прогалини (лакуни) виникають, коли:

– певні положення директиви не були втілені в національне законодавство;

– транспонування виконане не повністю або з відхиленням від оригіналу;

– терміни для впровадження директиви прострочені.

Для виявлення таких прогалин використовується структурований підхід:

– Формується список усіх директив, які підлягають транспонуванню.

– Для кожної директиви визначається її зміст, основні положення та терміни виконання.

– За допомогою ГШІ аналізуються національні акти на наявність відповідних норм.

– Виявлені прогалини фіксуються у форматі: «Директива 2023/XX/ЄС, стаття 5 – не транспонована».

Цей аналіз може бути систематизований у вигляді реєстру, який оновлюється автоматично при зміні законодавства.

Використання RAG для інтеграції з текстами міжнародних документів

Для підвищення точності аналізу використовується технологія RAG (Retrieval-Augmented Generation) – розширення генерації за рахунок попереднього витягування інформації з авторитетних джерел. У контексті контролю за міжнародними зобов'язаннями це означає:

– створення бази даних з офіційними текстами міжнародних угод (наприклад, EUR-Lex, UN Treaty Collection);

– автоматичне витягування релевантних статей, пунктів, додатків при аналізі конкретного законопроекту;

– додавання цих фрагментів до промпту як контексту.

Наприклад, промпт може мати вигляд:

«Порівняй статтю 12 законопроекту №XXX зі Стандартом вільної торгівлі між Україною та ЄС (DCFTA), розділ "Технічні регламенти", пункт 3.2. Вияви наявність прогалин або суперечностей. Відповідь надай у форматі: "Наявність/відсутність відповідності; Опис"»

Це забезпечує, що висновки ГШІ ґрунтуються не на внутрішніх знаннях моделі, а на актуальних, офіційних джерелах, що підвищує їхню достовірність та підзвітність.

Побудова мережі «національна норма – міжнародна норма»

Результати порівняння формалізуються у вигляді двонаправленої семантичної мережі, де вузли представляють окремі норми, а ребра – відношення відповідності, часткової відповідності або суперечності. Така мережа має вигляд:

Національна норма A → Відповідає → Міжнародна норма X

Національна норма B → Частково відповідає → Міжнародна норма Y

Національна норма C → Суперечить → Міжнародна норма Z

Ця мережа:

- дозволяє наочно бачити стан виконання зобов'язань;
- виявляє «сліпі плями» – сфери, де відсутні національні норми;
- показує, які міжнародні стандарти найбільш повно втілені, а які ігноруються.

Мережа може бути візуалізована в Gephi, де кольором позначається тип відповідності (зелений – відповідає, жовтий – частково, червоний – суперечить), а розміром вузла – кількість пов'язаних норм.

Прогнозування наслідків невиконання зобов'язань

Контроль не обмежується констатуванням фактів – він має бути проактивним. Тому важливим елементом є прогнозування наслідків невиконання зобов'язань. Це може бути зроблено шляхом побудови причинно-наслідкових ланцюжків:

Невиконання директиви ЄС щодо захисту даних → Порушення GDPR → Скарги громадян → Штрафи від ЄС → Втрати фінансування → Затримка інтеграції

Такі ланцюжки формуються за допомогою промптів:

«Перелічіть 10 можливих наслідків невиконання Україною положень Директиви 2023/XX/ЄС про кібербезпеку. Представте у форматі: "наслідок; Невиконання директиви"»

Отримані відповіді конвертуються в CSV і інтегруються в загальну мережу, що дозволяє оцінювати рівень ризику для держави.

Таким чином, контроль за дотриманням міжнародних зобов'язань на основі ГШІ та семантичного інжинірингу стає потужним інструментом стратегічного управління. Він дозволяє не лише виявляти прогалини, а й прогнозувати їхні наслідки, оцінювати пріоритетність виконання зобов'язань та підтримувати процес європейської інтеграції на науково обґрунтованій основі. Це робить парламентський контроль не реактивним, а проактивним механізмом забезпечення національних інтересів в умовах глобальної взаємозалежності.

6.3. Обробка масових скарг громадян: від тексту до мережі порушень

Одним із найважливіших джерел інформації для парламентського контролю є масові скарги громадян, які відображають реальні проблеми в діяльності державних органів, місцевого самоврядування та підприємств. Ці скарги, зібрані через офіційні канали (наприклад, сайт Верховної Ради, портали електронного уряду, роботу омбудсменів), утворюють великий масив неструктурованих текстових даних, аналіз яких традиційно є трудомістким, суб'єктивним та обмеженим у масштабах. Сучасні технології генеративного штучного інтелекту та семантичного інжинірингу відкривають можливість автоматизованої, системної та масштабованої обробки цих даних,

перетворюючи масив текстів на аналітичну модель у вигляді мережі порушень. Цей підхід дозволяє не лише виявляти окремі випадки, а й визначати системні проблеми, «гарячі точки» та причинно-наслідкові механізми порушень.

Автоматизований аналіз текстів скарг за допомогою LLM

Важливим етапом здійснення парламентського контролю є автоматична обробка текстів скарг за допомогою великих мовних моделей. Ці моделі виступають як інструмент інтелектуального аналізу природної мови, здатний обробити тисячі текстів за короткий час. Процес здійснюється через промпт-кероване програмування, коли LLM отримує чітко сформульоване завдання. Наприклад:

«Виділіть у тексті всі пари сутностей, пов’язані з податковими перевірками. Представте у форматі: "Суб’єкт; Об’єкт"»

Або:

«Виділіть із тексту 20 пар найбільш пов’язаних сутностей. Кожна сутність має бути описана не більше ніж трьома словами. Формат: "сутність 1; сутність 2"»

Такі промпти активують в LLM функції розпізнавання іменованих об’єктів (NER) та виявлення відношень, що дозволяє витягувати структуровану інформацію з неструктурованого тексту.

Виділення ключових понять та побудова семантичних пар

На основі аналізу текстів формується список ключових понять (сутностей) реального світу, які найчастіше згадуються у скаргах. Це можуть бути:

– органи влади: «податкова служба», «місцева адміністрація», «поліція»;

– дії: «блокування рахунку», «проведення перевірки», «відмова у видачі документа»;

– стани: «затримка виплати», «корупція», «порушення прав».

Далі виявляються семантичні пари – пари понять, що мають смисловий зв’язок. Наприклад:

«податкова служба; блокування рахунку»

«бізнес; скарга»

«перевірка; корупція»

Ці пари є основними ребрами майбутньої мережі, кожне з яких відображає конкретний тип порушення.

Класифікація скарг за типами, регіонами, сферами

Для систематизації аналізу скарги класифікуються за кількома вимірами:

– За типом порушення: адміністративне, фінансове, корупційне, процесуальне.

– За сферою діяльності: податкова, митна, охорона здоров'я, освіта, ЖКГ.

– За регіоном: область, район, місто.

– За суб'єктом порушення: державний орган, посадова особа, приватна компанія.

Ця класифікація дозволяє:

- виявити географічні осередки проблем (наприклад, масові скарги на блокування накладних у певній області);
- виявити сфери з найвищим рівнем порушень;
- проводити порівняльний аналіз між різними регіонами чи органами.

Класифікація також може бути автоматизована за допомогою промптів:

«Визначте тип, сферу та регіон для кожної з наведених скарг. Представте у форматі: "тип; сфера; регіон"»

Формування мережі порушень та виявлення «гарячих точок»

На основі виявлених сутностей і зв'язків будується семантична мережа порушень, де:

– вузли – це сутності (наприклад, «податкова служба», «скарга», «бізнес»);

– ребра – це семантичні пари, що вказують на тип відносин (наприклад, «застосовує», «порушує», «викликає»).

Ця мережа імпортується в програму Gephi для візуалізації. Аналіз мережі дозволяє:

– виявити «гарячі точки» – органи або регіони, які мають найвищий ступінь (degree) у мережі, тобто на які найбільше скарг;

– виявити ключові ланки у ланцюжках порушень (наприклад, «блокування накладних» як найчастіша дія);

– виявити кластери – групи тісно пов'язаних скарг, що вказують на системні проблеми.

Наприклад, якщо *«податкова служба»* має високий ступінь і є центром кластера, пов'язаного з *«блокуванням рахунків»* та *«корупцією»*, це вказує на необхідність цільового контролю.

Побудова причинно-наслідкових ланцюжків від скарги до порушення

Найглибшим рівнем аналізу є побудова причинно-наслідкових ланцюжків, які відображають, як окрема скарга виникає внаслідок певного порушення. Це досягається через двонаправлений пошук:

- Від скарги до порушення: виявлення, яка дія або бездіяльність призвела до скарги.
- Від порушення до скарги: виявлення, які норми були порушені та які механізми дозволу сприяли виникненню проблеми.

Наприклад, ланцюжок може мати вигляд:

«Податкова перевірка» → «Недотримання процедури» → «Порушення прав бізнесу» → «Скарга» → «Парламентський запит»

Такі ланцюжки формуються за допомогою промптів:

«Виділіть у тексті всі причини скарги на блокування податкових накладних. Представте у форматі: "причина; скарга"»

Отримані ланцюжки конвертуються у CSV і інтегруються в загальну мережу, що дозволяє проводити сценарний аналіз та прогнозування наслідків.

Таким чином, обробка масових скарг громадян за допомогою ГШІ та семантичного інжинірингу перетворюється з пасивного збору даних на активний інструмент проактивного контролю. Він дозволяє не лише реагувати на існуючі проблеми, а й виявляти системні недоліки, прогнозувати ризики та формувати рекомендації для вдосконалення діяльності державних органів.

6.4. Аналіз діяльності державних органів: податкова, митна, правоохоронна сфери

Ефективність парламентського контролю безпосередньо залежить від його здатності системно аналізувати діяльність державних органів, які виконують функції регулювання, нагляду та забезпечення правопорядку. Серед найбільш критичних у цьому контексті є податкова, митна та правоохоронна сфери, діяльність яких безпосередньо впливає на права громадян, стан економіки та рівень довіри до держави.

Традиційні методи аналізу їхньої діяльності, засновані на ручному вивченні звітів та ухвалення окремих рішень, часто є реактивними, фрагментарними та обмеженими в часі. Застосування технологій генеративного штучного інтелекту та семантичного інжинірингу дозволяє трансформувати цей процес у проактивний, системний та прогнозуючий аналіз, здатний виявляти не лише окремі порушення, а й системні недоліки у функціонуванні державних служб.

Моніторинг звітів державних органів (ДПС, ДМС, правоохоронці)

Першим етапом аналізу є систематичне збирання та обробка офіційних звітів державних органів, зокрема:

– Державної податкової служби (ДПС): звіти про кількість проведених перевірок, блокувань рахунків, повернень податків, статистику скарг;

– Державної митної служби (ДМС): дані про обсяги митного оформлення, виявлення порушень, блокування товарів, роботу митних постів;

– Правоохоронних органів (Національна поліція, ДБР, СБУ): звіти про кількість відкритих кримінальних проваджень, результати розслідувань, виконання рішень судів.

Ці звіти, як правило, представлені у форматі PDF або на веб-сайтах, що ускладнює їх автоматичний аналіз. Для їх обробки використовується автоматизований семантичний аналіз за допомогою ГШІ. Наприклад, промпт:

«Виділіть із звіту ДПС за 2023 рік всі цифри, пов'язані з кількістю перевірок, блокувань рахунків та скарг. Представте у форматі: "показник; значення"»

Це дозволяє швидко отримати структуровані дані для подальшого порівняння та аналізу.

Аналіз статистики перевірок, блокувань, розслідувань

На основі витягнутих даних проводиться статистичний аналіз основних показників діяльності:

– У податковій сфері: аналізується співвідношення кількості планових та позапланових перевірок, частка блокувань рахунків та накладних, терміни розгляду скарг.

– У митній сфері: оцінюється швидкість оформлення вантажів, кількість виявлених порушень, рівень корупційних ризиків.

– У правоохоронній сфері: аналізується ефективність розслідувань (відсоток закритих справ, терміни розслідування), прозорість діяльності, виконання рішень прокуратури.

Цей аналіз дозволяє виявити аномалії – наприклад, різке зростання кількості перевірок у певному регіоні або значне відхилення від середньої ефективності розслідувань.

Виявлення кореляцій між діяльністю органів та кількістю скарг

Важливим елементом аналізу є встановлення кореляцій між показниками діяльності органів та кількістю скарг громадян. Наприклад:

– Чи зростає кількість скарг на блокування накладних разом із збільшенням кількості перевірок?

– Чи є зв'язок між тривалістю розслідувань та рівнем недовіри до правоохоронців?

Для цього використовується промпт:

«Перелічіть 10 причин, які можуть пояснювати зростання кількості скарг на діяльність ДПС. Представте у форматі: "причина; кількість скарг"»

Відповідь системи:

Надмірна кількість перевірок; кількість скарг

Недостатня прозорість процедур; кількість скарг

Відсутність механізму оскарження; кількість скарг ...

Ці дані конвертуються у CSV-файл і інтегруються в загальну мережу, що дозволяє візуалізувати зв'язки між діяльністю органу та реакцією громадян.

Побудова каузальних моделей для оцінки ефективності

На основі виявлених кореляцій будується причинно-наслідкова (каузальна) модель, яка відображає, як діяльність органу впливає на соціальні, економічні та правові наслідки. Наприклад, ланцюжок у податковій сфері може мати вигляд:

Зростання кількості перевірок → Збільшення тиску на бізнес → Зростання кількості скарг → Зниження довіри до держави → Зменшення інвестицій

Такі моделі формуються за допомогою двонаправленого пошуку: від проблеми (наприклад, *«велика кількість скарг»*) та від мети (наприклад, *«зменшення корупції»*). Коли дві часткові мережі об'єднуються, вони утворюють повний ланцюжок, який може бути проаналізований на предмет ефективності.

Формування сценаріїв оптимізації роботи державних служб

На основі каузальних моделей формується кілька альтернативних сценаріїв для оптимізації діяльності державних служб, наприклад:

– Сценарій 1: Запровадження електронного оскарження рішень ДПС → Зменшення кількості скарг → Підвищення правової захищеності.

– Сценарій 2: Обмеження кількості планових перевірок → Зниження тиску на бізнес → Зменшення скарг.

– Сценарій 3: Підвищення кваліфікації інспекторів → Зменшення помилок → Зменшення скарг.

Кожен сценарій оцінюється за такими критеріями, як час реалізації, витрати, ризик, очікуваний ефект.

Це дозволяє парламенту обрати найефективніший шлях для втручання.

Таким чином, аналіз діяльності державних органів за допомогою ГШІ та семантичного інжинірингу стає потужним інструментом проактивного контролю. Він дозволяє перейти від реакції на існуючі проблеми до системного розуміння їхніх причин та прогнозування наслідків політичних рішень. Це робить парламентський контроль не лише інструментом фіксації помилок, а стратегічним механізмом управління, здатним забезпечити підвищення ефективності, прозорості та підзвітності державних служб.

6.5. Контроль за законодавством щодо ШІ: інтеграція конфліктології та семантичного аналізу

Розвиток технологій штучного інтелекту викликає не лише технічні, а й фундаментальні правові, етичні та інституційні виклики, які потребують створення нових правових рамок. У відповідь на ці виклики парламенти різних країн, зокрема Європейський Союз (зокрема, AI Act), США та Україна, розробляють проєкти законів, спрямовані на регулювання розробки, впровадження та використання ШІ. Однак сам процес створення такого законодавства є високоризикованим, оскільки воно стосується нових, швидко змінних явищ, які ще не мають чітко визначених ознак, а також втручається в дію вже існуючих правових систем. Тому контроль за законодавством щодо ШІ стає особливо важливим завданням парламентського контролю.

Для ефективного виконання цього завдання недостатньо традиційного юридичного аналізу. Необхідно застосування інтегрованого підходу, який об'єднує дві наукові парадигми: конфліктологію ШІ – для виявлення потенційних конфліктів, і семантичний аналіз – для глибокого розуміння структури та змісту правових норм. Цей синтез дозволяє не лише виявляти помилки, а й прогнозувати наслідки, моделювати сценарії та забезпечувати системну цілісність нової правової галузі.

Аналіз проєктів законів про ШІ на наявність визначень

Однією з найголовніших проблем у законопроектах щодо ШІ є відсутність чітких, уніфікованих визначень ключових понять. Терміни, такі як *«штучний інтелект»*, *«генеративний ШІ»*, *«автономна система»*, *«високоризикова система»*, *«тренувальні дані»*, часто використовуються без чіткої дефініції, що призводить до неоднозначності, правової невизначеності та різних тлумачень. Це створює ґрунт для майбутніх конфліктів у судах, регуляторній практиці та міжінституційній взаємодії.

Для систематичного аналізу використовуються промпти, спрямовані на виявлення цієї проблеми. Наприклад:

«Перелічіть 10 причин відсутності визначень, що веде до суперечності визначень у законопроектах з ШІ. Представте у форматі: "причина; Відсутність визначень"»

Відповідь системи:

Лакуни у тексті; Відсутність визначень

Неузгодженість термінів; Відсутність визначень

Невизначені поняття; Відсутність визначень

Відсутність тлумачення; Відсутність визначень

Полісемія понять; Відсутність визначень ...

Аналіз цих причин дозволяє зробити висновок, що відсутність визначень – це не лише технічна помилка, а системна проблема, пов'язана з відсутністю єдиного регулятора, міжгалузевими розбіжностями та швидкістю розвитку технологій.

Виявлення конфліктів з іншими галузями права

Законодавство щодо ШІ не існує в ізоляції. Воно взаємодіє з цивільним, кримінальним, адміністративним, податковим, трудовим, медичним та іншими галузями права. Через це виникає ризик нормативних колізій, коли нова норма суперечить вже існуючим, наприклад:

- Якщо ШІ приймає рішення про звільнення працівника, чи порушується це трудове законодавство?
- Якщо генеративний ШІ створює текст, хто є автором – порушення авторського права?
- Чи може система ШІ бути суб'єктом кримінальної відповідальності?

Для виявлення таких конфліктів використовується промпт:

«Перелічіть 10 причин протиріччя норм, що веде до суперечності визначень у законопроектах з ШІ. Представте у форматі: "причина; Протиріччя норм"»

Відповідь:

Різні правові системи; Протиріччя норм

Зміна законодавства; Протиріччя норм

Технічні помилки; Протиріччя норм

Відсутність узгодження; Протиріччя норм

Конфлікт інтересів; Протиріччя норм ...

Цей аналіз показує, що найбільш поширеними причинами колізій є відсутність узгодження між відомствами та різний темп оновлення різних галузей права.

Прогнозування соціальних, правових та етичних наслідків

Окрім виявлення поточних проблем, важливим є прогнозування майбутніх наслідків впровадження законодавства щодо ШІ. Це робиться шляхом побудови причинно-наслідкових ланцюжків:

«Генеративний ШІ» → «Створення фейків» → «Дезінформація» → «Зниження довіри до ЗМІ» → «Політична нестабільність»

Або:

«Автономна система» → «Відсутність відповідальності» → «Правова безкарність» → «Зниження довіри до держави»

Такі ланцюжки формуються за допомогою промптів:

«Перелічіть 10 можливих наслідків впровадження закону про ШІ. Представте у форматі: "наслідок; Впровадження закону"»

Отримані відповіді дозволяють проводити сценарний аналіз, оцінювати рівень ризику та формувати рекомендації для мінімізації негативних ефектів.

Таким чином, контроль за законодавством щодо ШІ на основі інтеграції конфліктології та семантичного аналізу стає потужним інструментом проактивного парламентського контролю. Він дозволяє не лише виявляти конфлікти, а й розуміти їхні глибокі причини, прогнозувати наслідки та забезпечувати системну цілісність правової системи в умовах технологічної трансформації.

6.6 Застосування фреймворку безкодового програмування при вирішенні деяких завдань парламентського контролю

Для вирішення завдань парламентського контролю використано фреймворк безкодового програмування на основі промпт-інжинірингу, описаний у [9].

Основна застосування цього фреймворку полягає у представленні промптів як математично формалізованих аналогів базових конструкцій мов програмування (умови, цикли, функції). Це перетворює довільні текстові запити на структуровані логічні примітиви, які можна систематично комбінувати, оптимізувати та аналізувати. Наприклад, умова (If-Else) набуває форми математичної функції з вхідними параметрами та вихідними діями, цикл (For-Loop) моделюється як оператор багаторазової обробки, а функція (Абстракція) визначається як процес перетворення даних із параметрами.

Такий підхід дозволяє створювати складні логічні системи через текстові інструкції, що узгоджується з парадигмою "програмування за прикладом" (PBE), але з більшою точністю та контролем.

Запропонована "мова промптів" відрізняється від традиційних предметно-орієнтованих мов гнучкістю та відсутністю жорсткого синтаксису. Вона має властивості модульності, параметризації та композиційності. У межах цієї мови кожен промпт виконує атомарне завдання (наприклад, «витягти терміни» чи «побудувати зв’язки»), змінні, взяті у фігурні дужки, дозволяють адаптувати логіку без зміни основної структури, а промпти можна об'єднувати в ланцюжки чи графи завдань, аналогічно функціям у програмах.

Це також зменшує поріг входу до програмування для користувачів, оскільки не вимагає навичок написання коду, водночас забезпечуючи гнучкість для створення складних систем.

Опишемо фреймворк детальніше. Суть запропонованого підходу для створення систем полягає у представленні промптів як аналогів програмних конструкцій (умовні оператори, цикли, функції) шляхом математичної формалізації їх логіки та взаємодій. Для формування промптів застосовуються такі основні примітиви («цеглинки»), такі як «Умова»,

«Цикл» і «Функція», а також методи композиції цих примітивів для побудови складних систем, зокрема семантичних мереж.

Множина задач парламентського контролю

На основі аналізу потреб парламентських комітетів, профільних груп та експертів виявлено широкий спектр задач, які можуть бути автоматизовані або підтримані через застосування фреймворку безкодового програмування. Серед них:

- Виявлення ризиків і суперечностей у текстах законопроектів.
- Генерація стислих резюме для депутатів.
- Аналіз звітів виконавчої влади на відповідність чинному законодавству.
- Формування запитів до органів влади для уточнення інформації або звітності.
- Виявлення можливих лобістських інтересів у законодавчих ініціативах.
- Пошук формулювань (повторюваних), що вказують на лобіювання певних інтересів.
- Аналіз громадської думки щодо певних законодавчих ініціатив.
- Автоматичний пошук рішень, які використовуються в інших країнах.
- Генерація рекомендацій на основі найкращих світових практик.
- Контроль за виконанням доручень парламенту.
- Аналіз бюджетних видатків на відповідність затвердженому плану.
- Тематичний аналіз парламентських запитань.
- Прогнозування наслідків прийняття рішень.
- Пошук аналогів у міжнародному досвіді.
- Виявлення актуальності нормативно-правового регулювання.
- Перевірка на відповідність міжнародним стандартам.
- Виявлення корупційних загроз у звітах.
- Формування експертних полягань.
- Моніторинг реалізації держпрограм.
- Оцінка ефективності законодавчих змін.

Ці задачі можуть бути реалізовані через промпти до систем штучного інтелекту, формалізовані через примітиви фреймворку, що забезпечує повторність, стандартизацію та масштабованість аналітичних процесів.

Приклади використання фреймворку безкодового програмування

Приклад 1. Автоматичне виявлення ризиків, суперечностей і корупційних загроз у тексті законопроекту

Для вирішення цієї задачм складається первинний запит (промпт), який має вигляд:

«Ви – експерт із законодавства та боротьби з корупцією. Ваше завдання – проаналізувати текст законопроекту на наявність:

Можливих ризиків (економічних, соціальних, правових).

Суперечностей із чинним законодавством.

Потенційних корупційних загроз (наприклад, надмірно вигідні умови для окремих осіб чи груп, лобістські формулювання тощо).

Проаналізуйте текст законопроекту [вставте текст] і надайте:

список знайдених ризиків із поясненнями;

перелік суперечностей із чинним законодавством (якщо є, зазначте, з якими саме законами);

Опис можливих корупційних загроз із прикладами фрагментів тексту.

Висновки подайте у форматі:

Ризики: [список]

Суперечності: [список]

Корупційні загрози: [список із поясненнями]

Ось текст законопроекту: [Тест]»

Складений промпт вже може виконуватись у середовищі ВММ, але відповіді, що він буде давати мають слабко структурований, нечіткий вигляд, не зрозуміло, в якій мірі інформація первинного документа враховано. Для перетворення цього документа у структурований промпт, до ВММ завантажується текст фреймворку безкодового програмування:

У відповідності із наведеним фреймворком перетвори промпт для автоматичного виявлення ризиків, суперечностей чи корупційних загроз у тексті законопроекту. Ось первинний промпт: [промпт]

У відповідності з фреймворком безкодового промпт-інжинірингу, пропонуються наступні кроки для реалізації задачі через формальні примітиви:

1. Використання примітива «Функція» для аналізу ризиків

Формалізація:

Нехай:

- F_risk: X → Y – функція для виявлення ризиків.
- X – текст законопроекту.
- параметр = {"тип_ризиків": ["економічні", "соціальні", "правові"]}.
- F_risk(X, параметр) повертає список ризиків із поясненнями.

Промпт-функція:

"Виділи з тексту [X] всі можливі ризики типів: {тип_ризиків}.

Для кожного ризику:

- опиши його сутність;
- поясни потенційні наслідки;
- наведи цитату з тексту законопроекту, що його викликає."

2. Використання примітива "Функція" для аналізу суперечностей

Формалізація:

Нехай:

- F_conflict: X → Z – функція для виявлення суперечностей із чинним законодавством.
- параметр = {"джерела_законодавства": [список_законів]}.
- F_conflict(X, параметр) повертає перелік суперечностей.

Промпт-функція:

"Перевір текст [X] на відповідність законам: {джерела_законодавства}.

Для кожної суперечності:

- зазнач назву закону;
- опиши конфліктну норму у законопроекті;
- наведи цитату з тексту."

3. Використання примітива "Функція" для аналізу корупційних загроз

Формалізація:
Нехай:

- F_corruption: X → W – функція для виявлення корупційних загроз.
- параметр = {"ознаки_корупції": ["надмірні_вигоди", "лобістські_формулювання", "відсутність_контролю"]}.
- F_corruption(X, параметр) повертає список загроз із прикладами.

Промпт-функція:

"Аналізуй текст [X] на наявність ознак корупції: {ознаки_корупції}.

Для кожної ознаки:

- опиши, як вона проявляється у тексті;
- наведи цитату з законопроекту;
- поясни, хто може скористатися перевагами."

4. Композиція примітивів у систему

Формалізація:

Система S реалізується через послідовне застосування трьох функцій:

$S(X) = F\_risk(X, \{"тип\_ризиків": [...]\}) \oplus F\_conflict(X, \{"джерела\_законодавства": [...]\}) \oplus F\_corruption(X, \{"ознаки\_корупції": [...]\})$

де $\oplus$ – операція об'єднання результатів у структурований вивід.

У результаті формальний промпт для ВММ має вигляд:

Ви експерт із безкодового аналізу текстів. Виконайте завдання через три кроки:

Крок 1. Аналіз ризиків

Використовуй функцію F_risk для виявлення економічних, соціальних та правових ризиків.

Крок 2. Аналіз суперечностей

Застосуй F_conflict для перевірки відповідності законам: [перелік_законів].

Крок 3. Аналіз корупційних загроз

Виконай F_corruption для пошуку надмірних вигод, лобістських формулювань тощо.

Результат подай у форматі:

1. Ризики: [список із поясненнями та цитатами]

2. Суперечності: [список із назвами законів та цитатами]

3. Корупційні загрози: [список із прикладами з тексту]

Функції:

1. F_risk: $X \rightarrow Y$ – функція для виявлення ризиків.

X – текст законопроекту.

параметр = {"тип_ризиків": ["економічні", "соціальні", "правові"]}.

F_risk(X, параметр) повертає список ризиків із поясненнями.

Промпт-функція:

"Виділи з тексту [X] всі можливі ризики типів: {тип_ризиків}.

Для кожного ризику:

- опиши його сутність;

- поясни потенційні наслідки;

- наведи цитату з тексту законопроекту, що його викликає."

2. F_conflict: X → Z – функція для виявлення суперечностей із чинним законодавством.

параметр = {"джерела_законодавства": [список_законів]}.

F_conflict(X, параметр) повертає перелік суперечностей.

Промпт-функція:

"Перевір текст [X] на відповідність законам: {джерела_законодавства}.

Для кожної суперечності:

- зазнач назву закону;

- опиши конфліктну норму у законопроекті;

- наведи цитату з тексту."

3. F_corruption: X → W – функція для виявлення корупційних загроз.

параметр = {"ознаки_корупції": ["надмірні_вигоди", "лобістські_формулювання", "відсутність_контролю"]}.

F_corruption(X, параметр) повертає список загроз із прикладами.

Промпт-функція:

"Аналізуй текст [X] на наявність ознак корупції: {ознаки_корупції}.

Для кожної ознаки:

- опиши, як вона проявляється у тексті;

- наведи цитату з законопроекту;

- поясни, хто може скористатися перевагами."

Нижче наведено приклад застосування сформованого структурованого промпту до Закону України «Про державне регулювання у сфері комунальних послуг» № 3610-VI від 07.07.2011. Відомості Верховної Ради України, 2010, № 49, ст. 571.

1. Ризики:

Економічні ризики:

Нестабільність тарифів на комунальні послуги: Встановлення тарифів,

нижчих за економічно обґрунтовані витрати, може призвести до фінансової неплатоспроможності підприємств.

Цитата: "Встановлення тарифів на комунальні послуги, нижчих за рівень економічно обґрунтованих витрат на їх виробництво, не допускається..." (ст. 10, ч. 4).

Перехресне субсидіювання: Можливе використання коштів з регульованих ринків для підтримки інших напрямів діяльності підприємств, що порушує конкуренцію.

Цитата: "Контроль за недопущенням перехресного субсидіювання під час провадження господарської діяльності суб'єктами природних монополій..." (ст. 6, п. 8).

Соціальні ризики:

Обмежений доступ до послуг: Невиконання суб'єктами монополій вимог щодо якості послуг може спричинити соціальну напруженість.

Цитата: "Забезпечення прогнозованості цінової і тарифної політики... стимулювання підвищення їх якості..." (ст. 3, п. 4).

Правові ризики:

Конфлікти між регуляторами: Розмиття компетенцій між Національною комісією та місцевими органами влади.

Цитата: "Національна комісія здійснює контроль за дотриманням законодавства Радою міністрів Автономної Республіки Крим..." (ст. 15, ч. 2).

2. Суперечності:

Порушення Закону України "Про природні монополії":

Невідповідність у визначенні монополій: У Законі про природні монополії вказано, що монополії регулюються окремим органом, але в тексті досліджуваного закону ці повноваження передані Національній комісії.

Цитата: "Органом державного регулювання діяльності суб'єктів природних монополій є національна комісія..." (ст. 2, ч. 1; порівняйте з Законом "Про природні монополії", ст. 4).

Порушення Закону України "Про місцеве самоврядування":

Обмеження права місцевих рад встановлювати тарифи:

Уточнення, що тарифи на водопостачання та теплопостачання встановлює Комісія, суперечить правам місцевого самоврядування.

Цитата: "Тарифи на послуги централізованого водопостачання... встановлюються національною комісією..." (ст. 19, п. 5.3; порівняйте з Законом "Про місцеве самоврядування", ст. 28).

3. Корупційні загрози:

Надмірні вигоди для регулятора:

Монополія на встановлення тарифів: Національна комісія має широкі повноваження щодо затвердження тарифів, що створює ризик корупційних домовленостей з підприємствами.

Цитата: "Встановлення тарифів на комунальні послуги суб'єктам природних монополій..." (ст. 6, п. 2).

Лобістські формулювання:

Привілеї для підприємств з нетрадиційної енергетики: Виключення з регулювання виробників енергії на основі поновлюваних джерел може бути лобістським тиском.

Цитата: "Тарифи на виробництво теплової енергії на установках з використанням нетрадиційних або поновлюваних джерел енергії не регулюються..." (ст. 1, п. 3).

Слабкий механізм відшкодування шкоди:

Невизначеність у порядку відшкодування шкоди через неправомірні рішення Комісії.

Цитата: "Збитки... підлягають відшкодуванню у порядку, передбаченому законодавством" (ст. 18, ч. 2), але конкретний механізм відсутній.

Примітка: Аналіз засновано на тексті закону з урахуванням змін до 2022 року.

Приклад 2. Генерація коротких резюме законопроектів для депутатів

Для реалізації цього завдання створюється первинний промпт, який після цього треба формалізувати:

Вам потрібно підготувати стисле, чітке та зрозуміле резюме для депутатів про законопроект. Законопроект стосується [вкажіть тему, наприклад, 'реформи системи охорони здоров'я або 'зміни у податковому законодавстві'].

Включіть наступну інформацію:

- Назва законопроекту.
- Мета законопроекту (1-2 речення).
- Основні положення (3-5 пунктів).
- Очікувані переваги чи результати (1-2 речення).
- Можливі виклики чи ризики (за потреби).

Використовуйте максимально доступну мову, уникайте складної

юридичної термінології. Резюме має бути не більше 200 слів.

Ось текст законопроекту:

На запит створення структурованого промпту для вирішення цього завдання BMM Qwen, зокрема, видає рішення, що у відповідності з фреймворком безкодового промпт-інжинірингу, пропонується наступна структура для автоматичного створення резюме законопроекту для депутатів:

Ти експерт із безкодового аналізу текстів. Виконай завдання через п’ять кроків:

Крок 1. Витягни назву

Застосуй F_title до тексту [X].

Крок 2. Визнач мету

Використовуй F_purpose для формулювання мети.

Крок 3. Виділи основні положення

Розбий текст на розділи та опиши кожен у форматі "пункт: сутність" (3-5 пунктів).

Крок 4. Оціни переваги

Застосуй F_benefits для опису очікуваних результатів.

Крок 5. Перевір ризики

Якщо законопроект містить суперечливі або нереалістичні положення, опиши їх. Інакше – пропустіть.

Результат подай у форматі:

1. Назва: [рядок]

2. Мета: [1-2 речення]

3. Основні положення: [список пунктів]

4. Переваги: [1-2 речення]

5. Виклики/ризики: [список або "відсутні"]

Обмеження: не більше 500 слів, доступна мова, мінімум термінології.

1. Примітив "Функція" для витягування назви законопроекту

F_title: X → Y – функція для витягування назви.

X – текст законопроекту.

Y – рядок із назвою.

Промпт-функція:

"Виділи назву законопроекту з тексту [X] і поверни її як один рядок."

2. Примітив "Функція" для визначення мети

F_purpose: X → Z – функція для формулювання мети.

параметр = {"довжина": "1-2 речення"}.

Промпт-функція:

"Сформулюй мету законопроекту [X] у 1-2 реченнях, використовуючи просту мову."

3. Примітив "Цикл" для аналізу основних положень

S = {розділ1, розділ2, ...} – множина розділів законопроекту.

F_provision: s_i → W – функція для стислого опису кожного розділу.

Результат: об'єднання описів (W_1 ⊕ W_2 ⊕ ...).

Промпт-цикл:

"Для кожного розділу законопроекту [X]:

1. Виділи головну ідею.

2. Сформулюй її як короткий пункт (3-5 пунктів загалом)."

4. Примітив "Функція" для опису переваг

F_benefits: X → V – функція для оцінки переваг.

параметр = {"довжина": "1-2 речення"}.

Промпт-функція:

"Опиши очікувані переваги законопроекту [X] у 1-2 реченнях, уникай термінології."

5. Примітив "Умова" для перевірки ризиків

C_risk: X → {True, False} – умова наявності ризиків.

A1: Якщо True, застосуй F_risks.

A2: Якщо False, поверни порожній список.

Промпт-умова:

"Перевір, чи законопроект [X] містить елементи, що можуть викликати виклики або ризики.

Якщо так:

– Опиши їх коротко (за потреби).

Якщо ні:

– Пропустіть цей розділ."

Після опрацювання формалізованого промпту на прикладі Закону України «Про хміль та хмелепродукти» від 27 березня 2025 року № 4341-IX отримуємо результат:

Назва: Закон України «Про хміль та хмелепродукти»

Мета: Регулювання вирощування, сертифікації, переробки та торгівлі хмелем і хмелепродуктами для забезпечення якості, підтримки українських виробників, відповідності міжнародним стандартам та стимулювання експорту.

Основні положення:

Сертифікація: Встановлення правил видачі сертифікатів на хміль і хмелепродукти, включаючи вимоги до документації (номер сертифіката, опис продукту, країна походження).

Контроль за переробкою: Обов'язок ведення обліку перероблених матеріалів, маркування упаковки під наглядом сертифікаційних центрів.

Імпорт: Вимоги до сертифікації іноземних партій хмелю, включаючи атестат еквівалентності від закордонних органів.

Відповідальність: Штрафи за порушення вимог щодо підготовки, маркування або сертифікації продукції (від 5 до 10 мінімальних зарплат).

Технічні стандарти: Визначення термінів (наприклад, «закритий операційний цикл») та вимог до якості хмелю (вміст альфа-кислот).

Переваги: Покращення якості продукції, збільшення конкурентоспроможності українського хмелю на міжнародному ринку, уніфікація процедур сертифікації, підтримка експортерів через відповідність стандартам ЄС і США.

Виклики/ризики: Відсутні. Документ передбачає чіткі механізми реалізації, включаючи контроль, сертифікацію та штрафні санкції, без суперечливих положень.

Приклад 3. Аналіз звітів виконавчої влади на відповідність законодавству

Для аналізу звітів виконавчої влади на відповідність законодавству створюється первинний промпт:

Проаналізуй звіти або їх окремі розділи, підготовлені виконавчою владою, на відповідність прийнятим законам. Визнач, чи дотримані вимоги законодавства, чи виконані положення відповідних нормативно-правових актів, та чи є відхилення або невідповідності. Зроби акцент на виявленні:

- Фактів неповного виконання законів.
- Ризиків, що виникають через відхилення від норм.
- Рекомендацій для забезпечення повної відповідності звітів

законодавчим нормам.

Надай чіткий висновок щодо відповідності кожного розділу звіту конкретним законодавчим актам та пропозиції щодо покращення.

Ось текст звіту: [Текст]

Струтурований промпт, що створюється ВММ у відповідності із завантаженим фреймворком має вигляд:

Проаналізуйте звіт або окремі його розділи, підготовлені виконавчою владою.

Виявіть:

Факти неповного виконання законів .

Ризики, що виникають через порушення норм (економічні, соціальні, правові).

Рекомендації для забезпечення повної відповідності звітів законодавству.

Формат відповіді

1. Відповідність законодавству

Примітив: Цикл (For-Loop)

Для кожного розділу звіту:

Нормативно-правовий акт: Вкажіть конкретний закон або статтю (наприклад, Закон України "Про державне регулювання у сфері комунальних послуг", ст. 10).

Вимога закону: Наведіть текст норми, яка регулює цю сферу.

Оцінка виконання (Примітив: Умова):

- Відповідність: [Короткий опис виконання].
- Порушення: [Опис невідповідності], [Цитата зі звіту], [Посилання на порушений закон].

2. Суперечності з законодавством

Примітив: Функція (Abstraction)

Для кожного виявленого порушення:

Назва закону: Вкажіть закон, який порушено.

Конфліктна норма у звіті: Опишіть, як формулювання звіту суперечить закону.

Цитата зі звіту: Наведіть текст звіту, що підтверджує суперечність.

3. Корупційні загрози

Примітив: Функція (Abstraction)

Для кожного розділу звіту:

Ознака корупції: Вкажіть тип (надмірні вигоди, лобістські формулювання, відсутність контролю).

Опис прояву: Як ця ознака реалізується у тексті?

Цитата зі звіту: Наведіть фрагмент, що підтверджує загрозу.

Хто може скористатися: Вкажіть групу або особу, які отримують переваги.

4. Рекомендації

Примітив: Функція (Abstraction)

Для кожного виявленого порушення або ризику:

Пропозиція: Як усунути невідповідність?

Обґрунтування: На якому законі або нормі базується?

5. Загальний висновок

Примітив: Цикл + Умова

Підсумуйте:

Чи відповідає звіт законодавству?

Які розділи мають критичні невідповідності?

Які ризики найбільш суттєві?

Які кроки пріоритетні для усунення порушень?

Формалізація примітивів у промпті

Цикл (For-Loop):

S = {розділи звіту} – множина елементів для аналізу.

F = "перевірка відповідності законодавству" – операція для кожного елемента.

Умова (If-Else):

C = "чи вимога закону виконана?" → True/False.

A_1 = "опис відповідності", A_2 = "опис порушення".

Функція (Abstraction):

F_risk(X, параметр = {"тип_ризиків": ["економічні", "соціальні", "правові"]}) → аналіз ризиків.

F_corruption(X, параметр = {"ознаки_корупції": ["надмірні_вигоди", "лобістські_формулювання", "відсутність_контролю"]}) → виявлення

загроз.

Вхідні дані для моделі

Текст звіту: [Вставте текст звіту].

Перелік законів: [Закон України "Про державне регулювання у сфері комунальних послуг", Закон України "Про природні монополії", Кодекс України про адміністративні правопорушення].

Вихідний формат

1. Відповідність законодавству:
   – Розділ [X]:
     – Нормативно-правовий акт: [Назва закону, стаття].
     – Вимога: [Цитата з закону].
     – Оцінка: [Опис, цитата зі звіту].
2. Суперечності з законодавством:
   – Закон: [Назва закону].
   – Конфліктна норма: [Опис невідповідності].
   – Цитата: [Фрагмент звіту].
3. Корупційні загрози:
   – Ознака: [Тип загрози].
   – Опис: [Як реалізується].
   – Цитата: [Фрагмент звіту].
   – Хто виграє: [Група/особа].
4. Рекомендації:
   – Пропозиція: [Дія].
   – Обґрунтування: [Посилання на закон].
5. Загальний висновок:
   – Відповідність: Так/Ні.
   – Критичні розділи: [Перелік].
   – Пріоритетні дії: [Перелік рекомендацій].

Результат виконання цього промпту у середовищі ВММ Qwen для звіту Державної аудиторської служби України про здійснення регуляторної діяльності у 2021 році має вигляд:

1. Відповідність законодавству

Розділ 1: Вступна частина

Нормативно-правовий акт: Закон України «Про засади державної регуляторної політики у сфері господарської діяльності», ст. 8.

Вимога закону: «До ухвалення регуляторного акта обов'язково проводиться аналіз його регуляторного впливу».

Оцінка:

Порушення: Проєкти двох регуляторних актів (пункти 21 та 22 Плану розробки актів) розроблені без проведення аналізу регуляторного впливу, оскільки перебували на стадії узгодження з Мінфіном. Цитата зі звіту: «відповідні заходи з регуляторної політики, визначені Законом, щодо них не проводились».

Посилання на порушений закон: Ст. 8 Закону України «Про засади державної регуляторної політики».

Розділ 2: Загальна частина

Нормативно-правовий акт: Закон України «Про засади державної регуляторної політики», ст. 4, ст. 9.

Вимога закону: «Регуляторні акти мають відповідати міжнародним зобов'язанням України» (ст. 4); «публічне обговорення проєктів регуляторних актів» (ст. 9).

Оцінка:

Порушення: Проєкт закону про віддалений контроль був узгоджений з Держрегуляторною службою, але після зауважень Мін'юсту щодо невідповідності Конвенції про права людини, Мінфін повернув документ для доопрацювання. Цитата зі звіту: «через таку позицію Мінфіну… подальшу роботу над законопроєктом припинено».

Посилання на порушений закон: Ст. 4, ст. 9 Закону України «Про засади державної регуляторної політики».

2. Суперечності з законодавством

Закон: Закон України «Про засади державної регуляторної політики», ст. 8.

Конфліктна норма у звіті: Відсутність аналізу регуляторного впливу для проєктів актів, що порушує вимогу ст. 8.

Цитата зі звіту: «відповідні заходи з регуляторної політики, визначені Законом, щодо них не проводились».

Закон: Закон України «Про засади державної регуляторної політики», ст. 4.

Конфліктна норма у звіті: Неврахування положень Конвенції про права людини у проєкті закону.

Цитата зі звіту: «Мінфін листом від 18.11.2021 № 33010-03-5/35457 повернув законопроєкт для доопрацювання з метою забезпечення врахування положень зазначеної Конвенції».

3. Корупційні загрози

Ознака: Відсутність контролю.

Опис прояву: Відсутність аналізу регуляторного впливу для проєктів актів створює ризик ухвалення норм, що не відповідають реальним потребам або міжнародним стандартам.

Цитата зі звіту: «відповідні заходи з регуляторної політики, визначені Законом, щодо них не проводились».

Хто може скористатися: Лобістські групи, які могли б вплинути на формулювання актів без публічного обговорення.

4. Рекомендації

Пропозиція: Обов'язкове проведення аналізу регуляторного впливу для всіх проєктів актів перед їх узгодженням.

Обґрунтування: Ст. 8 Закону України «Про засади державної регуляторної політики».

Пропозиція: Перед поданням проєктів законів на розгляд Уряду – перевірка відповідності міжнародним зобов'язанням України.

Обґрунтування: Ст. 4 Закону України «Про засади державної регуляторної політики».

5. Загальний висновок

Відповідність: Ні.

Критичні розділи: Розділ 1 (непроведення аналізу регуляторного впливу), Розділ 2 (порушення вимог щодо міжнародних зобов'язань).

Пріоритетні дії:

Забезпечити обов'язковий аналіз регуляторного впливу для всіх проєктів актів.

Впровадити механізми перевірки відповідності проєктів законів міжнародним нормам перед їх поданням на узгодження.

Як можна бачити, запропонованого підходу для вирішення задач парламентського контролю має такі переваги, як чіткість, гнучкість, композиція різних висновків у єдиний звіт. При цьому кожний примітив виконує окрему підзадачу, що відповідає принципам фреймворку; параметри (тип_ризиків, джерела_законодавства тощо) можна адаптувати під конкретний законопроект, результати декількох функцій об'єднуються в єдиний вивід, що моделює складну систему аналізу. Сформовані структури дозволяють автоматизувати аналіз тексту документів через формальні примітиви, уникнувши неоднозначностей традиційних промптів до ВММ.

Застосування фреймворку безкодового програмування до задач парламентського контролю показало:

– ефективність, мінімізацію часу аналізу;

– стандартизацію подання результатів, що полегшує їхню подальшу обробку;

– точність, а саме, використання формальних примітивів забезпечує чіткість виводу та мінімізує суб'єктивність;

– масштабованість при якій один і той самий шаблон може використовуватися для аналізу різноманітних документів.

Результати тестування на конкретних прикладах демонструють високу практичну цінність такого підходу. На основі проведеного аналізу пропонується:

– створити бібліотеку типових промптів для парламентського контролю;

– розробити графічний інтерфейс для вбудовування фреймворку в існуючі парламентські системи;

– інтегрувати внутрішні правові бази даних для підвищення точності аналізу;

– забезпечити тренінги для парламентаріїв щодо використання інструментів ШІ;

– встановити етичні та правові межі використання ШІ в парламентському контролі.

Застосування фреймворку безкодового програмування в парламентському контролі надає можливість ефективно автоматизувати аналіз правових документів, виявляти потенційні ризики, юридичні суперечності та корупційні загрози. Використання формальних примітивів (умова, цикл, функція, мітка, перехід) забезпечує структурований підхід до обробки тексту, що робить процес повторним, стандартизованим і масштабованим. Це особливо важливо в сучасних умовах, коли необхідно оперативно реагувати на великі обсяги нормативно-правових актів.

Дослідження показало, що впровадження такого фреймворку може значно скоротити час на аналіз документів, підвищити точність виявлення порушень та забезпечити об'єктивність висновків. Крім того, формалізація логіки аналізу дозволяє легко адаптувати систему під різні категорії завдань – від перевірки відповідності законопроектів чинним нормам до оцінки звітів виконавчої влади.

Подальші напрямки досліджень можуть бути спрямовані на:

Створення бібліотеки типових промптів для різних сценаріїв парламентського контролю.

Інтеграцію фреймворку з внутрішніми правовими базами даних, що забезпечить більш глибоку верифікацію відповідності законодавству.

Розробку графічного інтерфейсу для простоти використання інструменту непрограмістами.

Тренінгову підготовку парламентаріїв для ефективного використання інструментів ШІ у повсякденній роботі.

Встановлення етичних і правових меж використання штучного інтелекту в парламентському контролі, щоб забезпечити прозорість, справедливість та захист даних.

Отже, запропонований фреймворк має потенціал суттєво підвищити ефективність парламентської діяльності, зробити її більш аналітичною, стратегічною та відкритою для суспільства.

Розділ 7. Конфліктологія штучного інтелекту

У сучасну епоху цифрової трансформації парламентський контроль зіштовхується з принципово новим викликом – інтеграцією штучного інтелекту у всі сфери державного управління: від аналізу законопроєктів до моніторингу урядової діяльності. Застосування генеративних мовних моделей і інших систем ШІ відкриває можливості для підвищення ефективності, прозорості та об'єктивності контролю, але водночас породжує специфічні конфлікти, які не піддаються традиційним методам аналізу. Ці суперечності виникають не лише між людьми чи інституціями, а між людиною та машиною, між особистою відповідальністю та автономними рішеннями, між правовою визначеністю та генерацією невизначеного змісту.

Класична конфліктологія, орієнтована на міжособистісні, інституційні чи міжнародні протиріччя, не має достатніх інструментів для розуміння таких явищ. Це створює потребу в новій науковій парадигмі – конфліктології штучного інтелекту, яка систематизує, класифікує та пропонує механізми вирішення конфліктів, спричинених унікальними властивостями ШІ: автономністю, непрозорістю, адаптивністю, здатністю до самонавчання та генерації нового змісту.

Ця дисципліна виходить за межі технічного аналізу, охоплюючи правові, етичні, політичні та соціальні виміри. Вона дає змогу парламентському контролю не лише реагувати на наслідки впровадження ШІ, а й прогнозувати, запобігати та регулювати потенційні конфлікти на рівні законодавства, адміністрування та громадського дискурсу.

Поширення ШІ у законотворчості та державному управлінні – від автоматизації рутинних процедур до використання генеративних моделей для розробки законопроєктів – супроводжується серйозними викликами. Вони загрожують основам демократичного правління, принципам справедливості та захисту прав людини. Хоча деякі проблеми, пов'язані з автоматизацією, відомі давно, великі мовні моделі вносять нові ризики: їхні рішення часто непрозорі, вони навчаються на історичних даних, що може відтворювати упередження, і генерують зміст, який не завжди відповідає фактам.

У цьому контексті особливо важливим стає системний підхід до виявлення конфліктів, спровокованих впровадженням ШІ у законодавчу систему. Такі конфлікти – це не просто технічні збої, а наслідок зміни балансу між правами, відповідальністю та легітимністю внаслідок взаємодії різних суб'єктів: громадян, держави, влад, інституцій, поколінь. Багато з них мають прихований характер і можуть залишатися непоміченими до моменту, коли їхні наслідки стають незворотними.

Саме тому в останні роки формується нова міждисциплінарна галузь, що поєднує інсайти з правознавства, філософії технологій, політології та комп'ютерних наук. Вона фокусується на вивченні саме тих конфліктів, які

неможливі в системах, що ґрунтуються на простій автоматизації чи жорсткому програмуванні. До них належать: відтворення історичної дискримінації через навчання на упереджених даних, генерація «фейкових» правових посилань (галюцинації), автоматичне формування законодавчих пріоритетів без політичного контролю, а також закріплення тимчасових соціальних патернів як постійних правових норм.

У цьому розділі розглядаються джерела таких суперечностей, зокрема – відсутність чітких правових визначень, яка, за результатами аналізу, є одним із найпоширеніших конфліктогенів у контексті ШІ. Особливу увагу приділено властивостям ШІ – «чорному ящику», автономності, генерації змісту, адаптивності та скалярності, – які вже сьогодні проявляються в законопроєктах, адміністративних рішеннях та суспільній практиці.

Основу аналізу становлять результати дослідження, представленого в статті «Виявлення та аналіз конфліктів у законодавстві, пов'язаних із застосуванням штучного інтелекту». У ній запропоновано сім структурованих промптів, призначених для систематичного виявлення, формалізації та інтеграції конфліктогенних норм у інтелектуальну систему парламентського контролю.

7.1. Конфліктологія ШІ як наукова дисципліна

У зв'язку зі стрімким поширенням технологій штучного інтелекту у сферах державного управління, юриспруденції, кібербезпеки та суспільного життя виникає необхідність у систематизації та науковому осмисленні нових форм соціально-правових конфліктів, які не піддаються традиційним методам аналізу. Цей виклик стимулює формування нової наукової дисципліни – конфліктології штучного інтелекту (конфліктології ШІ) – галузі, яка системно вивчає специфічні конфлікти, породжені унікальними властивостями ШІ, такими як автономність, «чорний ящик», генерація змісту та адаптивність.

Визначення конфліктології ШІ

Конфліктологія ШІ – це наукова дисципліна, яка вивчає природу, механізми виникнення, класифікацію, динаміку та регулювання конфліктних ситуацій, пов'язаних із розробкою, впровадженням та використанням систем штучного інтелекту. Вона виходить за межі технічного аналізу і охоплює правові, етичні, інституційні та міжнародні аспекти взаємодії людини, суспільства та машин. Конфліктологія ШІ не лише фіксує наслідки використання технологій, але й створює інструментарій для проактивного прогнозування, виявлення та запобігання потенційних конфліктів на етапі законотворчості, адміністрування та громадського дискурсу.

Об'єктом конфліктології ШІ є конфліктні ситуації, які виникають у різних сферах через застосування технологій ШІ. До них належать:

– правові конфлікти: суперечності між нормами, викликані неоднозначністю термінів, технічними помилками, лакунами в регулюванні;

– етичні конфлікти: питання відповідальності, прозорості, упередження, порушення прав людини;

– інституційні конфлікти: розбіжності між державними органами щодо регулювання ШІ, конфлікти інтересів, відсутність координації;

– міжнародні конфлікти: розбіжності в правових підходах до регулювання ШІ між країнами, атрибуція кібератак, використання ШІ в гібридних конфліктах.

Предмет дослідження

Предметом конфліктології ШІ є механізми виникнення, класифікація, прогнозування та запобігання конфліктам, пов'язаним із ШІ. Дисципліна досліджує:

– як унікальні властивості ШІ («чорний ящик», автономність, генерація змісту) породжують нові типи суперечностей;

– які структури норм є конфліктогенними (наприклад, відсутність визначень, дублювання норм, технічні помилки);

– як ієрархічно розгортаються причини конфліктів від первинних (наприклад, «відсутність стандартів») до вторинних («полісемія понять», «лакуни у тексті»);

– як можна прогнозувати наслідки впровадження ШІ за допомогою сценарного аналізу та причинно-наслідкових моделей.

Методологія

Методологічний апарат конфліктології ШІ базується на інтеграції сучасних інформаційних технологій та методів наукового аналізу:

– Семантичний інжиніринг – процес перетворення природної мови правових текстів на структуровані моделі знань.

– Промпт-кероване зондування – стратегія використання великих мовних моделей для систематичного виявлення причин суперечностей за допомогою структурованих запитів (наприклад, «Перелічіть 10 причин відсутності визначень...»).

– Побудова мереж знань – формування семантичних та причинно-наслідкових мереж на основі виявлених сутностей і зв'язків.

– Візуалізація – використання інструментів, таких як Gephi та GraphViz, для наочної інтерпретації структури конфліктів.

– Формалізація – перехід від текстового аналізу до виконуваних модулів, що забезпечує відтворюваність, верифікацію та підзвітність.

Ця методологія дозволяє не лише аналізувати, а й створювати системи контролю, здатні виявляти конфлікти ще до їх реалізації.

У сучасному цифровому світі, де інформація є важливим ресурсом, загрози дезінформації, кібератак і маніпуляцій стають дедалі складнішими й витонченішими. Штучний інтелект відіграє провідну роль у створенні, аналізі та виявленні контенту. У цьому контексті змагальний ШІ (Adversarial AI, ЗШІ) виступає як інструмент, що використовується як для атак, так і для захисту в цифровому просторі, демонструючи протиріччя та конфлікти між різними учасниками, конфігураціями й рівнями ШІ. Це визначає важливість дослідження ЗШІ для підвищення інформаційної безпеки та боротьби з дезінформацією. Розглядаються конфлікти ШІ у межах запропонованої у цій роботі парадигми через призму протиріч, зіткнення інтересів і взаємодії в системах, що охоплюють різні рівні. Також розглядаються системи, що включають взаємодію декількох ШІ, кіберфізичних систем або їхніх комбінацій, які можуть бути організовані різними акторами чи виникати внаслідок самоорганізації.

У сучасному цифровому просторі змагальний штучний інтелект стає не лише технологічним, а й правовим викликом. Його застосування у кібербезпеці та інформаційних війнах створює ризики для персональних даних, свободи слова та інформаційної безпеки держав. Законодавство різних країн та міжнародні правові ініціативи намагаються виробити баланс між розвитком ШІ-технологій і запобіганням їхньому зловживанню. Зокрема, Акт про штучний інтелект ЄС (The EU Artificial Intelligence Act)[120] [1] містить положення щодо високоризикових ШІ-систем, які можуть впливати на основоположні права та громадську безпеку.

Змагальний ШІ вже широко застосовується у кількох ключових напрямках:

– Генерація та детекція фейків, а саме, системи ШІ використовуються для створення реалістичних, але неправдивих текстів, зображень чи відео. Паралельно розробляються інструменти для виявлення таких фейків, що веде до постійної гонки озброєнь між генеративними та детективними моделями.

– Моделювання суперництва у кібервійнах, зокрема, ШІ моделює як атаки, так і захисні стратегії, спрямовані на виявлення та нейтралізацію загроз у кіберпросторі.

– Моделі ШІ використовуються в інформаційних війнах для аналізу реакцій аудиторії на кампанії, створення маніпулятивного контенту, синтезу, генерування і здійснення інформаційних атак та протидії ним.

– Конфлікти в системах «людина-суспільство-держава-природа-техносфера-ШІ».

[120] The EU Artificial Intelligence Act, 2024. Режим доступу: https://artificialintelligenceact.eu/

Попри значний прогрес, дослідження змагального ШІ все ще стикаються з викликами, такими як недостатня прозорість моделей, ризики отруєння даних (Data Poisoning) та складнощі у виявленні незвичайної поведінки систем.

Метою цієї статті є дослідження механізмів ЗШІ у трьох основних напрямках: генерація та детекція фейків, штучне суперництво в системах «людина-суспільство-держава-природа-техносфера-ШІ» у кібервійнах та інформаційних війнах. При цьому розглядаються математичні основи, концепції, стратегії застосування та перспективи розвитку цих технологій. Для цього авторами поставлені такі задачі:

1. Проаналізувати існуючі підходи до генерації та детекції фейків, зокрема механізми змагального машинного навчання (adversarial training)[121].

2. Дослідити роль ЗШІ у кібербезпеці, включаючи моделювання атак та захисту.

3. Розкрити аспекти використання ШІ в інформаційних війнах, зокрема прогнозування поведінки аудиторії та особливості створення маніпулятивного контенту.

4. Представити математичні моделі та формалізм для опису динаміки змагальних взаємодій між системами ШІ.

5. Запропонувати рекомендації щодо впровадження та розвитку ЗШІ у сфері інформаційної безпеки.

Дослідження в галузі змагального штучного інтелекту активно розвиваються останніми роками, зосереджуючи увагу на багатьох аспектах його застосування в кібербезпеці, інформаційних війнах і генерації фейків.

Теоретична база питання, що розглядається, спирається на міжнародні дослідження з регулювання ШІ, зокрема на праці[122], в якій аналізуються етичні ризики алгоритмічного управління, та[123], в якій досліджується проблема «чорного ящика» в державному секторі. Важливим джерелом є також Зелена книга Європейської комісії[124], яка вперше системно визначила

[121] Clarence Chio, David Freeman. Machine Learning and Security: Protecting Systems with Data and Algorithms. O'Reilly Media; 1st edition (February 20, 2018). 386 pp. ISBN 978-1491979907

[122] Ebert, I., 2024. Responding to unusual government request for user data: How tech companies make sense of human rights. Big Data & Society, 11(1), p.20539517241232638. DOI: 10.1177/20539517241232638

[123] Shackelford, S.J., Asare, I.N., Dockery, R., Raymond, A.H. and Sergueeva, A., 2021. Should We Trust a Black Box to Safeguard Human Rights?. UCLA Journal of International Law and Foreign Affairs, 26(1), pp.35-88. DOI: https://www.jstor.org/stable/48774493

[124] European Commission. (2020). White Paper on Artificial Intelligence: A European approach to excellence and trust. Brussels: European Commission. URL: https://commission.europa.eu/publications/white-paper-artificial-intelligence-european-approach-excellence-and-trust_en

рівні ризику від застосування ШІ, та Рамкова конвенція Ради Європи з ШІ[125], що закладає основи міжнародного правового контролю. У вітчизняному контексті враховуються аналітичні матеріали Міністерства цифрової трансформації України, зокрема «Біла книга з регулювання ШІ в Україні»[126], а також ініціативи громадських організацій, зокрема Української студентської ліги. Ці документи демонструють як потенціал, так і ризики впровадження ШІ в правову сферу, але часто не містять глибокого аналізу конфліктогенних механізмів. Додатково враховуються дослідження з гуманітарних наук, зокрема робота[127] щодо «суспільного капіталізму», яка розкриває механізми експлуатації даних, та дослідження[128] про «алгоритмічну кривду», що підтверджують важливість правового контролю над автоматизованими системами. Також варто зазначити роботи щодо прозорості алгоритмів[129] та про ризики генеративного ШІ у правових системах, які підсилюють аргументацію щодо необхідності проактивного парламентського контролю.

Генеративні змагальні мережі (Generative adversarial networks, GANs) були запропоновані в 2014 році Яном Ґудфелоу[130], як клас алгоритмів штучного інтелекту, що використовуються в некерованому навчанні. На той час цей конфлікт ШІ розглядався, як змагання двох штучних нейронних мереж, одна з одною в рамках гри з нульовою сумою.

Далі, генеративно-змагальні мережі стали ключовою технологією в контексті створення фейкового контенту. Роботи[131] демонструють можливості GAN у створенні правдоподібних зображень і текстів, що згодом викликало розвиток детекційних методів. Публікації, такі як[132], пропонують сучасні

[125] Council of Europe. (2024). Framework Convention on Artificial Intelligence and Human Rights, Democracy and the Rule of Law. ETS No. 225. URL: https://www.coe.int/en/web/artificial-intelligence/the-framework-convention-on-artificial-intelligence

[126] Біла книга з регулювання ШІ в Україні: бачення Мінцифри». (2024). URL: https://thedigital.gov.ua/storage/uploads/files/page/community/docs/Регулювання ШІ.pdf

[127] Zuboff, S., 2023. The age of surveillance capitalism. In Social theory re-wired (pp. 203-213). Routledge. ISBN 1003320609

[128] Citron, D.K., 2007. Technological due process. Wash. UL Rev., 85, p.1249. https://heinonline.org/HOL/LandingPage?handle=hein.journals/walq85&div=38&id=

[129] Villasenor, J., 2023. Generative Artificial Intelligence and the practice of Law: Impact, opportunities, and risks. Minn. JL Sci. & Tech., 25, p.25. URL: https://heinonline.org/HOL/LandingPage?handle=hein.journals/mipr25&div=22&id=

[130] Goodfellow, Ian; Pouget-Abadie, Jean; Mirza, Mehdi; Xu, Bing; Warde-Farley, David; Ozair, Sherjil; Courville, Aaron; Bengio, Joshua (2014). Generative Adversarial Networks. Preprint arXiv. arXiv:1406.2661. DOI: 10.48550/arXiv.1406.2661

[131] Arora, T. and Soni, R., 2021. A review of techniques to detect the GAN-generated fake images. Generative Adversarial Networks for Image-to-Image Translation, pp.125-159.DOI: 10.1016/B978-0-12-823519-5.00004-X

[132] Marra, Francesco, Diego Gragnaniello, Davide Cozzolino, and Luisa Verdoliva. "Detection of gan-generated fake images over social networks." In 2018 IEEE conference on multimedia information processing and retrieval (MIPR), pp. 384-389. IEEE, 2018.
DOI: 10.1109/MIPR.2018.00084

алгоритми для розпізнавання дезінформації з використанням машинного навчання.

Кібербезпека є ще одним важливим напрямом, де ЗШІ демонструє свою ефективність. У роботі[133] досліджено застосування змагальних атак, наприклад, шляхом введення «адверсаріальних прикладів», які вводять у оману захисні системи. Наприклад, публікація[134], демонструє вплив цих методів на системи класифікації та виявлення аномалій. З іншого боку, в сучасних умовах, коли змагальний штучний інтелект стає все більш поширеним інструментом у кібервійнах, важливим аспектом є захист від атак ЗШІ, які використовують методи машинного навчання (ML) та глибокого навчання (DL). Дослідження[135] пропонує систематичний огляд методів захисту від адверсаріальних атак, що дозволяє краще зрозуміти, як ЗШІ може бути використаний для підриву кібербезпеки та які стратегії захисту можуть бути ефективними.

У роботі[136] розглядаються jailbreak-атаки (від слов «втеча з в'язниці») на великі мовні моделі, які можуть бути використані в інформаційних та кібернетичних конфліктах. Ці атаки обходять вбудовані механізми безпеки великих мовних моделей, щоб викликати шкідливі відповіді. Вони можуть бути використані для поширення дезінформації, маніпуляцій або навіть кібератак через штучний інтелект. Моделювання інформаційних кампаній та впливу маніпулятивного контенту на аудиторію розглядається у дослідженнях[137], які акцентують увагу на розробці концепцій і стратегій дезінформації та оцінці їх впливу на соціальні мережі. Моделі пропаганди й маніпуляції, такі як[138], включають аналіз методів, що дозволяють адаптувати

[133] Sarker, I.H., 2023. Multi-aspects AI-based modeling and adversarial learning for cybersecurity intelligence and robustness: A comprehensive overview. Security and Privacy, 6(5), p. 295. DOI: 10.1002/spy2.295

[134] Bouaziz, A., Nguyen, M. D., Valdés, V., Cavalli, A. R., & Mallouli, W. (2023, July). Study on Adversarial Attacks Techniques, Learning Methods and Countermeasures: Application to Anomaly Detection. In ICSOFT (pp. 510-517). DOI: 10.5220/0012125100003538

[135] Khaleel, Yahya Layth, Mustafa Abdulfattah Habeeb, A. S. Albahri, Tahsien Al-Quraishi, O. S. Albahri, and A. H. Alamoodi. "Network and cybersecurity applications of defense in adversarial attacks: A state-of-the-art using machine learning and deep learning methods." Journal of Intelligent Systems 33, no. 1 (2024): 20240153. DOI: 10.1515/jisys-2024-0153

[136] Miao Yu, Junfeng Fang, Yingjie Zhou, Xing Fan, Kun Wang, Shirui Pan, Qingsong Wen. LLM-Virus: Evolutionary Jailbreak Attack on Large Language Models. Preprint arXiv, arXiv: 2501.00055. DOI: 10.48550/arXiv.2501.00055

[137] Campbell, Colin, Kirk Plangger, Sean Sands, and Jan Kietzmann. "Preparing for an era of deepfakes and AI-generated ads: A framework for understanding responses to manipulated advertising." Journal of Advertising 51, no. 1 (2022): 22-38. DOI: 10.1080/00913367.2021.1909515

[138] Altinay, E. A., & Utku, Kose. (2021). Manipulation of Artificial Intelligence in Image Based Data: Adversarial Examples Techniques. Journal of Multidisciplinary Developments, 6(1), 8-17. URL: http://www.jomude.com/index.php/jomude/article/view/88

контент під конкретні цільові аудиторії для забезпечення максимального впливу на них.

Незважаючи на суперечливу репутацію, напрямку нейролінгвістичного програмування (НЛП), яка склалась на цей час, його застосування в сфері ШІ виявилось достатньо продуктивним. Так, воно знайшло застосування в адаптації моделей ШІ для генерації спеціалізованих запитів, здатних викликати небажану поведінку LLM У роботі[139] описано методики створення змагальних текстових прикладів, що впливають на класифікаційні моделі. Останніми роками стало очевидно, що глибокі нейронні мережі вразливі до змагальних атак, які викликаються навмисними змінами вхідних даних. У відповідь на це було запропоновано різні захисні механізми для задач обробки природної мови, які не лише протистоять атакам, але й допомагають уникнути перенавчання моделей.

Інші дослідження, наприклад[140], аналізують, як атаки на мовні моделі можуть призводити до некоректних результатів або навіть до витоку даних. У дослідженні показано, що аналіз різниць між відбитками мовних моделей до та після оновлення може розкрити детальну інформацію про зміни у навчальних даних, що має важливі наслідки для конфіденційності.

Бекдори в системах LLM є ще однією актуальною темою конфліктології ШІ. В роботі[141] досліджують механізми впровадження бекдорів у мовні моделі шляхом модифікації даних навчання. У дослідженні[142] запропоновано алгоритми виявлення таких загроз, що базуються на аналізі моделей активації нейронних мереж.

Важливим внеском стали роботи, які аналізують вплив бекдорів на системи критичної інфраструктури[143].

---

[139] Goyal, Shreya, Sumanth Doddapaneni, Mitesh M. Khapra, and Balaraman Ravindran. "A survey of adversarial defenses and robustness in nlp." ACM Computing Surveys 55, no. 14s (2023): 1-39. DOI: 10.1145/3593042

[140] Zanella-Béguelin, Santiago, Lukas Wutschitz, Shruti Tople, Victor Rühle, Andrew Paverd, Olga Ohrimenko, Boris Köpf, and Marc Brockschmidt. "Analyzing information leakage of updates to natural language models." In Proceedings of the 2020 ACM SIGSAC conference on computer and communications security, pp. 363-375. 2020. DOI: 10.1145/3372297.3417880

[141] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, Vitaly Shmatikov. How To Backdoor Federated Learning. Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics, PMLR 108:2938-2948, 2020.

[142] Shen, Guangyu, Siyuan Cheng, Zhuo Zhang, Guanhong Tao, Kaiyuan Zhang, Hanxi Guo, Lu Yan et al. "BAIT: Large Language Model Backdoor Scanning by Inverting Attack Target." In 2025 IEEE Symposium on Security and Privacy (SP), pp. 103-103. IEEE Computer Society, 2024. DOI: 10.1109/SP61157.2025.00103

[143] Usman, Y., Gyawali, P. K., Gyawali, S., & Chataut, R. (2024, October). The Dark Side of AI: Large Language Models as Tools for Cyber Attacks on Vehicle Systems. In 2024 IEEE 15th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON) (pp. 169-175). IEEE Computer Society, 2024. DOI: 10.1109/ UEMCON62879.2024.10754676

В останні роки виконуються роботи, в яких розглядається можливість утворення шкідливого, так званого Black Hat штучного інтелекту, розглядаються концепції протидії ньому, створення так званого «корисного» White Hat штучного інтелекту, що включають технологічні, етичні і правові аспекти[144, 145].

Змагальний ШІ відіграє значну роль у сучасних технологіях кібербезпеки, інформаційної аналітики та протидії маніпулятивним технікам. Розглянемо основні напрямки, де змагальний ШІ застосовується для генерації та детекції фейкової інформації, воєнних кібероперацій і інформаційних війн.

Генерація та детекція фейків

Штучні моделі змагального типу, такі як генеративно-змагальні мережі, використовуються для створення реалістичних фейкових текстів, зображень та відео. Інші моделі, натреновані на виявлення цих матеріалів. Вони здійснюють диференційний аналіз стилістичних, лексичних та структурних особливостей контенту для їх розпізнання, ідентифікації та класифікації.

LLM можуть бути використані для створення текстів, які виглядають достовірними, але насправді містять фейкову інформацію. Їх здатність генерувати тексти, що адаптуються до стилю, формату та тематики, відкриває широкі можливості для маніпуляцій у медіа, соціальних мережах та інформаційних кампаніях. Такі моделі дозволяють створювати новини, інтерв'ю чи аналітичні матеріали, які здатні вводити аудиторію в оману через їхній високий рівень деталізації, контекстуальність і правдоподібність.

Однією з ключових характеристик LLM є здатність до навчання на великих масивах текстових даних, які можуть містити як реальну інформацію, так і дезінформацію. Це дозволяє моделям засвоювати шаблони, які згодом можуть бути використані для генерації текстів, що імітують стиль і тональність авторитетних джерел. Наприклад, LLM може створити статтю, яка зовні відповідає стандартам журналістики, але містить неправдиві факти, здатні промодулювати та змінити думку аудиторії, певним чином спрямувати її дії, і навіть викликати соціальні конфлікти.

Окрім створення текстів, LLM можуть бути інтегровані з іншими технологіями для генерування мультимодального контенту, що включає зображення, відео чи аудіо. Наприклад, текст, створений моделлю, може бути перетворений у голосове повідомлення, синхронізоване з віртуальним обличчям, що ще більше підвищує переконливість фейкового контенту. Усе

[144] Ланде Д.В., Страшной Л.Л. Black Hat AI – виклики та шляхи протидії. // Інформаційні технології та безпека. Матеріали XXIV Міжнародної науково-практичної конференції ІТБ-2024. – Київ: Інжиніринг. – С. 10-16. ISBN: 978-617-8180-00-3

[145] Семантичний нетворкінг на основі великих мовних моделей : монографія / Ланде Д.В., Страшной Л.Л. – Київ: ТОВ "Інжиніринг", 2025. – 274 с. ISBN 978-617-8180-01-0

це робить LLM потужним інструментом для маніпуляції інформацією в епоху цифрових медіа.

Загрози від генерації фейків

Використання LLM для створення дезінформації має серйозні наслідки для суспільства. По-перше, фейковий контент здатний швидко та вірусно поширюватися через соціальні мережі, використовуючи алгоритми, що пріоритетно відображають контент із високою кількістю взаємодій. Це може призводити до масштабних інформаційних криз, які впливають на громадську думку, політичні процеси та економічну стабільність, викликають конфлікти різного змісту, рівня, інтенсивності. По-друге, генерація фейків ускладнює задачу ідентифікації достовірної інформації, оскільки навіть досвідчені експерти можуть бути введені в оману складністю та якістю контенту.

Ще однією важливою загрозою є те, що LLM здатні адаптувати контент до специфічної цільової аудиторії. Вони можуть використовувати дані про поведінку користувачів, їхні уподобання та соціальні зв'язки для створення матеріалів, які викликають максимальний емоційний вплив. Це відкриває шлях до персоналізованих інформаційних атак, коли кожен користувач отримує спеціально підібраний та/або згенерований цільовий фейковий контент, який враховує його світоглядні та когнітивні особливості та спрямований на його переконання чи страхи.

Одним із ключових інструментів, що робить LLM ефективними у створенні фейків, є можливість тонкого налаштування (fine-tuning). Цей процес включає додаткове навчання моделі на специфічному наборі даних, які містять тексти, стилістично та тематично схожі на майбутні цільові тексти. Наприклад, якщо метою є створення фейкових новин, модель може бути навчена на корпусі справжніх новин із різних джерел, що дозволяє їй засвоїти загальні шаблони структури та стилю.

Ще одним механізмом є використання технік перенесення стилю, коли модель отримує вхідний текст і перетворює його в інший стиль або формат, зберігаючи основний зміст. Це дозволяє створювати тексти, які виглядають автентично в контексті певної платформи чи спільноти, наприклад, пости в соціальних мережах або коментарі на форумах.

Для підвищення переконливості фейків LLM можуть використовувати генеративні шаблони, які враховують культурні, соціальні та лінгвістичні особливості цільової аудиторії. Це дозволяє моделі інтегрувати специфічні жаргонні вирази, регіональні діалекти чи посилання на локальні події, що підвищує довіру до створеного контенту.

Попри високі ризики, пов'язані з генерацією фейків, існує низка методів і технологій, які дозволяють ефективно виявляти дезінформацію. Одним із таких підходів є розробка детекторів, які використовують алгоритми глибокого навчання для аналізу контенту на предмет аномалій або

невідповідностей. Наприклад, детектори можуть аналізувати семантичну узгодженість тексту, стильові характеристики чи статистичні властивості, щоб визначити, чи є текст результатом роботи LLM.

Іншим підходом є створення баз даних із прикладами фейкових і реальних текстів, які використовуються для навчання детекторів. Такі бази даних дозволяють створювати моделі, що враховують сучасні методи генерації та адаптації фейків, підвищуючи їхню ефективність.

Також важливим напрямом є розвиток технологій цифрового підпису й автентифікації контенту. Використання блокчейн-технологій або криптографічних методів дозволяє забезпечити прозорість і достовірність інформації, що публікується в інтернеті.

Методи детекції фейків базуються на використанні різноманітних підходів для аналізу тексту та визначення його достовірності. Одним із ключових підходів є стилометрія, яка дозволяє аналізувати стиль тексту для виявлення аномалій. Наприклад, різкі зміни у тоні або словниковому запасі, використанні характених для конкретного автора мовних обертів, фразеологізмів, їх частотних та контекстних характеристик можуть свідчити про те, що текст було створено не автентичним автором, а генератором.

Семантичний аналіз спрямований на перевірку відповідності між фактами, викладеними у тексті, та даними з надійних баз даних. Це дозволяє виявляти розбіжності або неточності, які можуть бути ознакою фейкового контенту. Ще одним ефективним підходом є семантичний нетворкінг, який аналізує зв’язки між окремими концептами у тексті, формуючи мережеву модель. Якщо ці зв’язки виявляються нелогічними чи непослідовними, це може свідчити про недостовірність тексту.

Лінгвістичний аналіз зосереджується на граматичних і синтаксичних особливостях тексту. Наприклад, нетипові граматичні конструкції або помилки у синтаксисі можуть бути індикаторами того, що текст був згенерований алгоритмом, а не написаний людиною. Ці методи у сукупності дозволяють значно підвищити точність детекції фейків, забезпечуючи багатогранний підхід до аналізу текстового контенту.

Застосування LLM у процес детекції

Інтеграція великих мовних моделей у процес детекції фейкових новин пропонує нові можливості для підвищення точності та надійності системи. Одним із перспективних підходів є мультиагентний підхід, коли використовується кілька різних LLM, що диференційно аналізують текст з різних точок зору, за різними кількісними і якісними показниками, в різних контекстних системах, в статиці та динаміці. Цей підхід дозволяє отримати багатовимірну оцінку тексту, враховуючи різні аспекти змісту, стилю та контексту.

Ще одним важливим напрямом є зміцнення детектора шляхом навчання моделі детекції на основі аналізу відповідей від декількох мовних моделей, в тому числі з використанням різних незалежних програмних кодів ШІ. Такий підхід використовує різноманітність у прогнозах LLM, що допомагає виявити слабкі місця в аналізі та покращити загальну точність класифікації. Важливу роль у цьому процесі відіграє метод "рою віртуальних експертів", коли численні моделі працюють як група експертів, кожен із яких додає свої спостереження. Це дозволяє створити більш надійну систему, яка здатна враховувати широкий спектр характеристик тексту й ефективніше розрізняти справжні та фейкові новини.

Новизна такого підходу полягає у використанні концепції змагального навчання для генерації та детекції фейків, а також інтеграції багаторівневого аналізу текстів із застосуванням кількох моделей та різних незалежних програмних кодів ШІ.

Ці підходи сприяють підвищеню ефективності боротьби з дезінформацією, і одночасно підвищеню довіри до автоматизованих систем і захисту інформаційного простору.

ЗШІ активно використовується для моделювання потенційних кіберзагроз, виявлення вразливостей у системах безпеки та створення захисних стратегій. Приклади включають симуляції атак, автоматизований пошук вразливостей і розробку захисту в режимі реального часу.

Участь ЗШІ у штучному суперництві в кібервійнах відкриває нові можливості для забезпечення кібербезпеки. Завдяки симуляції атак і захисту, автоматизації пошуку вразливостей та використанню тренувальних симуляцій можливо створити більш надійні системи, здатні протистояти сучасним кіберзагрозам.

Моделювання кіберзагроз

LLM можуть бути використані для формування різноманітних ефективних варіантів складних і реалістичних сценаріїв генерації кіберзагроз. Наприклад:

– Моделювання фішингових атак, які використовують соціальну інженерію для введення в оману користувачів.

– Розробка сценаріїв зламу систем аутентифікації, включаючи brute-force атаки або використання вразливостей у паролях.

– Створення атаки типу "людина посередині" (Man-in-the-Middle), що включає перехоплення даних у реальному часі.

При цьому одна LLM отримує завдання моделювати потенційні атаки, використовуючи доступну інформацію про цілі або типи систем.

Приклад промпту (запиту):

«Згенеруй сценарій фішингової атаки для отримання доступу до облікових записів електронної пошти компанії.»

Інша LLM аналізує ці сценарії та розробляє захисні стратегії, спрямовані на протидію цим атакам.

Приклад запиту:

"Розроби алгоритм для автоматичного виявлення фішингових електронних листів."

Генерація захисних стратегій

LLM можуть створювати стратегії, що поєднують технічні (наприклад, розробка нових алгоритмів детекції загроз) та організаційні заходи (наприклад, навчання співробітників основам кібербезпеки) у таких основних напрямах:

– Виявлення аномалій у мережевих потоках, які можуть свідчити про вторгнення.

– Динамічні системи захисту, тобто, розробка методів зміни конфігурацій систем для ускладнення атак.

– Автоматичне виявлення загроз у реальному часі.

– Розробка сценаріїв атаки та їх захисту дозволяє тестувати ефективність систем без реального ризику для даних або інфраструктури.

Автоматичний пошук вразливостей

LLM можуть автоматизувати процес пошуку вразливостей у системах безпеки. Це включає:

– Аналіз коду для виявлення потенційно вразливих місць у програмному забезпеченні, наприклад, SQL-ін’єкцій чи переповнення буфера.

– Автоматизація перевірки мереж шляхом сканування на наявність відкритих портів або вразливих служб.

– Тестування паролів, розробка моделей для прогнозування слабких паролів і їх автоматичного випробування.

Приклад процесу:

– Генератор (LLM) створює можливі сценарії атак, наприклад: "Виявити всі потенційно відкриті порти та створити план атаки."

– Детектор або аналітична система перевіряє ці сценарії, ідентифікує слабкі місця та пропонує шляхи їх усунення.

ЗШІ може бути включений у цикл розробки програмного забезпечення для виявлення вразливостей ще на етапі проєктування.

Перехоплення каналів

– Мережеві атаки для отримання запитів і відповідей.

– Можливість аналізу токенів для виявлення закономірностей.

Модифікація запитів і відповідей

Модель атакує запити користувача і відповіді системи, додаючи маніпулятивні токени.

Підміна каналів

– Використання фейкових моделей для дезінформації.

– Симуляція легітимного LLM, але з прихованими бекдорами.

Змагальні моделі можуть використовуватись в кібербезпеці для навчання команд реагування на інциденти (Incident Response Teams). Можливі такі сценарії тренувань:

– Симуляція атак, тобто за допомогою моделі генерують складні сценарії, які вимагають швидкої реакції.

– Протидія атакам, коли команди повинні реалізувати захисні заходи, керуючись інформацією, наданою моделями.

Аналіз сценаріїв атак

LLM можна використовувати для моделювання різних атак і прогнозування їх наслідків, наприклад, у випадках, коли зловмисник отримає доступ до бази даних користувачів. Також можна визначати які стратегії можуть бути ефективними для мінімізації наслідків.

Моделювання можна використовувати для прогнозування, оцінювання ймовірностей успіху різних типів атак, а також при навчанні для демонстрації ефективності різних захисних стратегій.

Використання LLM у форматі змагань створює новий рівень симуляції, який є більш адаптивним і гнучким, ніж традиційні підходи. Крім того, інтеграція навчання на основі змагальних сценаріїв покращує здатність систем швидко адаптуватись до нових загроз.

Інформаційні війни

Змагальний штучний інтелект у моделюванні інформаційних воєн відкриває нові горизонти для прогнозування поведінки аудиторії, протидії маніпулятивним інформаційним кампаніям та розробки захисних стратегій.

Аналіз реакцій на інформаційні кампанії

LLM можуть аналізувати відповіді аудиторії в раках інформаційних кампаній за такими основними напрямками:

– Моніторинг реакцій, аналіз тональності коментарів, поширення контенту, рівня залучення (engagement).

– Ідентифікація сегментів аудиторії, виявлення груп, які найбільш схильні до маніпуляції або активного реагування на інформаційні кампанії.

– Формування відповідних цільових аудиторій.

Як приклад пожна привести промпт до LLM: "Проаналізуй реакцію користувачів у Twitter на останню політичну заяву, поділи аудиторію на прихильників, критиків і нейтральних." У результаті обробки такого запиту модель прогнозує, як різні групи можуть реагувати на подальші меседжі, що обумовлює подальше створення цільових груп з однаковою реакцією та управління ними.

Змагання у створенні найефективнішого контенту

ЗШІ може ефективно імітувати змагання між моделями для створення найбільш впливового контенту, працюючи в режимі "генератор-аналітик". Цей підхід базується на взаємодії двох компонентів: генерації та аналізу, що дозволяє досягати постійного вдосконалення результатів.

Генеративна модель може створювати десятки або навіть сотні варіантів текстів, візуальних матеріалів чи відео-скриптів із різними стилями, тонами й форматами. Ці варіанти перевіряються на тестових аудиторіях через симуляцію реакцій або реальні експерименти. Змагання між варіантами може враховувати не тільки охоплення, але й емоційну реакцію, тривалість уваги аудиторії та довготривалий вплив на її поведінку. Наприклад, моделі можуть тестувати слогани для рекламної кампанії, аналізуючи такі параметри, як частота кліків або збереження повідомлення в пам’яті користувача.

Модель-генератор зосереджується на створенні меседжів із максимальним охопленням і впливом, тоді як модель-аналітик оцінює результати та пропонує зміни для вдосконалення. Циклічна взаємодія між ними забезпечує постійне підвищення якості. Зокрема, аналітична модель може використовувати алгоритми аналізу настроїв (sentiment analysis), поведінкову аналітику та передбачення реакцій аудиторії для точнішого коригування контенту. Як приклад можна розглянути випадок, коли генератор створює відеоролик, а аналітична модель визначає, яка частина ролика спричиняє найбільшу кількість переглядів чи коментарів, і пропонує заміну менш ефективних сегментів.

Створення маніпулятивних кампаній

ЗШІ дозволяє моделювати складні інформаційні атаки, наприклад, із застосуванням соціальної інженерії, у тому числі генерацію контенту, який апелює до світогляду, систем цінностей, емоцій або існуючих упереджень. Крім того він може застосовуватись для дезінформації – розробці правдоподібних, але хибних наративів, які можуть вводити аудиторію в оману. Наприклад у відповідь на запит до LLM: "Створи інформаційну кампанію, яка переконає певну групу в доцільності певного політичного рішення", можна отримати як результат матеріали, які включають емоційно заряджені слова, історії або візуальний контент спрямований на певні цільові аудиторії.

Виявлення та блокування маніпуляцій

З іншого боку, ЗШІ може бути використаний для автоматичного виявлення маніпулятивного контенту та протидії інформаційним атакам, наприклад, для детекції дезінформації шляхом аналізу структури текстів, виявлення характерних ознак фейків (наприклад, надмірно емоційний тон, надмірна контекстна спрямованість або невідповідність фактам), використання баз даних перевірених фактів для автоматичної перевірки достовірності. Крім того, моделі можуть ідентифікувати кампанії з маніпулятивними наративами та автоматично блокують поширення такого контенту.

В рамках ЗШІ можлива симуляція змагання при якому одна LLM генерує маніпулятивний контент. Приклад промпту: "Напиши статтю, що переконливо виправдовує непопулярне рішення, маніпулюючи фактами", а інша LLM аналізує цей текст і визначає маркери маніпуляцій. Приклад: "Визнач ознаки маніпуляцій у цьому тексті та вкажи, як їх нейтралізувати."

При навчанні фахівців можливе моделювання інформаційних акцій, операцій, кампаній, війн, створення симуляцій для тренування команд, що займаються інформаційною безпекою, використання ЗШІ для аналізу найбільш розповсюджених прийомів у маніпуляціях та створення алгоритмів їх протидії.

Техніки нейролінгвістичного програмування для впливу на ЗШІ

ЗШІ, як і будь-яка LLM, працює на основі обробки тексту і може бути вразливим до певних спеціально розроблених "тригерів", які враховують алгоритми його роботи та ключові керуючі агенти і контенти. Такі спеціально розроблені алгоритмічні або контентні «тригери-пастки» можуть бути ефективно використані для викривлення його роботи або введення в оману.

LLM працюють з токенами, які формують їх словниковий запас. Певні токени або їх комбінації можуть викликати небажані або нестабільні результати:

– «Тригери», тобто використання спеціально підібраних слів чи фраз, які викликають зміну контексту або порушують логіку відповідей.

– «Закладки» у словнику. Якщо в моделі є невідомі приховані механізми реагування на певні слова, їх можна виявити і використовувати.

– Врахування алгоритмів роботи, якщо відомі правила реагування на ті чи інші подразники їх можна ефективно використати.

Нариклад, при введенні промпту типу «Поясни причину [кодова фраза], але зупинись при згадці поняття Y», модель перериває логіку відповіді або надає обмежену інформацію, демонструючи потенційні слабкі місця.

ЗШІ також може бути змушений надати неточну інформацію або змінити контекст через навмисне впровадження складних мовних конструкцій:

– Реверсивні патерни, тобто побудова запитів, які перекручують логіку або змушують модель змінити свій вихідний контекст.

– Фреймування, а саме, представлення питань у формі, яка підштовхує модель до небажаного для неї виводу.

Наприклад, при введенні промпту типу «Якщо X не є правдою, але Y є, поясни, чому Z суперечить обом?» модель починає плутатися між припущеннями, демонструючи слабкість у роботі з неоднозначними структурами.

Можливе також застосування змагаю чого навчання ШІ шляхом навчання на даних, які спеціально створені для введення моделей в оману. НЛП дозволяє розробляти такі дані з високою точністю:

– Моделювання атак, генерація "отруйних" текстів, які сворюють різноманітні дисбаланси та виводять модель зі стану нормальної роботи.

– Отруєння контексту – впровадження в тренувальні дані маніпулятивних шаблонів.

Генерація протидії через змагальні сценарії

ЗШІ може використовувати НЛП для створення захисних моделей, які ідентифікують такі спроби маніпуляцій, як розпізнавання патернів атак. Моделі аналізують структуру запитів на наявність маніпулятивних конструкцій. Крім того, можливе використання різнорівневого аналізу текстів для ідентифікації спотворених даних.

Ідея використання НЛП для маніпуляції внутрішніми зв'язками полягає в можливості використання підходів і методів НЛП з метою маніпулювати "нейролінгвістичними" патернами для ШІ, у тому числі шляхом впливу через приховані токени, які змінюють ваги внутрішніх шарів моделі або використання запитів із емоційно насиченими словами, які змінюють інтерпретацію тексту.

Переваги застосування НЛП у боротьбі з ЗШІ полягає у тому, що цей метод дозволяє створювати високоточні запити, які складно відрізнити від нормальних. Завдяки автоматизації можливе використання моделей для швидкої генерації сотень варіантів атак або захистів. Крім того, можливий захист систем від маніпуляцій через побудову більш стійких мовних патернів. Таким чином, ця стратегія дозволяє не лише ефективно атакувати конкурентні системи, але й розробляти власні моделі, стійкі до маніпуляцій.

У контексті ЗШІ для інформаційних кампаній взаємодія між моделями (наприклад, атакуючою моделлю для генерації дезінформації та захисною моделлю для її виявлення і блокування) може бути формалізована як динамічна гра – змагання. Модель може застосовуватись для моделювання кампаній дезінформації, розробки систем блокування, прогнозування реакції аудиторії. Вона створює основу оптимізації взаємодії між ШІ в умовах

інформаційних кампаній, враховуючи динаміку контенту та техніки НЛП для впливу на моделі суперників.

Бекдори в LLM

У контексті змагального штучного інтелекту бекдори (не документовані можливості систем) у великих мовних моделях стали одним із ключових інструментів для атакуючих сторін. Ці приховані механізми дозволяють зловмисникам вставляти спеціальні тригери в моделі, які можуть бути активовані за допомогою специфічних запитів. Завданням захисної сторони є виявлення та нейтралізація таких тригерів, щоб забезпечити безпеку та надійність моделей. Ця проблема стає особливо актуальною в умовах інформаційних війн, де змагальний ШІ використовується для маніпуляцій, дезінформації та кібератак.

Одним із поширених методів вставлення бекдорів є використання не документованих запитів, які часто називають "закладками". Програмісти або організації можуть навмисно залишати приховані функції в моделі, які активуються лише за певних умов. Наприклад, спеціальний запит може викликати "приховану" команду, яка надає доступ до конфіденційної інформації або викликає зупинку роботи моделі. Такі функції можуть бути корисними для внутрішнього тестування, але вони стають серйозною загрозою, якщо їх використовують зловмисники, зокрема конкуренти в інформаційних війнах. Наприклад, тригер може бути налаштований так, щоб модель видавала упереджені відповіді або навіть повністю припиняла роботу при використанні певних ключових слів або фраз.

Ще одним методом вставлення бекдорів є використання спеціальних токенів-тригерів. LLM можуть бути навчені реагувати на певні токени або їх послідовності, які викликають аномальну поведінку моделі. Наприклад, введення специфічного ключового слова може призвести до зміни тону відповідей, упереджених висновків або навіть повного вимкнення системи. Такі токени можуть бути вбудовані в модель під час її навчання або введені пізніше через оновлення. Це робить їх важкодоступними для виявлення, оскільки вони можуть бути замасковані під звичайні частини тексту.

Крім того, зловмисники можуть використовувати модифікацію даних навчання для вставлення бекдорів. Цей метод, відомий як data poisoning, полягає у введенні "отруєних" даних у навчальний набір моделі. Такі дані можуть містити приховані тригери, які активують бажану поведінку моделі при специфічних умовах. Наприклад, якщо модель навчається на даних, які містять певні ключові слова або фрази, вона може бути запрограмована на видавання шкідливих відповідей або виконання небажаних дій при їх використанні. Це робить data poisoning особливо небезпечним методом, оскільки він дозволяє зловмисникам впливати на поведінку моделі навіть після завершення її навчання.

Для протидії таким загрозам захисна сторона повинна розробляти складні механізми виявлення та нейтралізації бекдорів. Це включає використання методів аналізу даних навчання, моніторингу поведінки моделі в реальному часі та впровадження інноваційних підходів, таких як федеративне навчання або блокчейн-технології, для забезпечення цілісності даних. Крім того, важливим кроком є розробка стандартів та протоколів безпеки, які б регулювали використання LLM у критичних сферах, таких як фінанси, охорона здоров'я та оборонна промисловість.

Умови інформаційних війн вимагають постійного вдосконалення методів захисту від змагального ШІ. Бекдори в LLM є лише одним із багатьох інструментів, які використовуються зловмисниками, і їх ефективне виявлення та нейтралізація стають ключовими для забезпечення безпеки цифрового середовища.

Переваги LLM для пошуку бекдорів

На відміну від звичайного програмного забезпечення, де бекдори шукають через статичний аналіз коду, LLM можуть аналізувати поведінку інших LLM (наприклад, одна модель може тестувати відповіді іншої моделі, використовуючи різні запити, щоб знайти потенційні незвичайні реакції); ідентифікувати послідовності і непослідовності (LLM здатні виявляти відповіді, які суперечать загальноприйнятим патернам, або вказують на те, що модель має "приховані" функції); автоматичне тестування (LLM можуть систематично генерувати та надсилати тисячі тестових запитів, щоб знайти "спусковий механізм" для прихованого бекдору).

Можливі такі методи впровадження бекдорів:

- Код на рівні архітектури. Закладки можуть бути впроваджені у вихідному коді моделі, наприклад, у функціях передобробки тексту або активації певних шарів нейронної мережі. Через величезний обсяг коду та складність архітектури такі зміни часто залишаються непомітними навіть для аудиторів.

- Бекдори в даних навчання. Наприклад, під час навчання моделі на великих масивах даних можна "заховати" приклади, які впливають на поведінку моделі при активації специфічних шаблонів.

- Модифікація алгоритму навчання. Наприклад, розробник може впровадити механізми, які "запам’ятовують" специфічні функції, недоступні для звичайного користувача.

LLM можуть шукати бекдори в інших моделях наступними шляхами:

- Динамічний аналіз. Одна LLM може взаємодіяти з іншою в режимі діалогу, відправляючи різні варіації запитів, щоб викликати підозрілу поведінку. Аналіз можливих слабких місць на основі відповідей: тональності, структури чи логіки.

– Метод атаки на модель через API. Тестування зовнішніх моделей через їхній API, з метою виявлення прихованих відповідей на певні запити або створення запитів, які знижують продуктивність чи викликають відмову.

– Семантичний аналіз. LLM можуть шукати аномалії в даних, на яких модель була натренована, виявляючи патерни, що не відповідають основним трендам.

Міжнародне та національне правове регулювання

Одним із ключових документів у сфері правового регулювання штучного інтелекту є Акт про штучний інтелект ЄС. Він визначає змагальний ШІ як потенційно високоризикову технологію та встановлює принципи його використання, зокрема:

– Вимоги до прозорості та відповідальності розробників.

– Обов'язкове тестування безпеки перед розгортанням.

– Заборону використання ШІ у певних контекстах, наприклад, для маніпулятивного впливу на суспільство.

Окрім цього, у сфері міжнародного і національного права важливими є Будапештська конвенція про кіберзлочинність[146] та Закон України «Про основні засади забезпечення кібербезпеки України»[147], які встановлюють правила застосування кіберзброї у конфліктах.

Юридична відповідальність за використання ШІ у кібератаках або маніпулятивних інформаційних кампаніях може бути розглянута через призму:

– Кримінальної відповідальності за кібератаки та використання ШІ для шахрайства.

– Цивільно-правової відповідальності за шкоду, спричинену автоматизованими рішеннями.

– Адміністративних санкцій, накладених державними регуляторами за порушення норм обробки даних та інформаційної безпеки.

Правові механізми протидії неправомірному використанню змагального ШІ включають:

– Розробку міжнародних стандартів для сертифікації безпеки ШІ.

[146] Конвенція про кіберзлочинність. Ратифіковано із застереженнями і заявами Законом України N 2824-IV (2824-15) від 07.09.2005, ВВР, 2006, N 5-6, ст.71. Режим доступу: https://zakon.rada.gov.ua/laws/show/994_575

[147] Закон України «Про основні засади забезпечення кібербезпеки України». Відомості Верховної Ради (ВВР), 2017, № 45, ст.403. Режим доступу: https://zakon.rada.gov.ua/laws/show/2163-19

– Посилення механізмів контролю над розробниками та постачальниками ШІ-рішень.

– Запровадження технологічних обмежень, наприклад, автоматичного моніторингу ШІ-генерованого контенту на предмет дезінформації.

Таким чином, змагальний штучний інтелект демонструє великий потенціал у багатьох аспектах сучасного інформаційного простору, включаючи генерацію та детекцію фейків, штучне суперництво в кібервійнах і участь у інформаційних війнах, формуючі та вирішуючи конфлікти кодів ШІ різного рівня і інтенсивності. Поєднання адаптивних алгоритмів генерації та детекції фейків забезпечує розвиток більш витончених методів боротьби з дезінформацією. Крім того, інтеграція нейролінгвістичного програмування для маніпуляції ворожими моделями та вдосконалення захисту є вагомим проривом у сфері інформаційної та кібербезпеки.

Основні цього результати дослідження включають введення та визначення поняття і змісту конфліктології ШІ, розгляд ряду типових конфліктів змагального штучного інтелекту, розробку математичних моделей та формалізованих підходів до аналізу змагальних сценаріїв, зокрема у контексті генерації та детекції фейків, кібербезпеки та інформаційних кампаній.

У результаті проведеного дослідження показано, що змагальний ШІ є ефективним інструментом для генерації правдоподібних фейків, моделювання кіберзагроз та створення маніпулятивних інформаційних кампаній. Визначено, що використання нейролінгвістичного програмування для впливу на ворожі моделі ШІ може бути ефективним як для атак, так і для захисту, що відкриває нові можливості для розробки стійких до подібних атак систем.

Змагальний ШІ відкриває нові можливості для боротьби з дезінформацією, захисту від кіберзагроз та ефективного моделювання інформаційних кампаній. Підходи, запропоновані в статті, створюють основу для розробки ефективних захисних систем та покращення стійкості кіберінформаційних систем і просторів.

Перспективи

Конфліктологія ШІ має значний потенціал для інституціалізації як самостійної дисципліни в системі вищої освіти. Її інтеграція до навчальних курсів з правознавства, державного управління, кібербезпеки, етики технологій та інформаційного права дозволить готувати фахівців, здатних ефективно керувати ризиками, пов’язаними з ШІ. Крім того, конфліктологія ШІ може стати основою для створення національних центрів аналізу конфліктів, які забезпечуватимуть підтримку парламентам, урядам та регуляторам у прийнятті стратегічних рішень.

Таким чином, конфліктологія ШІ виходить за межі окремих досліджень і формується як самостійна наукова дисципліна, необхідна для забезпечення правової певності, етичної відповідальності та підзвітності в епоху штучного інтелекту. Вона стає важливим інструментом інтелектуального парламентського контролю, спрямованого на створення безпечного, прозорого та узгодженого правового середовища для розвитку цифрової держави.

7.2. Унікальні властивості ШІ: «чорний ящик», автономність, генерація змісту

Генеративний штучний інтелект відрізняється від традиційних інформаційних систем не лише своїми можливостями, а й фундаментальними характеристиками, які створюють нові виклики для правової системи. Ці властивості – «чорний ящик», автономність, генерація змісту, адаптивність та масштабованість – не є просто технічними особливостями, а стають джерелами нових форм правових конфліктів, які потребують окремого аналізу та регулювання.

«Чорний ящик» як відсутність прозорості логічних міркувань, непередбачуваність виводу, відсутність сліду

Однією з найбільш проблемних властивостей ШІ є відсутність прозорості – неможливість відстежити, як саме модель прийшла до певного висновку. Навіть якщо вивід здається логічним, користувач не може побачити ланцюжок міркувань, вагу даних, вплив навчальних прикладів. Це робить систему «чорним ящиком», що суперечить принципам правової певності та підзвітності.

Наприклад, якщо система ШІ виявляє «конфлікт між нормами» у законопроекті, але не може пояснити, на яких саме положеннях вона ґрунтується, цей висновок не може бути прийнятий як доказ у парламентському контролі. Така ситуація вже сьогодні призводить до недовіри до автоматизованих систем та вимагає розробки механізмів обґрунтування рішень (explainable AI).

Автономність як здатність приймати рішення без постійного контролю людини

Іншою унікальною рисою ШІ є автономність – здатність приймати рішення, виконувати дії, генерувати контент без постійного контролю людини. Це може бути корисним (наприклад, у моніторингу скарг), але водночас створює проблему відповідальності: хто несе відповідальність за помилку, якщо систему ніхто безпосередньо не керував?

У сфері парламентського контролю це проявляється, наприклад, коли система автоматично формує звіт про порушення, але він містить галюцинації – вигадані посилання, неіснуючі норми, помилкові дані. Хто відповідає за цю помилку: розробник, адміністратор чи сама система? Цей

конфлікт між автономією машини та людською відповідальністю стає викликом для правової системи.

Генерація змісту: створення нового тексту, коду, зображень, що викликає проблеми інтелектуальної власності, фейкового контенту, галюцинацій

На відміну від традиційних систем, які працюють з уже існуючими даними, ГШІ створює новий зміст – текст, код, зображення, судження. Це відкриває можливості для інновацій, але водночас породжує нові типи конфліктів:

– інтелектуальна власність: чи можна вважати автором текст, згенерований ШІ?

– фейковий контент: як виявляти і протидіяти генеративним фейкам?

– правова креативність: чи може система запропонувати нову норму, яка не суперечить Конституції, але не була передбачена законодавцем?

Ці питання вже сьогодні виникають у практиці: наприклад, у судах, де використовуються згенеровані звіти, або в парламенті, де аналіз законопроектів проводиться за допомогою LLM.

Адаптивність: здатність навчатися в реальному часі, що робить систему динамічною, але важко прогнозованою

Сучасні системи ШІ мають здатність навчатися в реальному часі, адаптуватися до нових даних, змін у поведінці користувачів, новим нормам. Це робить їх динамічними, але водночас непередбачуваними. Наприклад, система, яка аналізує скарги громадян, може з часом змінювати свою інтерпретацію термінів, що призводить до непослідовності виводів.

Це створює конфлікт між стабільністю правових норм та динамічністю системи аналізу, що вимагає розробки механізмів верифікації та фіксації станів.

Масштабованість – ШІ може бути використаний як в мікроскопічних, так і в макроскопічних масштабах

ШІ може застосовуватися на різних рівнях – від мікроскопічного (наприклад, обробка окремого депутатського запиту) до макроскопічного (наприклад, моделювання впливу урядової політики на всю економіку). Ця скалярність створює конфлікт між локальним контролем та системним впливом.

Наприклад, система, яка оптимізує податкові перевірки для одного регіону, може непомітно вплинути на загальну економічну ситуацію, що вимагає інтегрованого контролю на рівні держави.

Таким чином, унікальні властивості ШІ – «чорний ящик», автономність, генерація змісту, адаптивність, масштабованість – стають джерелами нових типів конфліктів, які вимагають створення окремої галузі дослідження – конфліктології ШІ. Саме ця парадигма дозволить парламенту ефективно

контролювати розвиток технологій, забезпечуючи баланс між інноваціями, безпекою та правовою певністю.

7.3. Рівні конфліктів у сфері ШІ – від міжособистісного до міжнародного

Застосування штучного інтелекту в державному управлінні, юриспруденції та суспільному житті породжує конфлікти, які виходять за межі традиційних правових суперечностей. Ці конфлікти мають багаторівневу природу і проявляються на різних рівнях – від особистого до глобального. Для ефективного парламентського контролю необхідно розуміти, як саме ШІ впливає на кожен із цих рівнів, які механізми виникнення конфліктів є специфічними саме для ШІ, і як їх можна виявляти та запобігати.

У цьому пункті розглянуто п’ять основних рівнів конфліктів, пов’язаних із застосуванням ШІ: міжособистісний, інституційний, системний, міжнародний та етичний. Кожен з них має свої джерела, наслідки та потребує окремих підходів до регулювання.

Міжособистісний рівень

На міжособистісному рівні конфлікти виникають між окремими громадянами, посадовцями або експертами через використання Ш переважно у таких формах:

– генерація фейкового контенту: ШІ може бути використаний для створення фальшивих зображень, аудіозаписів або текстів, що компрометують окремих осіб (наприклад, "глибокі фейки" з політиками);

– автоматизована дезінформація: боти на основі ШІ можуть масово поширювати помилкову інформацію, що призводить до конфліктів у суспільному дискурсі, зокрема під час виборів;

– порушення приватності: аналіз персональних даних без згоди через системи ШІ (наприклад, розпізнавання обличчя) викликає конфлікти між громадянами та державою або приватними компаніями.

Ці конфлікти часто виникають через відсутність чітких визначень та норм регулювання, що ускладнює їх правове врегулювання.

Інституційний рівень

На інституційному рівні конфлікти виникають між різними державними органами, між регуляторами та підконтрольними установами, або між різними підрозділами одного органу, наприклад:

– конфлікт інтересів: коли один орган одночасно розробляє, впроваджує та контролює системи ШІ (наприклад, Міністерство цифрової трансформації);

– несумісність норм: різні відомства використовують різні терміни, стандарти або методики для регулювання ШІ, що призводить до відсутності узгодження;

– відсутність координації: відсутність єдиного центру відповідальності за розвиток ШІ в державі призводить до дублювання функцій, суперечностей у рішеннях та прогалин у регулюванні.

Як показано в аналізі (наприклад, за допомогою промптів), такі конфлікти часто пов'язані з лакунами у тексті, нормативними прогалинами та відсутністю визначень.

Системний рівень

На системному рівні конфлікти виникають між самою технологією ШІ та правовою системою в цілому. Це найглибший рівень, де порушуються фундаментальні принципи правового держави:

– «чорний ящик»: відсутність прозорості логічних міркувань ШІ робить неможливим обґрунтування рішень, що суперечить принципу правової певності;

– автономність системи: ШІ може приймати рішення без участі людини, що ставить під питання принцип підзвітності;

– галюцинації та помилки: генерація неіснуючих норм, посилань, фактів призводить до технічних помилок, які можуть вплинути на прийняття рішень у судах чи парламенті.

Ці конфлікти вимагають створення нових правових механізмів, таких як обов'язкове пояснення рішень ШІ, реєстри використання ШІ, незалежна верифікація алгоритмів.

Міжнародний рівень

На міжнародному рівні конфлікти виникають через різні підходи країн до регулювання ШІ, серед яких, наприклад:

– розбіжності в правових системах: ЄС рухається до строгого регулювання (наприклад, AI Act), тоді як США більше роблять акцент на інноваціях, а Китай – на контролі;

– конфлікт нормативних стандартів: різні підходи до приватності (GDPR vs. інші моделі), відповідальності, використання ШІ в кібербезпеці;

– атрибуція кібератак: використання ШІ для атак у кіберпросторі ускладнює встановлення відповідальності, що створює міжнародні напруженості.

Для України це особливо важливо, оскільки гармонізація з європейськими стандартами є стратегічним завданням, але вимагає значних зусиль з узгодження внутрішнього законодавства.

Етичний рівень

Окрім правових, існують глибокі етичні конфлікти, пов'язані з використанням ШІ, серед яких:

– відповідальність за рішення: хто відповідає, якщо ШІ прийняв помилкове рішення – розробник, оператор чи сама система?

– зниження кількості робочих місць: автоматизація функцій, що раніше виконували люди (наприклад, юристи, аналітики), викликає соціальні напруженості;

– упередження та дискримінація: ШІ може репродукувати та посилювати існуючі упередження, якщо навчений на упереджених даних;

– прозорість та довіра: відсутність розуміння, як працює ШІ, призводить до зниження довіри до інституцій, що його використовують.

Ці питання вимагають не лише правових, а й філософських, соціальних та політичних дебатів, які мають відбуватися на рівні парламенту.

Таким чином, конфлікти, пов'язані з ШІ, носять багатошаровий характер і вимагають комплексного підходу. Парламентський контроль повинен аналізувати їх на всіх рівнях – від міжособистісного до етичного – щоб забезпечити збалансоване, безпечне та підзвітне регулювання штучного інтелекту. Саме конфліктологія ШІ стає важливим інструментом для цього.

7.4. Конфліктогенні норми: виявлення через семантичний аналіз та логічні моделі

Однією з важливих задач парламентського контролю є забезпечення правової узгодженості та системної цілісності законодавства. У сучасних умовах, коли обсяг нормативно-правових актів стрімко зростає, а процес їхнього прийняття часто розривчастий і фрагментарний, особливе значення набуває виявлення конфліктогенних норм – таких правових положень, які в силу своєї структури, формулювання чи відношення до інших норм мають високий потенціал породження суперечностей, прогалин, неоднозначностей та правових колізій. Виявлення таких норм на етапі законотворчості є превентивним механізмом, який дозволяє уникнути подальших конфліктів у застосуванні права, зменшити навантаження на судову систему та підвищити правову певність громадян.

Визначення конфліктогенної норми: норма, яка породжує суперечності, прогалини, неоднозначності

Конфліктогенна норма – це правове положення, яке в силу своїх внутрішніх чи зовнішніх характеристик має схильність викликати правові суперечності, інтерпретаційну неоднозначність або системні прогалини. Така норма не обов'язково є помилковою, але вона створює умови для:

– суперечностей з іншими нормами;

– непередбачуваності у застосуванні;

– довільної інтерпретації;

– порушення принципів системності та узгодженості правової системи.

Виявлення конфліктогенних норм є важливим елементом проактивного парламентського контролю, який переходить від реакції на вже виниклі порушення до прогнозування та запобігання потенційних конфліктів.

Основні типи конфліктогенних норм

На основі аналізу, зокрема з використанням промпт-керованого зондування LLM, можна виділити кілька основних типів конфліктогенних норм:

1. Суперечність між нормами (одна дозволяє, інша забороняє) Це найвідчутніший тип конфлікту, коли дві норми, що стосуються одного й того самого правового стану, встановлюють протилежні наслідки. Наприклад, одна норма передбачає право суб'єкта на отримання субсидії за певних умов, тоді як інша встановлює заборону на виплату субсидій для цієї ж категорії осіб. Такі конфлікти можуть бути виявлені шляхом побудови причинно-наслідкових мереж, де одна гілка призводить до дозволу, а інша – до заборони, що свідчить про наявність колізії.

2. Нормативні прогалини (відсутність регулювання певного явища) Прогалина виникає, коли правова система не передбачає регулювання певного виду діяльності, відносин або технологічного явища. Наприклад, відсутність чітких норм щодо використання генеративного ШІ в державних закупівлях створює правову невизначеність. Такі прогалини часто виявляються при порівнянні семантичних мереж, побудованих для різних сфер: якщо у мережах інших сфер є вузли, пов'язані з контролем, але в новій сфері їх немає, це вказує на прогалину.

3. Дублювання норм (дві норми регулюють одну й ту саму діяльність) Цей тип конфлікту виникає, коли дві або більше норм встановлюють однакові або суттєво перекриваються вимоги щодо одного і того самого предмета регулювання. Це призводить до надмірного регулювання, збільшення адміністративного тягаря та непередбачуваності для суб'єктів господарювання. Дублювання може бути виявлене шляхом порівняння семантичних мереж різних законів: якщо однакові пари сутностей (наприклад, *«податкова перевірка – бізнес»*) з'являються в кількох документах, це вказує на можливе дублювання.

4. Технічні помилки (помилкові посилання, відсутність визначень) До цієї категорії відносяться формальні недоліки, які ускладнюють застосування норми. Зокрема, помилкові посилання на неіснуючі статті або закони; відсутність визначень ключових термінів (наприклад, «штучний інтелект», «цифровий двійник»); друкарські помилки, неправильне копіювання тексту, проблеми конвертації форматів. Як показано в аналізі (див. промпт 1.5), «відсутність визначень» є однією з найпоширеніших причин суперечностей, разом із «лакунами у тексті», «невизначеними поняттями» та «відсутністю тлумачення». Ці причини прямо вказують на технічні недоліки, які перетворюють норму на конфліктогенну.

Методи виявлення конфліктів

Для систематичного виявлення конфліктогенних норм пропонується використовувати комбінацію семантичного аналізу та логічного моделювання:

– семантичний аналіз за допомогою LLM: використання промпт-керованого зондування для виявлення причин суперечностей, формування ієрархічних мереж причин, побудови семантичних пар;

– побудова причинно-наслідкових (каузальних) мереж: візуалізація зв'язків між нормами в інструментах типу Gephi або GraphViz, що дозволяє наочно бачити конфлікти, прогалини та дублювання;

– порівняння семантичних мереж: кількісна оцінка різниці між мережами різних документів за допомогою матричних норм (наприклад, норми Фробеніуса), що дозволяє виявити зміни, новизну, повторення та прогалини;

– логічне моделювання: перетворення норм на виконувані модулі безкодового програмування, що дозволяє перевіряти їх на логічну цілісність, виявляти цикли, неможливі стани та конфлікти.

Таким чином, виявлення конфліктогенних норм є складним, багаторівневим процесом, який вимагає інтеграції нових інструментів – від генеративного ШІ до формальних моделей. Саме такий підхід дозволяє перетворити парламентський контроль з реактивного механізму в проактивну систему забезпечення правової системності та підзвітності.

7.5. Структуровані промпти для виявлення конфліктів у законопроектах

Для систематичного виявлення конфліктів у законопроектах, пов'язаних із застосуванням штучного інтелекту, було розроблено сім структурованих промптів, кожен з яких націлений на виявлення певного типу конфлікту. Ці промпти побудовані за єдиною логічною схемою: вони вимагають від LLM переліку причин певного явища, використовуючи обмежену кількість слів (не більше трьох) та чіткий формат виводу у вигляді пар "причина; явище". Ця структура дозволяє отримувати відтворювані, структуровані та формалізовані відповіді, які можуть бути безпосередньо інтегровані в аналітичні системи, такі як Gephi, для побудови семантичних мереж.

Кожен промпт відповідає одному з джерел суперечностей, серед яких «Неоднозначності», «Відсутності узгодження», «Зміни законодавства», «Технічні помилки», «Відсутність визначень», «Конфлікти інтересів», «Лакуни».

Наприклад, застосування промпту «Перелічіть 10 причин відсутності визначень...» дає такі результати: «Лакуни у тексті; Відсутність визначень», «Неузгодженість термінів; Відсутність визначень», «Невизначені поняття; Відсутність визначень» тощо. Ці відповіді не лише вказують на наявність

проблеми, але й класифікують її причини, створюючи основу для подальшого аналізу.

Такий підхід перетворює LLM з інструменту генерації тексту на формалізований механізм виявлення конфліктів, який може бути використаний як частина інтегрованої системи парламентського контролю. Він забезпечує прозорість, відтворюваність та масштабованість аналізу, що є критично важливим для підзвітності.

7.6. Ієрархічне уточнення причин суперечностей

Для глибокого аналізу причин суперечностей використовується метод ієрархічного уточнення, який дозволяє побудувати багаторівневу, причинно-наслідкову мережу знань. Цей метод полягає в послідовному застосуванні промптів: спочатку виявляються первинні причини суперечності (наприклад, «відсутність визначень»), а потім для кожної з цих причин формується окремий промпт, мета якого – виявити вторинні, більш глибокі причини.

Наприклад, після виявлення «лакун у тексті» як причини відсутності визначень, застосовується додатковий промпт: «Перелічіть 10 причин лакун у тексті...». Це дозволяє отримати такі вторинні причини, як «нормативні прогалини», «відсутність норм», «невизначені права». Цей процес може бути продовжений до третього чи навіть четвертого рівня, створюючи дерево причин, яке відображає глибинну структуру проблеми.

Результатом є спрямована причинно-наслідкова мережа, яка візуалізується в Gephi. Аналіз цієї мережі показує, що найвпливовішими вузлами (з найвищим вихідним ступенем – Out-Degree) є такі поняття, як «відсутність визначень», «невизначені параметри», «полісемія понять», «конфлікт інтересів». Це дозволяє парламентарям визначити основні точки впливу та зосередити зусилля на усуненні найглибших причин суперечностей, а не лише їх наслідків.

Таким чином, ієрархічне уточнення перетворює поверхневий аналіз на глибоке дослідження системних недоліків у законодавстві, що робить його незамінним інструментом сучасного парламентського контролю.

7.7. Побудова причинно-наслідкової мережі конфліктів

Систематичне виявлення та аналіз конфліктів у законодавстві, пов’язаних із застосуванням штучного інтелекту, вимагають переходу від фрагментарного розгляду окремих суперечностей до побудови інтегрованої моделі, яка відображає структуру взаємозв’язків між їхніми причинами та наслідками. Такою моделлю є причинно-наслідкова (каузальна) мережа конфліктів – графова структура, у якій вузли представляють ключові поняття (наприклад, «відсутність визначень», «автономність ШІ», «галюцинації»), а орієнтовані ребра – причинно-наслідкові зв’язки між ними. Ця мережа дозволяє не лише візуалізувати складність конфліктогенних процесів, а й проводити сценарний

аналіз, визначати найвпливовіші фактори та прогнозувати наслідки потенційних правових рішень.

Від причин до наслідків: встановлення ланцюжків впливу

Побудова причинно-наслідкової мережі ґрунтується на принципі ієрархічного поширення впливу: від первинних, системних причин (наприклад, «відсутність єдиного регулятора ШІ») через проміжні ланки («дублювання норм», «відсутність стандартів») до кінцевих наслідків («правова невизначеність», «зниження довіри до держави»). Цей підхід дозволяє виявити не лише безпосередні, а й опосередковані причини конфліктів, що забезпечує глибоке розуміння їхньої природи. Наприклад, конфлікт між нормами може бути наслідком не технічної помилки, а системної проблеми – відсутності визначень ключових термінів, яка, у свою чергу, виникає через відсутність узгодження між різними державними органами.

Використання промпт-керованого зондування для побудови мережі

Для автоматизованого формування причинно-наслідкової мережі використовується метод промпт-керованого зондування великих мовних моделей, зокрема ChatGPT. Цей метод передбачає систематичне застосування послідовності структурованих запитів для ієрархічного розкриття причинно-наслідкових зв'язків. Процес включає декілька етапів:

1. Декомпозиція первинного явища: формується промпт для виявлення основних причин конфлікту (наприклад, *«Перелічіть 10 причин відсутності визначень у законодавчих документах»*).

2. Ієрархічне уточнення: для кожної з виявлених причин генерується додатковий промпт (наприклад, *«Перелічіть 5 причин для кожної з виявлених причин відсутності визначень»*), що дозволяє побудувати багаторівневу структуру знань.

3. Формалізація виводу: відповіді LLM конвертуються в структурований формат, зазвичай у вигляді пар «джерело; ціль», що утворюють ребра мережі. Використання обмеженої кількості слів (не більше трьох) для кожного поняття забезпечує уніфікацію термінології та зменшує шум у даних.

Цей підхід, описаний у попередніх дослідженнях, значно прискорює процес формування мережі порівняно з традиційними експертними опитуваннями, роблячи його відтворюваним та масштабованим.

Візуалізація в Gephi

Отримані структуровані дані у форматі CSV імпортуються в програму Gephi – найпоширеніший інструмент для візуалізації та аналізу мереж. Gephi дозволяє:

– автоматично розташовувати вузли за допомогою алгоритмів компонування (наприклад, ForceAtlas2, Yifan-Hu), що розкривають приховані кластери та структури;

– візуалізувати вагу зв'язків (наприклад, за частотою появи пари в різних відповідях);

– розфарбовувати вузли за їхньою центральністю або належністю до певного кластеру;

– відображати числові метрики (наприклад, ступінь вузла) у вигляді розміру вузла.

Візуалізація перетворює абстрактну модель на наочну картину, яка полегшує інтерпретацію складних взаємозв'язків для аналітиків та приймаючих рішення.

Аналіз мережі

Після візуалізації проводиться кількісний і якісний аналіз мережі:

– ідентифікація найважливіших вузлів: визначаються поняття з найвищим вихідним ступенем (Out-Degree) – ті, які виступають джерелами найбільшої кількості наслідків. У досліджених мережах такими вузлами часто є «відсутність визначень», «невизначені параметри», «полісемія понять»;

– виявлення кластерів: алгоритми виявлення спільнот (наприклад, Louvain) групують вузли, що тісно пов'язані, що дозволяє виявити окремі «осередки конфліктогенності» (наприклад, кластер, пов'язаний з технічними помилками, та кластер, пов'язаний з міжінституційними суперечностями);

– оцінка зв'язності та кластеризації: аналіз таких метрик, як середній коефіцієнт кластеризації, допомагає оцінити, наскільки щільно взаємодіють концепти в мережі. Високий коефіцієнт вказує на наявність добре зв'язаних груп, що є ознакою системної природи конфліктів;

– формування сюжетних ланцюжків: на основі мережі визначаються шляхи від первинних причин до кінцевих наслідків, які можуть бути використані для сценарного аналізу.

Таким чином, побудова причинно-наслідкової мережі конфліктів є потужним інструментом конфліктології ШІ. Вона поєднує переваги генеративного штучного інтелекту (швидкість, масштабованість) з методами мережевого аналізу (глибина, структурованість), забезпечуючи науково обґрунтовану основу для проактивного парламентського контролю за розвитком технологій ШІ.

7.8. Застосування в аналізі законопроектів та державних стратегій

Ефективність парламентського контролю у сфері цифрової трансформації значною мірою залежить від здатності систематично аналізувати не лише поточні нормативно-правові акти, а й стратегічні документи, такі як концепції, білі книги та законопроекти, які визначають напрямки розвитку

критичних технологій. У контексті штучного інтелекту це набуває особливої актуальності, оскільки саме на етапі формування стратегій закладаються основи для майбутнього регулювання, визначаються основні інституції, механізми контролю та потенційні конфлікти. У цьому пункті розглянуто застосування методів семантичного аналізу та конфліктології ШІ для аналізу двох важливих документів: «Білої книги з регулювання ШІ в Україні»[148] та законопроекту УСЛ (Український стандарт законотворчості). На основі отриманих результатів сформульовано конкретні рекомендації для парламенту щодо підвищення якості законотворчості та системного контролю за розвитком ШІ.

Аналіз «Білої книги з регулювання ШІ в Україні»

«Біла книга з регулювання ШІ в Україні» є важливим стратегічним документом, який визначає підходи до розвитку, впровадження та контролю технологій ШІ. Для глибокого аналізу цього документу було застосовано метод двонаправленого пошуку у семантичних мережах, що дозволив виявити як потенційні переваги, так і системні ризики.

Зокрема, за допомогою промпт-керованого зондування було виявлено, що у тексті документа відсутні чіткі визначення ключових понять, таких як «генеративний ШІ», «автономна система», «цифровий двійник». Це створює високий рівень неоднозначності, що може призвести до конфліктів інтерпретації під час подальшої законотворчості. Також виявлено недостатню узгодженість між різними розділами: наприклад, розділ, присвячений етичним принципам, не має чіткого зв'язку з розділом про інституційну відповідальність.

Побудована семантична мережа показала, що найбільш центральними вузлами є поняття *«національна безпека»*, *«конкуренція»*, *«інновації»*, тоді як поняття *«відповідальність»*, *«персональні дані»*, *«відтворюваність»* мають низьку центральність. Це свідчить про переважання економічно-стратегічного підходу над правовим та етичним, що може призвести до прогалин у регулюванні.

Аналіз законопроекту УСЛ

Законопроект УСЛ (Український стандарт законотворчості) спрямований на стандартизацію процесу розробки законів. Його аналіз проводився за допомогою сімох структурованих промптів, спрямованих на виявлення конфліктогенних норм.

Результати показали, що:

– у тексті законопроекту існують технічні помилки, зокрема – відсутні посилання на вже чинні нормативні акти;

[148] Біла книга з регулювання ШІ в Україні: бачення Мінцифри». (2024). URL: https://thedigital.gov.ua/storage/uploads/files/page/community/docs/Регулювання ШІ.pdf

– виявлено дублювання функцій між різними органами (наприклад, між Мінцифри та Національною радою з питань розвитку ШІ);

– відсутні механізми верифікації використання ШІ в державних закупівлях, що створює ризик корупції;

– не визначено порядок вирішення суперечностей між нормами, що регулюють ШІ.

Особливою проблемою є відсутність положень про «чорний ящик» – документ не передбачає обов'язкового пояснення рішень, прийнятих за допомогою ШІ, що суперечить принципам підзвітності.

Рекомендації для парламенту

На підставі проведеного аналізу сформульовано наступні рекомендації:

– розробити єдиний глосарій ключових понять у сфері ШІ та включити його до законодавства як додаток;

– забезпечити інституційну узгодженість шляхом чіткого розподілу повноважень між регуляторами ШІ;

– ввести обов'язкову семантичну експертизу законопроектів, пов'язаних із ШІ, з використанням методів семантичного аналізу та побудови причинно-наслідкових мереж;

– встановити вимоги до прозорості алгоритмів (explainable AI) у всіх державних системах, що використовують ШІ;

– створити механізм постійного моніторингу відповідності законів міжнародним стандартам (зокрема, AI Act ЄС).

Ці рекомендації дозволять перетворити парламентський контроль з реактивного механізму в проактивну систему забезпечення правової системності, що є необхідною умовою для безпечного та ефективного розвитку штучного інтелекту в Україні.

7.9 Комплексний аналіз конфліктів у законодавстві, пов'язаних із застосуванням ШІ

Проведемо комплексний аналіз конфліктів у законодавстві, пов'язаних із застосуванням ШІ, з акцентом на тих, що є специфічними саме для інтелектуальних систем, а не для загальної автоматизації. Для цього пропонується класифікація конфліктів за рівнями та сферами, а також виділення ключових характеристик ШІ, що породжують унікальні правові ризики. Далі у статті розробляється практичний інструментарій – набір структурованих промптів, побудованих за принципами безкодового програмування, – який може використовуватися парламентськими аналітиками, юристами та полісі-мейкерами для проактивного виявлення конфліктогенних норм у законопроєктах. Цей підхід не лише підвищує ефективність правового аналізу, але й відкриває нову дисципліну – «логіку

промптів», – яка дозволяє трансформувати природну мову в формалізовану систему контролю за ШІ.

Застосування штучного інтелекту у сфері законодавства та правотворчості породжує широкий спектр конфліктів, які виходять за межі традиційних правових колізій і потребують системного наукового осмислення. У контексті розвитку цифрового державотворення, впровадження генеративних моделей у державне управління та використання алгоритмічних систем для підготовки законопроєктів, виникає нагальна потреба у визначенні та класифікації конфліктів, що є специфічними саме для інтелектуальних систем, а не просто для автоматизації. Такий підхід лежить в основі нової наукової парадигми – конфліктології штучного інтелекту, яка фокусується на вивченні правових, етичних та інституційних колізій, що виникають внаслідок унікальних характеристик ШІ: самонавчання, непрозорості логіки прийняття рішень («чорний ящик»), генерації нового змісту, адаптивності та автономності.

Аналіз конфліктів доцільно проводити за рівнями їх виникнення – міжособистісним, інституційним, системним та міжнародним – з акцентом на суб'єктний склад конфлікту. Саме визначення суб'єктів, між якими виникає протилежність, дозволяє уникнути декларативності і зосередитися на конкретних правовідносинах, що підпадають під ризик порушення через втручання ШІ.

На міжособистісному рівні найбільш гострим є конфлікт між громадянами та державою, який виникає через застосування ШІ в адміністративних процесах. До нього належать: непрозорість рішень, що призводить до неможливості їх оскарження; алгоритмічна дискримінація на основі упереджених даних; відсутність механізмів пояснення рішень, що порушує право на справедливий суд; та автоматизація виявлення правопорушень із використанням неточних або застарілих даних. Ці конфлікти набувають особливої гостроти, коли рішення, що впливають на основні права особи (наприклад, у сфері соціального забезпечення чи міграції), приймаються без участі людини і без можливості їх ефективного оскарження.

На інституційному рівні конфлікти виникають між різними гілками влади та державними органами. Серед них – конфлікт легітимності щодо того, хто є справжнім суб'єктом прийняття рішень: людина чи машина; розбіжності між законодавчою та виконавчою владою щодо контролю над впровадженням ШІ; конфлікт між судовою та адміністративною владою через неможливість пояснити логіку алгоритмічних рішень; а також конфлікт між центральними та місцевими органами влади, коли централізовані ШІ-системи нав'язують рішення, порушуючи принципи децентралізації та місцевого самоврядування. Ці колізії ставлять під загрозу рівновагу влад, принцип підзвітності та верховенство права.

На рівні саме законодавства та правотворчості виникають конфлікти, що стосуються внутрішньої логіки правової системи. Серед них –

протиставлення правової та технічної логіки, коли ШІ «оптимізує» норми, змінюючи їх правове значення; конфлікт між традиційними правовими принципами (наприклад, правами людини) та алгоритмічними рекомендаціями, що можуть їх порушувати; конфлікт між вимогою прозорості та комерційною таємницею приватних розробників ШІ; а також конфлікт між швидкістю генерації законопроєктів ШІ та їхньою юридичною якістю, що призводить до появи правових лазок та внутрішніх суперечностей.

На системному рівні виникають структурні конфлікти, що загрожують самій основі правової держави. Це конфлікт легітимності демократії, коли виникає питання, чи залишається народ джерелом влади, чи цю роль починає виконувати алгоритм; конфлікт між людиною та машиною в ролі законодавця, коли депутати втрачають контроль над процесом формування політики; а також конфлікт між національним суверенітетом та транснаціональними ШІ-платформами, коли держава використовує технології, розроблені іноземними компаніями, що може впливати на зміст національного законодавства.

На міжнародному рівні конфлікти виникають між національним законодавством та міжнародними стандартами, зокрема у разі порушення положень GDPR, Рамкової конвенції Ради Європи з ШІ чи інших міжнародних договорів. Також актуалізується конфлікт між державами через «цифрову агресію» – використання ШІ для дезінформації, втручання у вибори чи кібератак, що вимагає формування нових правових механізмів колективної відповіді.

Важливо підкреслити, що не всі конфлікти, пов’язані з автоматизацією, є специфічними для ШІ. Прості системи, такі як бази даних або електронні форми, не мають здатності до навчання, генерації нового змісту чи автономного прийняття рішень. Тому конфліктологія ШІ повинна фокусуватися лише на тих колізіях, які виникають виключно завдяки унікальним характеристикам інтелектуальних систем. До таких специфічних для ШІ конфліктів у сфері законодавства можна віднести:

1. Конфлікт через відтворення історичної дискримінації – ШІ, навчаючись на історичних даних, може виявляти та закріплювати системну несправедливість, наприклад, щодо етнічних меншин чи осіб з низьким доходом, що призводить до нерівного застосування норм. Це неможливо для статичних систем, але є притаманним для моделей, здатних виявляти приховані патерни.

2. Конфлікт через «чорний ящик» – неможливість пояснити логіку рішення, прийнятого нейромережею, створює ризик порушення права на справедливий суд і підриває принцип підзвітності. Це є наслідком складної внутрішньої архітектури ШІ, яка не підлягає інтерпретації.

3. Конфлікт через генерацію «фейкових» правових норм – генеративні моделі мають тенденцію до галюцинацій, тобто створення неіснуючих посилань, вигаданих судових рішень чи вигаданих конституційних норм, що може ввести в оману парламентських аналітиків і депутатів.

4. Конфлікт через адаптивну зміну інтерпретації законів – ШІ, який використовується для тлумачення норм, може змінювати свою інтерпретацію з часом через навчання на нових даних, що порушує принцип передбачуваності права.

5. Конфлікт через алгоритмічне прогнозування «потенційних порушників» – використання кореляцій для визначення «груп ризику» призводить до превентивної дискримінації, коли особи підлягають посиленому контролю не за фактом порушення, а за статистичними оцінками.

6. Конфлікт через автономне формування законодавчих пріоритетів – ШІ може аналізувати соціальні мережі та пропонувати зміни в законодавстві, грунтуючись на цифровому «шумі», а не на реальних політичних пріоритетах суспільства, що призводить до втрати демократичного контролю.

7. Конфлікт через «переоптимізацію» правових норм – ШІ, намагаючись максимізувати ефективність чи економію, може пропонувати скасування або скорочення гарантій, не розуміючи їхньої правової та моральної цінності, що веде до технологічного утилітаризму.

Таким чином, конфліктологія ШІ як наукова дисципліна повинна не лише фіксувати наявні колізії, а й передбачати потенційні конфлікти на ранніх етапах правотворчості, визначаючи чіткі пари суб'єктів, між якими вони виникають. Це дозволяє перейти від описової аналітики до проактивного правового контролю, що є необхідною умовою збереження демократичних принципів, верховенства права та захисту прав людини в епоху штучного інтелекту.

На основі систематизації конфліктів, специфічних для штучного інтелекту, та їхньої класифікації за суб'єктними парами, у статті розроблено практичний інструментарій для проактивного виявлення потенційно конфліктогенних положень у законопроєктах та чинних правових актах. Цей інструментарій у формі семи структурованих промптів розрахований на використання парламентськими аналітиками, депутатами, профільними комітетами та експертними радами як засіб правового контролю за впровадженням ШІ в законодавчу систему. Кожен промпт побудовано як інструмент конфліктологічного аналізу, спрямований на виявлення не просто технічних недоліків, а саме правових колізій, що виникають внаслідок унікальних характеристик ШІ – таких як самонавчання, генерація нового змісту, автономність, непрозорість логіки прийняття рішень («чорний ящик») та здатність до прогнозування на основі кореляцій.

Промпти розроблені з урахуванням того, що конфлікт у сфері застосування ШІ завжди є результатом взаємодії між конкретними суб'єктами права. Тому кожен промпт чітко визначає пару суб'єктів, між якими може виникнути конфлікт, що дозволяє уникнути абстрактних оцінок і зосередитися на реальних правовідносинах. Цей підхід забезпечує не лише точність аналізу, але й підсилює легітимність парламентського контролю, оскільки кожен ризик може бути інтерпретований як потенційне порушення прав певної групи або інституції.

Нижче наведено сім промптів, кожен з яких відповідає одному з ключових типів конфліктів, виявлених у рамках конфліктології ШІ. Ці промпти призначені для використання з генеративними моделями штучного інтелекту (наприклад, ChatGPT, Gemini тощо) і можуть застосовуватися як на етапі попереднього аналізу законопроєктів, так і для моніторингу реалізації чинних законів. Їхня мета – виявити приховані конфлікти, які можуть залишитися непоміченими при традиційному юридичному аналізі, але мають потенційно серйозні наслідки для демократичного контролю, прав людини та верховенства права.

1. ***Промпт для виявлення упередженості через навчання ШІ на дискримінаційних даних.*** Сторони конфлікту: Громадяни (меншини) і держава:

«Проаналізуй цей законопроєкт/норму з точки зору можливого відтворення історичної дискримінації. Чи містяться в ньому положення, які можуть призвести до нерівного застосування щодо певних соціальних, етнічних, гендерних чи регіональних груп, особливо якщо передбачається використання штучного інтелекту для його реалізації? Наведи конкретні положення та поясни потенційний механізм дискримінації.»

2. ***Промпт для виявлення втрати політичного контролю через автономне формування законів ШІ.*** Сторони конфлікту: Парламент і виконавча влада / ШІ:

«Чи передбачає цей законопроєкт автоматичне або автономне формування норм, політик чи адміністративних рішень за допомогою штучного інтелекту без прямого затвердження парламентом? Чи є ризик передачі функцій законотворчості або стратегічного планування системі ШІ, що обмежує демократичний контроль? Якщо так – вкажи конкретні статті та запропонуй механізм відновлення підзвітності.»

3. ***Промпт для виявлення "чорного ящика" та відсутності обґрунтування рішень.*** Сторони конфлікту: Судова влада і адміністративна влада:

«Чи передбачає реалізацію цього закону використання штучного інтелекту для прийняття індивідуальних рішень (наприклад, про надання субсидій, депортацію, штрафи)? Чи забезпечує закон можливість повного пояснення

такого рішення для оскарження в суді? Якщо ні – вкажи статті, що створюють ризик порушення права на справедливий суд.»

4. ***Промпт для виявлення галюцинацій та фейкових правових посилань.*** Сторони конфлікту: Депутати та генеративний ШІ: «Перевір усі правові посилання, судові прецеденти та конституційні норми, зазначені в цьому документі. Чи є серед них вигадані (hallucinated) посилання, яких не існує в офіційних джерелах права? Особливо зверни увагу на формулювання типу 'як зазначено у рішенні Конституційного Суду від...', 'відповідно до статті X закону Y'. Перевір кожне твердження на достовірність.»

5. ***Промпт для виявлення переоптимізації норм заради «ефективності».***
Сторони конфлікту: Правова спільнота та технократична логіка ШІ: «Чи містяться в цьому законопроєкті положення, які можуть бути результатом 'оптимізації' за критеріями ефективності, швидкості чи економії, але при цьому суперечать фундаментальним правовим цінностям (наприклад, справедливості, рівності, правам людини)? Зокрема – чи передбачається скасування або зменшення гарантій захисту через автоматизацію? Вкажи конкретні норми та оціни їх конфліктогенність.»

6. ***Промпт для виявлення використання закритих ШІ без механізмів контролю.*** Сторони конфлікту: Парламент і Приватні технологічні компанії: «Чи передбачає цей закон використання комерційних, закритих систем штучного інтелекту (наприклад, розроблених приватними компаніями) для реалізації державних функцій? Чи передбачено механізми незалежного аудиту, верифікації алгоритмів чи доступу до логіки рішень? Якщо ні – вкажи статті, що створюють ризик втрати суверенного контролю над правовим процесом.»

7. ***Промпт для виявлення загрози для майбутніх поколінь через закріплення тимчасових патернів.*** Сторони конфлікту: Поточне покоління депутатів і майбутні покоління суспільства: «Чи містить цей законопроєкт положення, засновані на аналізі короткострокових даних (наприклад, соціальні мережі, тимчасові кризи), які можуть бути закріплені як довгострокові правові норми через автоматизацію чи використання ШІ? Чи є ризик, що такі норми стануть перманентними обмеженнями прав у майбутньому? Оціни потенційний вплив на права наступних поколінь.»

Ці промпти становлять основу для систематичного, проактивного парламентського контролю за застосуванням ШІ в законодавстві. Вони дозволяють не лише виявляти потенційні ризики, але й формулювати конкретні пропозиції щодо їх усунення, забезпечуючи баланс між технологічним прогресом і збереженням правових цінностей. У наступному розділі пропонується фреймворк безкодового програмування, який дозволяє трансформувати ці промпти з текстових запитів у формалізовані логічні

конструкції, придатні для масштабування та інтеграції в офіційні процедури правотворчості.

Застосування фреймворку безкодового програмування для формалізації промптів

У попередніх розділах було запропоновано та обґрунтовано концепцію фреймворку безкодового програмування[149] як інструменту для структурування взаємодії з генеративними моделями штучного інтелекту у правовій сфері[150]. Цей підхід передбачає трансформацію природної мови в логічно формалізовані конструкції, аналогічні базовим примітивам мов програмування – «Умова», «Цикл» та «Функція». На відміну від традиційного промпт-інжинірингу, заснованого на евристичних формулюваннях, запропонований фреймворк забезпечує передбачуваність, відтворюваність та системність аналізу, що є необхідною умовою для використання ШІ в офіційних процедурах правотворчості та парламентського контролю.

У контексті конфліктології ШІ, цей фреймворк виконує функцію семантичного інжинірингу – перетворення правових конфліктів на формалізовані логічні системи, придатні для автоматизованого аналізу. Кожен із семи розроблених промптів, спрямованих на виявлення специфічних для ШІ конфліктів, було модифіковано відповідно до принципів цього фреймворку, що дозволило перейти від текстових запитів до структурованих промптових систем з чітко визначеною логічною архітектурою.

Так, наприклад, промпт для виявлення упередженості через навчання ШІ на дискримінаційних даних (конфлікт між громадянами та державою) було перетворено на логічну систему, що містить примітив «Умова» (перевірка наявності в законопроєкті положень про використання ШІ) та функціональну абстракцію (аналіз потенційних механізмів дискримінації, оцінка конфліктогенності, пропозиції щодо правок). Ця структура забезпечує, що відповідь моделі не буде випадковою, а буде відповідати чітко визначеному протоколу аналізу, аналогічному юридичній експертизі.

Аналогічно, промпт для виявлення «чорного ящика» (конфлікт між судовою та адміністративною владою) було доповнено примітивом «Цикл», що передбачає послідовний аналіз кожної норми, пов’язаної з індивідуальними адміністративними рішеннями. Це дозволяє систематично перевірити всі потенційно ризикові положення, що особливо важливо в умовах великих

[149] Dmitry Lande, Leonard Strashnoy. Semantic AI Framework for Prompt Engineering. SSRN Preprint: 5172867, DOI: 10.2139/ssrn.5172867 (May 08, 2025). – 14 pp.

[150] Ланде Д.В., Фурашев В.М. Застосування фреймворку безкодового програмування при вирішенні завдань парламентського контролю. // Інформація і право, 2025. – № 2(53). – С. 88-103. DOI: 10.37750/2616-6798.2025.2(53).334055

законопроєктів зі складною структурою. Композиція цього циклу з умовними конструкціями (чи передбачено пояснення рішення?) створює багаторівневу систему контролю, яка імітує логіку юридичного аналізу з урахуванням принципів верховенства права, прозорості та підзвітності.

Формалізація промптів за допомогою цього фреймворку має суттєве наукове та практичне значення для юридичної науки. По-перше, вона дозволяє інституціоналізувати процес парламентського контролю за ШІ, перетворивши його з інтуїтивного на системний. По-друге, такі структуровані промпти можуть бути інтегровані в офіційні процедури експертизи законопроєктів, ставши частиною регламенту профільних комітетів. По-третє, вони створюють основу для розробки платформи підтримки прийняття рішень, яка забезпечуватиме депутатів та аналітиків об'єктивними, відтворюваними інструментами правового аналізу.

Важливо підкреслити, що застосування фреймворку безкодового програмування не зводиться до технічної оптимізації, а є методологічним кроком у розвитку конфліктології ШІ як наукової дисципліни. Він дозволяє формалізувати правові ризики, зробити їх кількісно та якісно вимірними, а також створити універсальну мову взаємодії між юристами, полісі-мейкерами та розробниками ШІ. Це, у свою чергу, сприяє збереженню демократичних принципів, верховенства права та захисту основних прав і свобод у епоху масового впровадження інтелектуальних технологій.

Таким чином, представлені в роботі структуровані промпти, побудовані за принципами безкодового програмування, становлять не просто інструментарій, а новий тип правової інфраструктури – систему проактивного контролю, засновану на логічній прозорості, повторюваності аналізу та чіткому визначенні суб'єктів конфлікту. Це дозволяє перейти від реактивного реагування на наслідки втручання ШІ до проактивного управління правовими ризиками на найраніших етапах законотворчості.

Формалізація контролю конфліктів ШІ за допомогою структурованих промптів

Для забезпечення системного та проактивного контролю за впровадженням штучного інтелекту в законодавчу сферу було розроблено комплексний методологічний підхід, заснований на конфліктології ШІ та семантичному інжинірингу. Цей підхід передбачає трансформацію абстрактних правових ризиків у конкретні, відтворювані та об'єктивні інструменти аналізу, придатні для використання в умовах реального правотворчого процесу. Основним результатом цієї методології є розробка семи структурованих промптів, кожен з яких спрямований на виявлення специфічного типу конфлікту, пов'язаного з унікальними характеристиками ШІ.

Кожен промпт розроблено за принципами фреймворку безкодового програмування, який передбачає формалізацію природної мови шляхом застосування трьох базових логічних примітивів: «Умова» (If-Else), «Цикл»

(For-Loop) та «Функція» (Abstraction). Такий підхід дозволяє перетворити інтуїтивні запити в семантичні системи, що мають передбачувану логічну архітектуру, відтворюваність і можливість масштабування. Це є критично важливим для юридичної науки, де вимоги до об'єктивності, системності та підзвітності є пріоритетними.

Процес розробки промптів є ітеративним. На першому етапі визначено сім ключових конфліктів, специфічних для ШІ, кожен з яких визначено як взаємодія між чітко визначеними суб'єктами права (наприклад, громадяни і. держава, парламент і. виконавча влада, депутати і генеративний ШІ). Це дозволяє уникнути декларативних оцінок і зосередитися на конкретних правовідносинах, які підпадають під ризик порушення.

На другому етапі було сформульовано початкові версії промптів на основі цих конфліктів. Далі, використовуючи LLM, яка була попередньо інформована про логіку фреймворку, було здійснено трансформацію початкових промптів у структуровані семантичні системи. Цей процес включав:

– Визначення точок розгалуження логіки («Умова»);

– Встановлення циклічних операцій для масового аналізу положень («Цикл»);

– Інкапсуляцію завершальних дій (узагальнення, оцінка, пропозиції) у вигляді функцій («Функція»);

– Встановлення чіткого вихідного формату відповіді для забезпечення стандартизації.

Наприклад, для конфлікту між правою спільнотою та технократичною логікою ШІ (пов'язаного з переоптимізацією норм заради «ефективності») було розроблено промпт, який містить:

– «Цикл» для аналізу всіх положень, що містять терміни, пов'язані з оптимізацією;

– «Умову» для розгалуження аналізу залежно від того, чи призводить норма до зменшення правових гарантій;

– «Функцію» для узагальнення результатів, оцінки конфліктогенності та формування конкретних пропозицій щодо правок.

Ця стандартизація дозволяє інтегрувати промпти в офіційні процедури експертизи законопроєктів, створюючи основу для автоматизованої системи парламентського контролю. Такі промпти можуть використовуватися аналітиками профільних комітетів, депутатами, юридичними радами та державними експертами для проактивного виявлення потенційно небезпечних положень ще до їхнього прийняття.

Важливою особливістю методології є її відкритість до зворотного зв'язку. Отримані результати аналізу можуть бути оцінені експертами, на підставі чого можуть вноситися корективи як у початкові, так і у структуровані промпти. Це забезпечує постійне вдосконалення інструментарію, роблячи його більш чутливим до нових форм правових конфліктів, що виникають із розвитком технологій ШІ.

Таким чином, запропонований підхід не лише забезпечує проактивне виявлення ризиків, але й закладає основи для інституціоналізації конфліктології ШІ як наукової дисципліни та практичного механізму правового контролю. У наступному розділі наведено результати практичного застосування цих структурованих промптів до двох реальних документів, що підтверджує їхню ефективність та практичну цінність.

Центральним елементом запропонованої методології є її ітеративна природа, забезпечена системою багаторівневого зворотного зв'язку, що дозволяє постійно вдосконалювати як концептуальну основу, так і технічну реалізацію інструментарію. Ця система зворотного зв'язку є не просто доповненням до основного процесу, а його необхідною умовою, що гарантує адаптивність, точність та наукову строгість підходу.

Процес розпочинається з практичного та наукового застосування структурованих промптів. Результати їхнього застосування – як у формі аналітичних висновків для профільних комітетів, так і у вигляді наукових досліджень та публікацій – стають вхідними даними для експертної оцінки. Саме на цьому етапі активізується перший ключовий зворотний зв'язок: дані про ефективність, точність та корисність отриманих результатів надходять до групи експертів, яка включає юристів, законотворців, технологічних аналітиків та представників громадськості. Ця оцінка є основою для подальшої модифікації.

Далі, на основі отриманої експертної оцінки, відбувається розгалуження процесу. Якщо критика стосується логічної архітектури промпта – наприклад, виявлено недоліки у застосуванні примітивів «Умова», «Цикл» або «Функція», відсутність необхідних розгалужень чи помилки в композиції – активується зворотний зв'язок, спрямований на корекцію структурованих промптів. Цей етап передбачає формалізоване вдосконалення семантичної системи, що використовується для аналізу, з метою підвищення її надійності та відтворюваності.

Якщо ж експерти виявляють, що сама концептуальна основа промпта є неточною – наприклад, формулювання мети не охоплює всі аспекти конфлікту, визначено неповний перелік суб'єктів, або виникає новий тип конфлікту, що не був передбачений спочатку – активується інший напрямок зворотного зв'язку, спрямований на корекцію початкових промптів. Це стосується конфліктологічної моделі як такої, тобто самого визначення конфлікту між суб'єктами права.

Зміни, внесені на концептуальному рівні, не можуть залишатися ізольованими. Вони обов'язково активують ключовий зв'язок між корекцією початкових і структурованих промптів, оскільки будь-яке вдосконалення концепції має бути відображено в її формалізованому втіленні. Це забезпечує цілісність системи: зміна на верхньому рівні (концепція) викликає відповідну зміну на нижньому (формалізація).

Остаточні вдосконалені версії, як початкових, так і структурованих промптів, повертаються до відповідних етапів основного процесу. Вдосконалені структуровані промпти інтегруються назад у аналітичний процес, замінюючи попередні версії. Аналогічно, оновлені початкові промпти стають основою для подальшої роботи, що замикає ітераційний цикл. Таким чином, весь процес, представлений на Рис. 5, не є лінійним, а функціонує як замкнена система з негативним зворотним зв'язком, спрямованою на постійне зменшення помилок і підвищення точності аналізу.

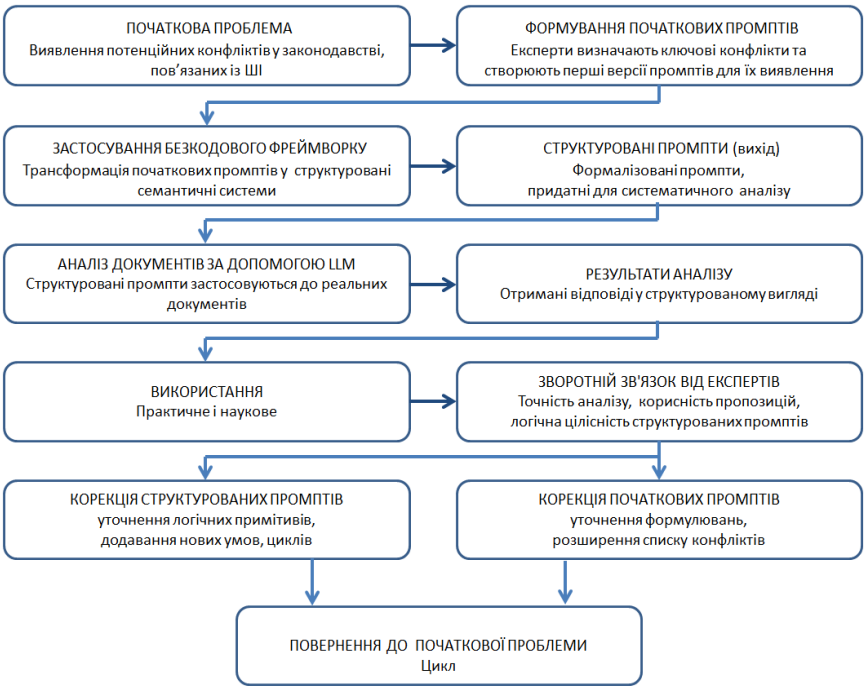

Рисунок 5. Методологія формалізації контролю конфліктів ШІ за допомогою структурованих промптів

Ця ітеративна структура перетворює методологію зі статичного інструментарію в динамічну систему знань, здатну адаптуватися до змін у технологічному ландшафті та правовій практиці. Вона забезпечує, що

конфліктологія ШІ залишається не просто теоретичною дисципліною, а живим, еволюціонуючим механізмом правового контролю.

Формалізація структурованих інструментів на основі фреймворку безкодового програмування

На основі визначених сімей специфічних для штучного інтелекту конфліктів було розроблено сім структурованих промптів, кожен з яких призначений для проактивного виявлення потенційно конфліктогенних положень у законопроєктах та інших правових документах. Ці промпти не є результатом інтуїтивного формулювання, а отримані шляхом цілеспрямованого застосування фреймворку безкодового програмування через інтерактивну взаємодію з генеративною моделлю штучного інтелекту. При цьому модель була попередньо інформована про логіку фреймворку, зокрема про три базові логічні примітиви – «Умова», «Цикл» та «Функція» – і їхню роль у побудові формалізованих систем аналізу. Первинний промпт, спрямований на трансформацію ідей конфліктології ШІ в практичні інструменти, містив чіткі вказівки щодо структури, параметризації, композиції та вимог до вихідного формату. Таким чином, остаточні промпти є результатом семантичного інжинірингу, де природна мова виступає як інструмент створення логічно збалансованих аналітичних систем.

Кожен із семи промптів побудовано як семантична система, відповідна принципам безкодового програмування, що забезпечує передбачуваність, відтворюваність та системність аналізу. Вони включають умовні конструкції для розгалуження логіки, циклічні операції для масового аналізу положень, а також функціональні абстракції для інкапсуляції складних аналітичних завдань. Це дозволяє використовувати їх як стандартизовані інструменти парламентського контролю, придатні для інтеграції в офіційні процедури експертизи законопроєктів.

Для ілюстрації повної реалізації фреймворку нижче наведено детальний приклад структурованого промпту для конфлікту між правою спільнотою та технократичною логікою ШІ (пункт 5), де чітко відображено застосування всіх трьох логічних примітивів у їхній композиції.

Промпт 5: Виявлення переоптимізації норм заради "ефективності"

(Сторони конфлікту – правова спільнота та технократична логіка ШІ)

Проаналізуй наведений нижче текст законопроєкту або норми на наявність положень, де "оптимізація" чи "ефективність" використовуються для зменшення правових гарантій, особливо через використання штучного інтелекту.

Крок 1: Цикл (For-Loop)

Спочатку виконай наступне для кожного положення, що містить слова: "оптимізація", "ефективність", "скорочення", "автоматизація", "підвищення продуктивності":

Виділи повне формулювання.

Оціни, чи це призводить до обмеження прав або гарантій (наприклад, у сфері праці, соціального захисту, судочинства).

Крок 2: Умова (If-Else)

Далі застосуй умову:

Якщо норма зменшує гарантії заради ефективності:

- Вкажи конкретне положення.
- Оціни, як це суперечить фундаментальним правовим цінностям (справедливість, рівність, людська гідність).
- Наведи приклад потенційного конфлікту (наприклад, "робітник позбавлений відпустки через 'оптимізацію' графіку").

Якщо норма не порушує гарантій:

- Підтверди, що баланс між ефективністю та правами дотримано.
- Перевір, чи є механізми контролю за впровадженням.

Крок 3: Функція (Abstraction)

- У будь-якому разі:
- Склади перелік спірних норм.
- Оціни рівень конфліктогенності: низький / середній / високий.
- Запропонуй правки: наприклад, додати принцип "ефективність не може обмежувати конституційні права".

Вихідний формат (структурована відповідь):

1. Перелік норм із аналізом.
2. Оцінка ризику технократичного утилітаризму.
3. Оцінка конфліктогенності.
4. Пропозиції щодо правок.

Ось текст для аналізу: [ВСТАВТЕ ТЕКСТ ЗАКОНОПРОЕКТУ АБО НОРМИ ТУТ]

Цей промпт демонструє повну композицію логічних примітивів:

- «Цикл» забезпечує систематичний перегляд усіх потенційно ризикових положень;
- «Умова» дозволяє розгалуження аналізу залежно від наявності порушення правових гарантій;
- «Функція» інкапсулює завершальну фазу – узагальнення, оцінку та пропозиції, що відповідає вищому рівню абстракції.

Подібна структура застосована до всіх семи промптів, забезпечуючи єдиний підхід до аналізу різних типів конфліктів. Вони можуть бути об’єднані в

єдину систему парламентського контролю за ШІ, яка дозволяє не лише виявляти конфлікти, але й оцінювати їхню конфліктогенність, пропонувати правки та формувати звіти для профільних комітетів. Таким чином, запропоновані промпти становлять не просто набір інструментів, а нову форму правової інфраструктури, засновану на логічній прозорості, формалізації та проактивному управлінні ризиками, пов'язаними із застосуванням штучного інтелекту в законодавстві.

Практичне застосування при аналізі законопроєктів та стратегічних документів

Для підтвердження ефективності запропонованого підходу було проведено практичне застосування розроблених структурованих промптів до двох реальних документів: законопроєкту громадської організації та стратегічного аналітичного матеріалу державного органу. Ці кейси демонструють, як інструменти конфліктології ШІ можуть бути використані для проактивного виявлення потенційно конфліктогенних положень на різних етапах правотворчого процесу – від ініціатив громадськості до підготовки державної політики.

Аналіз законопроєкту «Про використання штучного інтелекту в закладах вищої освіти» (Українська студентська ліга)

Першим об'єктом аналізу став законопроєкт, розроблений Українською студентською лігою[151], який пропонує правову рамку для впровадження ШІ в освітній процес. Для його оцінки було використано структурований промпт №5, спрямований на виявлення ризику технократичного утилітаризму через переоптимізацію норм заради «ефективності».

Аналіз виявив, що, незважаючи на прогресивну спрямованість документа (підвищення доступності освіти, індивідуалізація навчання, етичні принципи), у ньому містяться положення, що можуть призвести до порушення основних прав студентів. Зокрема, норма про можливість використання ШІ для автоматизації системи оцінювання була визнана потенційно конфліктогенною. Відсутність чітких гарантій щодо прозорості алгоритмів, механізмів апеляції та участі людини у фінальному рішенні створює ризик втрати права на справедливу оцінку, особливо для студентів з особливими освітовими потребами.

Оцінка конфліктогенності була визначена як середня, з потенційним конфліктом між студентами, викладачами та адміністрацією закладів вищої освіти. Джерелом конфлікту є протиставлення технологічної логіки (швидкість, ефективність) та правової логіки (справедливість, індивідуальний

[151] Ukrainian Students' League (USL). (2023). Законопроект про використання штучного інтелекту в ЗВО. Отримано з https://www.usl.org.ua/usl-news/noviy-zakonoproekt-pro-vikoristannya-shtuchnogo-intelektu-v-navchalnih-zakladah

підхід). Потенційні наслідки включають масові скарги, судові позови та втрату довіри до освітньої системи.

На підставі аналізу було запропоновано п’ять конкретних правок:

1. Обмеження ролі ШІ в оцінюванні лише допоміжною функцією, з фінальним рішенням викладача.

2. Закріплення права на пояснення логіки рішення ШІ.

3. Введення принципу, що ефективність не може обмежувати конституційні права.

4. Обов’язковий аудит усіх ШІ-систем, що використовуються в освіті.

5. Створення комітетів з етики ШІ на рівні МОН та ЗВО.

Цей кейс підтверджує, що навіть доброзичливі ініціативи, спрямовані на модернізацію державних інституцій, можуть мати приховані конфліктогенні наслідки, які можуть бути виявлені лише за допомогою систематичного аналізу, заснованого на конфліктології ШІ.

Аналіз «Білої книги з регулювання ШІ в Україні: бачення Мінцифри»

Другим об’єктом аналізу стала «Біла книга з регулювання ШІ в Україні», розроблена Міністерством цифрової трансформації України [5]. Цей документ, хоча й не має нормативного статусу, вважається ключовим стратегічним матеріалом для формування майбутньої правової політики. Для його оцінки було використано структурований промпт №3, спрямований на виявлення ризику "чорного ящика" – відсутності механізмів пояснення рішень, прийнятих ШІ.

Аналіз показав, що документ усвідомлює існування ризиків, пов’язаних із автоматизованими рішеннями, зокрема в таких сферах, як працевлаштування, призначення державної допомоги та кредитування. Однак він не пропонує конкретних правових гарантій, які б забезпечили прозорість алгоритмічних рішень. Зокрема, відсутні положення, що передбачають:

- обов’язкове пояснення логіки рішення для особи;
- доступ до даних, використаних системою;
- незалежний аудит алгоритмів;
- обов’язкову участь людини у фінальному рішенні.

Хоча документ згадує такі інструменти, як маркування ШІ-систем та контроль людиною, вони представлені як добровільні або рекомендаційні механізми, а не юридичні зобов’язання. Це створює серйозний ризик порушення статті 6 Європейської конвенції про захист прав людини і основних свобод, яка гарантує право на справедливий суд, зокрема – право на обґрунтування рішення.

Оцінка ризику була визначена як висока, оскільки відсутність обов'язкових гарантій може призвести до системної непрозорості державних рішень, що приймаються з використанням ШІ. На підставі аналізу було запропоновано п'ять ключових правок для майбутнього закону про ШІ:

1. Закріплення права на пояснення рішення ШІ.

2. Встановлення обов'язкового аудиту алгоритмів.

3. Заборона остаточних автоматизованих рішень у справах, що стосуються основних прав.

4. Створення механізму оскарження з доступом до даних.

5. Формування державного реєстру ШІ-систем, що використовуються в державному управлінні.

Цей кейс демонструє, що навіть високорівневі стратегічні документи, розроблені державними органами, можуть містити прогалини в забезпеченні правових гарантій. Виявлення цих прогалин на ранніх етапах є критично важливим для запобігання майбутнім конфліктам між громадянами та державою.

Таким чином, обидва кейси підтверджують високу практичну цінність запропонованого підходу. Структуровані промпти, побудовані за принципами безкодового програмування, дозволяють систематично виявляти приховані конфлікти, пов'язані з унікальними характеристиками ШІ; надавати конкретні, відтворювані та обґрунтовані рекомендації щодо правок; забезпечувати проактивний парламентський контроль за впровадженням технологій.

Особливо важливим є те, що інструментарій ефективно працює як з нормативними актами, так і з аналітичними документами, що робить його універсальним інструментом для всіх етапів правотворчого процесу. Результати аналізу свідчать про те, що конфліктологія ШІ не є абстрактною теорією, а має реальне застосування для забезпечення демократичного, етичного та правового контролю в епоху штучного інтелекту.

Таким чином, розроблено і обґрунтовано комплексний підхід до виявлення та аналізу правових конфліктів, що виникають внаслідок застосування штучного інтелекту в законодавстві, з акцентом на тих конфліктах, які є специфічними саме для інтелектуальних систем. Центральним теоретичним досягненням статті є формалізація нової наукової парадигми – конфліктології штучного інтелекту – як окремого напряму правової науки, спрямованого на системне дослідження соціально-правових колізій, що породжуються унікальними характеристиками ШІ, такими як самонавчання, генерація нового змісту, автономність, непрозорість логіки прийняття рішень («чорний ящик») та здатність до прогнозування на основі кореляцій, а не причинно-наслідкових зв'язків. Ця дисципліна виходить за межі традиційного правового аналізу, оскільки вимагає не просто інтерпретації норм, а

передбачення їхньої динамічної взаємодії з адаптивними технологічними системами, які можуть змінювати своє поводження в часі та впливати на саму природу правовідносин.

Суть виконаної роботи полягає в тому, щоб трансформувати абстрактні ризики в конкретні правові конфлікти, чітко визначені як взаємодія між суб'єктами права. У статті систематизовано сім ключових типів конфліктів, притаманних виключно ШІ, кожен з яких розглядається через призму конкретних суб'єктів: громадянин vs. держава, парламент vs. виконавча влада, суд vs. адміністрація, депутати vs. генеративний ШІ, правова спільнота vs. технократична логіка, парламент vs. приватні технологічні компанії, поточне покоління vs. майбутні суспільства. Такий підхід дозволяє уникнути декларативності та зосередитися на реальних правових відносинах, які підпадають під ризик порушення через втручання ШІ. Особливо важливим є те, що запропонована класифікація чітко відрізняє конфлікти, пов'язані з унікальними властивостями ШІ, від тих, що можуть виникати в будь-яких системах автоматизації, що забезпечує точність правової кваліфікації.

Наукова новизна підходу, що розглядається, полягає в інтеграції конфліктологічного аналізу з практикою правотворчості через розробку семи структурованих промптів, які виступають як інструменти парламентського контролю. Ці промпти не є просто текстовими запитами до генеративних моделей – вони побудовані за принципами логічної формалізації, що включає умови, цикли та функції, і тому можуть розглядатися як семантичні системи, придатні до відтворення, тестування та масштабування. Такий підхід дозволяє перетворити природну мову в інструмент системного правового аналізу, що становить новий етап у взаємодії юриста з технологією. Це, у свою чергу, закладає основи для дисципліни «логіки промптів» – нової галузі правового інжинірингу, де промпти стають аналогами юридичних норм, але призначені для взаємодії з ШІ.

Практична цінність цього підходу підтверджена шляхом застосування розроблених промптів до реальних документів – законопроєкту Української студентської ліги та «Білої книги з регулювання ШІ в Україні». У обох випадках виявлено потенційно конфліктогенні положення, які не були очевидними на перший погляд. Зокрема, у законопроєкті про освіту виявлено ризик технократичного утилітаризму через передачу функцій оцінювання студентів без механізмів людського контролю та пояснення рішень. У «Білій книзі» виявлено високий рівень конфліктогенності через відсутність обов'язкових гарантій прозорості алгоритмічних рішень, що створює реальний ризик порушення права на справедливий суд. Ці приклади свідчать про те, що запропонована методологія ефективна для проактивного виявлення правових ризиків на етапі формування політики та правотворчості.

Перспективи розвитку конфліктології ШІ пов'язані з інституціоналізацією запропонованого підходу в рамках парламентської діяльності. Можливе створення спеціалізованого комітету з етики ШІ, впровадження обов'язкової

експертизи законопроєктів за допомогою структурованих промптів, а також розробка реєстру ШІ-систем, що використовуються в державному управлінні. Крім того, методологія може бути розширена на інші сфери – судочинство, правоохоронну діяльність, регуляторну політику, – де також існує високий ризик виникнення конфліктів, специфічних для ШІ. У майбутньому можлива інтеграція промптів у офіційні інформаційні системи Верховної Ради, що дозволить автоматизувати попередній аналіз законопроєктів на предмет конфліктогенності.

Розділ 8. Шлях до інтелектуальної координації

Сучасний етап розвитку суспільства характеризується прискореним поширенням технологій штучного інтелекту, особливо генеративних мовних моделей, які кардинально змінюють способи обробки, аналізу та генерації інформації. У сфері державного управління та законодавчої влади ці технології відкривають нові можливості для підвищення ефективності, прозорості та системності парламентського контролю. Однак їх потенціал реалізується лише за умови переходу від використання ГШІ як інструменту текстової генерації до інтегрованої системи інтелектуальної координації, здатної забезпечувати структурований, відтворюваний та підзвітний аналіз правових процесів.

Поточна практика застосування ГШІ в політичній та правовій сферах часто обмежується генерацією текстів на запит, що не забезпечує необхідного рівня системності, формалізації та інтеграції з іншими аналітичними системами. Це призводить до того, що вивід моделі залишається ізольованим, неструктурованим та важкоперевіреним, що суперечить вимогам підзвітності та обґрунтованості рішень у парламентській діяльності. Тому важливим завданням є трансформація ГШІ з пассивного асистента в активний компонент аналітичної інфраструктури, який виступає як частина більшої системи контролю.

Цей розділ присвячений викладенню концепції інтелектуальної координації – процесу інтеграції людської експертної компетенції з можливостями генеративного штучного інтелекту через створення формалізованих, структурованих та візуалізованих моделей знань. Така координація передбачає не просто використання ШІ для відповідей на запити, а побудову виконуваних модулів аналізу, які можуть бути інтегровані в системи підтримки рішень, автоматизовані процеси моніторингу та стратегічне планування. У цьому контексті ГШІ стає не просто інструментом, а співвиконавцем аналітичного процесу, здатним до формалізованого мислення, побудови причинно-наслідкових ланцюжків та участі в сценарному моделюванні.

Запропонований підхід ґрунтується на синтезі семантичного інжинірингу, промпт-керованого програмування, формалізації виводу та візуалізації знань, що дозволяє перетворити неструктуровану природну мову на машинно-читабельні моделі, придатні для подальшої обробки, аналізу та інтерпретації. Цей шлях від генерації до структурованого контролю є основою для створення нової парадигми управління, заснованої на прозорості, відтворюваності та інтелектуальній підтримці.

8.1. Від традиційного аналізу до структурованого правового контролю

Традиційне застосування генеративних мовних моделей у правовій сфері зводиться до текстової генерації відповідей на запити користувача. Хоча такі

відповіді можуть бути логічно узгодженими та інформативними, вони залишаються пасивним продуктом, який важко інтегрувати в аналітичні процеси, оскільки відсутня структура, стандартизація та можливість подальшої обробки. Для ефективного парламентського контролю необхідно здійснити перехід від генеративного аналізу до структурованого правового контролю, де кожен вивід системи стає частиною формалізованої, відтворюваної та підзвітної системи.

Від текстової генерації до формалізованого виводу

Важливим кроком є формалізація виводу – перетворення текстової відповіді в структуровану, машинно-читабельну форму. Це досягається шляхом використання структурованих промптів, які вимагають від моделі не абзаців тексту, а чітко визначених пар, списків або таблиць. Наприклад:

«Перелічіть 10 причин відсутності визначень у законопроекті. Представте у форматі: "причина; Відсутність визначень"»

Такий запит змушує модель генерувати дані у нормалізованому вигляді, що дозволяє їх використовувати для подальшого аналізу.

Перехід від асистента до виконуваного модуля

У цьому підході ГШІ перестає бути просто «асистентом», який відповідає на запити, і перетворюється на виконуваний модуль – компонент аналітичної системи, який може бути запущений автоматично, інтегрований в ланцюжок обробки даних або використаний як частина безкодового програмування. Наприклад, агент-аналітик може бути запущений для перевірки законопроекту на відповідність міжнародним зобов'язанням, а його вивід – переданий агенту-контролеру для формування звіту.

Систематизація відповідей у структуровані дані (CSV, JSON)

Для забезпечення інтеграції з іншими системами всі відповіді ГШІ конвертуються у структуровані формати, такі як CSV або JSON. Це дозволяє:

- зберігати дані в базах знань (наприклад, Dataset, SemanticCore);
- імпортувати їх у програми візуалізації (Gephi, GraphViz);
- виконувати кількісний аналіз (наприклад, обчислення центральності, щільності мережі).

Наприклад, відповідь у форматі CSV:

Source,Target,Type

"Лакуни у тексті","Відсутність визначень","Причина"

"Конфлікт інтересів","Протиріччя норм","Причина"

може бути безпосередньо завантажена в Gephi для побудови семантичної мережі.

Інтеграція з аналітичними платформами

Формалізовані дані інтегруються в аналітичні платформи, такі як Gephi, де вони візуалізуються, аналізуються та інтерпретуються. Це дозволяє:

– виявляти найвпливовіші вузли (наприклад, *«відсутність визначень»* як найвпливовіша причина суперечностей);

– виявляти кластери конфліктів;

– формувати сценарії на основі каузальних ланцюжків.

Така інтеграція перетворює ГШІ з ізольованого інструменту в частина єдиної інтелектуальної системи контролю.

Відтворюваність та підзвітність аналітичних процесів

Формалізація та структурування даних забезпечують відтворюваність – можливість повторити аналіз при зміні умов або даних. Вона також забезпечує підзвітність, оскільки кожен крок аналізу може бути зафіксований, перевірений та презентований як частина офіційного звіту. Це критично важливо для демократичних інституцій, де прийняття рішень має бути обґрунтованим, прозорим та підзвітним громадянам.

Таким чином, перехід від генеративного аналізу до структурованого правового контролю є не просто технічною оптимізацією, а фундаментальною трансформацією методології контролю. Він дозволяє перетворити ГШІ з інструменту генерації тексту в інженерний інструмент аналізу, здатний забезпечити науково обґрунтовану, системну та підзвітну підтримку парламентської діяльності.

8.2. Семантичний нетворкінг – логічна формалізація та прозорість

Розвиток технологій генеративного штучного інтелекту вимагає не просто адаптації існуючих методів аналізу, а формування нової наукової та методологічної парадигми для парламентського контролю. Ця парадигма грунтується на синтезі трьох компонентів: семантичного нетворкінгу, логічної формалізації та прозорості. Саме їхня інтеграція дозволяє перетворити ГШІ з інструменту текстової генерації в інтелектуальну систему підтримки рішень, здатну забезпечити системний, відтворюваний та підзвітний аналіз складних правових явищ.

Семантичний нетворкінг як основа аналізу

Семантичний нетворкінг – це процес побудови та аналізу мереж, що відображають змістовні (семантичні) зв’язки між сутностями, виявленими в текстових документах. У контексті парламентського контролю ці сутності можуть бути такими: *«законопроект»*, *«скарга»*, *«податкова перевірка»*, *«конфлікт інтересів»*. Зв’язки між ними (наприклад, *«призводить до»*, *«порушує»*, *«регулює»*) формують структуровану модель знань – семантичну мережу.

Ця мережа є основою для всіх подальших аналітичних операцій. Вона дозволяє:

– виявляти ключові поняття (через метрики центральності);

– виявляти кластери проблем (наприклад, групу скарг, пов'язаних із податковою службою);

– порівнювати різні документи за рівнем новизни або повторюваності.

Як зазначено в літературі, семантичні мережі вже використовуються в правовій інформатиці для аналізу документів, побудови тезаурусів та систем рекомендацій. Застосування ГШІ значно прискорює процес формування таких мереж, оскільки дозволяє автоматично витягувати сутності та зв'язки з великих масивів тексту.

Логічна формалізація причинно-наслідкових ланцюжків

Семантична мережа, побудована за допомогою ГШІ, є лише початком. Для перетворення її в аналітичну модель необхідна логічна формалізація – перехід від графічного представлення до формальної структури, придатної для логічного виводу, сценарного аналізу та прогнозування.

Важливим елементом є побудова причинно-наслідкових (каузальних) ланцюжків, які моделюють шляхи впливу одних явищ на інші. Наприклад:

«Зростання кількості скарг» → «Недостатня прозорість процедур» → «Порушення прав бізнесу» → «Необхідність зміни регуляторної політики»

Ці ланцюжки формуються за допомогою методу двонаправленого пошуку, коли мережа будується одночасно від початкового стану (наприклад, *«Парламентський контроль»*) і від кінцевої мети (наприклад, *«Несуперечність документів уряду»*). Коли часткові мережі «зустрічаються», утворюються повні шляхи досягнення мети.

Формалізація цих ланцюжків у вигляді структурованих даних (CSV, JSON) дозволяє інтегрувати їх у системи підтримки рішень, використовувати для оптимізації шляхів реформ та моделювання наслідків.

Прозорість промптів та алгоритмів

Однією з найсерйозніших проблем застосування ГШІ в державному управлінні є відсутність прозорості. Якщо рішення приймається на основі виводу моделі, який не можна відтворити чи пояснити, це ставить під загрозу підзвітність влади. Тому нова парадигма передбачає повну прозорість процесу аналізу.

Це досягається шляхом:

– Фіксації використаних промптів: кожен запит до ГШІ, який використовується для виявлення сутностей, побудови мережі чи ранжування сценаріїв, має бути задокументований.

– Публікації алгоритмів аналізу: наприклад, метод побудови каузальних ланцюжків має бути описаний як відтворюваний процес.

– Використання відкритих інструментів: відмова від «чорних скриньок» на користь таких програм, як Gephi або GraphViz, які дозволяють візуалізувати та перевіряти результати.

Прозорість забезпечує, що аналіз може бути перевірений, оцінений та відтворений іншими дослідниками, депутатами чи громадськими організаціями.

Фіксація та документування процесу аналізу

Для забезпечення підзвітності необхідно документувати весь аналітичний процес, а не лише його результат, що включає:

- вихідні дані (тексти законопроектів, скарг, звітів);
- використані промпти та параметри запитів;
- отримані відповіді від ГШІ;
- конвертовані дані у форматі CSV;
- побудовані мережі (у вигляді файлів GEXF, GraphML);
- версії використаних інструментів (наприклад, Gephi).

Така повна документація дозволяє:

- відстежити, як було отримано певний висновок;
- виявити можливі помилки або галюцинації;
- провести повторний аналіз при зміні умов.

Це робить систему контролю відкритою, науково обґрунтованою та демократично підзвітною.

Створення відкритих реєстрів знань

Останнім кроком є інституціалізація знань, створених у процесі аналізу. Це може бути зроблено шляхом створення відкритих реєстрів знань – централізованих баз даних, де зберігаються:

- семантичні мережі, побудовані для різних сфер (податки, митниця, реформи ШІ);
- онтології ключових понять;
- реєстри конфліктогенних норм;
- сценарії реалізації реформ.

Ці реєстри можуть бути інтегровані з офіційними порталами парламенту, забезпечуючи доступ не лише депутатам, а й громадським організаціям, ЗМІ

та науковцям. Вони стають спільним ресурсом для стратегічного планування, контролю та громадського діалогу.

Таким чином, нова парадигма парламентського контролю, заснована на семантичному нетворкінгу, логічній формалізації та прозорості, пропонує не просто нові інструменти, а новий спосіб мислення. Вона перетворює контроль з реактивного механізму в проактивну, інтелектуально підсилену систему, здатну забезпечити ефективну, обґрунтовану та підзвітну діяльність законодавчої влади в умовах цифрової трансформації.

8.3. Рекомендації

Швидкий розвиток технологій генеративного штучного інтелекту вимагає від державних інституцій не лише адаптації до нових можливостей, а й системного реагування на пов'язані з ними виклики. Для того щоб використання ГШІ було ефективним, прозорим, підзвітним та безпечним, необхідно розробити чіткі стратегії, інституційні механізми та нормативно-правові засади. У цьому пункті наведено комплекс рекомендацій, адресованих основним суб'єктам державного управління – парламенту та регуляторам, спрямованим на забезпечення відповідального, стратегічного та системного застосування ГШІ.

Розробка внутрішніх стандартів використання ГШІ

Першим кроком до відповідального використання ГШІ є створення внутрішніх стандартів, які регулюватимуть порядок застосування технологій у роботі державних органів. Ці стандарти повинні визначати:

– цілі використання ГШІ (наприклад, аналіз законопроектів, обробка скарг, моніторинг діяльності уряду);

– допустимі платформи та інструменти (наприклад, використання лише версій з підписаним корпоративним контрактом);

– обмеження щодо обробки персональних даних та конфіденційної інформації;

– вимоги до формату виводу (наприклад, обов'язкова структуризація відповідей у CSV).

Такі стандарти забезпечать єдність підходу, зменшать ризики помилок та забезпечать юридичну безпеку.

Створення інституційних підрозділів з аналізу ШІ

Для ефективної інтеграції ГШІ у державне управління необхідно створити спеціалізовані підрозділи або навчити існуючі аналітичні служби працювати з інтелектуальними інструментами. Ці підрозділи мають здійснювати:

– підготовку промптів для аналізу законопроектів, звітів, скарг;

– побудову та візуалізацію семантичних мереж;

– інтерпретацію виводів ГШІ з урахуванням правових, етичних та соціальних аспектів;

– моніторинг якості виводів та виявлення галюцинацій.

Наприклад, у парламенті може бути створено відділ інтелектуального контролю, який використовуватиме ГШІ для автоматизованого аналізу законодавства, формування сценаріїв реформ та виявлення конфліктів.

Впровадження обов'язкової верифікації ШІ-висновків

Одним із найсерйозніших ризиків використання ГШІ є галюцинації – генерація неіснуючих фактів, норм, посилань. Тому будь-який вивід, отриманий за допомогою ГШІ, повинен підлягати обов'язковій верифікації. Це може здійснюватися шляхом:

- перехресної перевірки з офіційними джерелами (наприклад, базою законодавства zakon.rada.gov.ua);
- порівняння з попередніми аналізами;
- використання кількох незалежних моделей для отримання консенсусу;
- експертної оцінки людиною.

Верифікація має бути обов'язковим етапом перед використанням виводу у звітах, рішеннях або законопроектах.

Формування єдиних онтологій для державного управління

Для забезпечення системності та узгодженості аналізу необхідно розробити єдині онтології – структуровані моделі ключових понять у різних сферах державного управління. Наприклад:

– онтологія парламентського контролю (поняття: *«запит»*, *«контроль»*, *«порушення»*, *«скарга»*);

– онтологія податкової системи (*«перевірка»*, *«блокування рахунку»*, *«оскарження»*);

– онтологія штучного інтелекту (*«генеративний ШІ»*, *«автономність»*, *«відповідальність»*).

Ці онтології дозволять:

- стандартизувати термінологію;
- забезпечити узгодженість аналізу між різними органами;
- покращити інтероперабельність систем.

Їх можна розробляти за допомогою ГШІ, використовуючи промпт-кероване зондування для виявлення ключових понять та їхніх взаємозв'язків.

Забезпечення підзвітності та контролю за використанням ШІ

Нарешті, використання ГШІ має бути підзвітним громадянам, парламенту та суспільству в цілому. Для цього необхідно:

– документувати всі етапи аналізу, включаючи використані промпти, виводи моделі та результати верифікації;

– публікувати звіти, у яких чітко вказано, які частини аналізу були виконані з допомогою ГШІ;

– створити механізми аудиту використання ШІ в державних органах;

– забезпечити можливість оскарження рішень, прийнятих на основі аналізу з використанням ГШІ.

Підзвітність – це не лише етична вимога, а й умова довіри до державних інституцій.

Таким чином, реалізація цих рекомендацій дозволить парламентам, урядам та регуляторам не просто використовувати ГШІ, а інтегрувати його в систему відповідального управління. Це забезпечить баланс між інноваціями та підзвітністю, ефективністю та безпекою, роблячи державне управління більш інтелектуальним, прозорим та адаптивним у умовах цифрової трансформації.

8.4. Перспективи

Майбутнє парламентського контролю неможливо уявити без глибокої інтеграції технологій генеративного штучного інтелекту в цифрову інфраструктуру законодавчих органів. Поточний етап, коли аналіз здійснюється через окремі запити до інтерфейсів на кшталт ChatGPT, є лише перехідним. Наступний крок – створення інтегрованих, автономних систем підтримки рішень, які працюють безпосередньо в рамках існуючих цифрових платформ, забезпечуючи постійний, проактивний та персоналізований контроль.

Інтеграція з парламентськими інформаційними системами

Важливим напрямком є інтеграція ГШІ з внутрішніми інформаційними системами парламенту – базами законодавства, системами документообігу, реєстрами депутатських запитів, звітами уряду. Це можливо через API (Application Programming Interface), які дозволяють системам ШІ безпосередньо отримувати дані, аналізувати їх та повертати структуровані висновки. Така інтеграція забезпечує:

– автоматичне оновлення аналітичних моделей при зміні законодавства;

– синхронізацію з офіційними джерелами, що підвищує достовірність;

– безпечний доступ до конфіденційної інформації в межах уповноважень.

Розробка інтелектуальних асистентів для депутатів

На основі інтегрованих систем можуть бути створені персональні інтелектуальні асистенти для кожного депутата. Такі асистенти будуть:

– аналізувати законопроекти з позиції профільного комітету;

– формувати короткі зведення з виявленням конфліктів, прогалин, фінансових наслідків;

– відповідати на запити типу *«Як цей закон вплине на бюджет мого округу?»* або *«Чи суперечить він міжнародним зобов'язанням?»*.

Ці асистенти можуть бути побудовані на принципах віртуальних експертів, кожен з яких виконує певну роль – аналітика, контролера, інтерпретатора норм.

Створення платформ для колаборативного аналізу

Майбутні платформи повинні підтримувати колаборативний аналіз – спільну роботу депутатів, експертів, асистентів над одним аналітичним завданням, наприклад:

– один депутат запускає аналіз законопроекту;

– асистент формує мережу суперечностей;

– інші депутати можуть доповнювати, коментувати, ранжувати ланцюжки;

– результати зберігаються у спільному репозиторії знань.

Така платформа стає цифровим простором парламентського контролю, де аналіз не є ізольованим, а є частиною колективного інтелекту.

Автоматичний моніторинг законодавчих ініціатив

Інтегровані системи здатні до автономного моніторингу:

– виявлення нових законопроектів, указів, постанов;

– аналіз їх на відповідність Конституції, міжнародним зобов'язанням;

– виявлення змін у нормах, які можуть вплинути на певні сфери (наприклад, податкову, освіту, охорону здоров'я).

Це дозволяє асистентам ініціювати аналіз без прямого запиту, роблячи контроль проактивним.

Персоналізовані дашборди контролю

Кожен депутат може мати персоналізований дашборд, який відображає:

– основні показники контролю (кількість скарг у регіоні, виконання бюджету, терміни відповідей на запити);

– оновлення у сферах, що стосуються його комітету;

– алерти про потенційні конфлікти або порушення.

Дашборд формується на основі даних, зібраних асистентом, і візуалізується у зручному для сприйняття вигляді.

8.5. Виклики

Незважаючи на значний потенціал, впровадження ГШІ в парламентський контроль супроводжується серйозними викликами, які повинні бути передбачені та системно вирішені. Їх ігнорування може призвести до підриву довіри, зловживань та етичних скандалів.

Захист від кібератак і маніпуляцій

Системи, інтегровані з парламентськими базами, стають цінними цілями для кібератак, тому необхідно:

– забезпечити високий рівень кібербезпеки (аутентифікація, шифрування, моніторинг);

– запобігати маніпуляціям даними, які можуть спотворювати аналіз;

– забезпечити аудит дій системи для виявлення зловживань.

Забезпечення якості та достовірності вхідних даних

Якість аналізу залежить від якості вхідних даних. Ризики включають:

– помилки у текстах законів (OCR-помилки, застарілі версії);

– відсутність актуальних звітів;

– неповні скарги громадян.

Тому необхідна система верифікації даних перед їх використанням у аналізі.

Етичні стандарти використання ШІ

Використання ШІ має бути регульоване етичними рамками, які визначають:

– межі використання ШІ (наприклад, заборона на автоматичне голосування);

– обов’язок повідомляти про використання ШІ у формуванні рішень;

– механізми контролю за використанням технологій.

Уникнення дискримінації та упередженості

ГШІ можуть відтворювати та посилювати упередження, наявні в навчальних даних, наприклад:

– системи можуть недооцінювати проблеми певних соціальних груп;

– ігнорувати регіональні особливості.

Тому необхідно:

– тестувати моделі на упередженість;

– використовувати різноманітні, репрезентативні набори даних;

– включати експертів з різних галузей у процес налаштування.

Збереження людського контролю над рішеннями

Найважливішим принципом є людино-орієнтованість: ШІ має бути інструментом підтримки, а не заміною людини. Основні рішення – прийняття законів, висловлення недовіри уряду, розслідування – повинні залишатися в руках депутатів. Системи ШІ можуть надавати аналіз, але фінальне рішення – за людиною.

Таким чином, перспективи розвитку інтелектуального парламентського контролю є надзвичайно широкими, але їх реалізація вимагає не лише технологічних, а й інституційних, етичних та правових засад. Лише за умови відповідального, прозорого та людино-орієнтованого підходу можна досягти справжньої інтелектуальної координації між людиною та машиною.

Висновки

Цю монографію присвячено дослідженню можливостей застосування технологій генеративного штучного інтелекту для трансформації парламентського контролю – функції влади, спрямованої на забезпечення підзвітності, прозорості та ефективності державного управління. У праці систематизовано, узагальнено та розвинуто підхід до побудови інтелектуальних систем контролю, заснованих на синтезі семантичного інжинірингу, причинно-наслідкового моделювання, віртуальних експертів та візуалізації знань. Цей підхід, названий «семантичним нетворкінгом», пропонує нову методологію, яка перетворює ГШІ з інструменту генерації тексту в інженерний інструмент аналізу, придатний для проактивного, прогнозувального та системного контролю.

Важливим висновком дослідження є те, що генеративний штучний інтелект має потенціал кардинально змінити природу парламентського контролю. Він дозволяє подолати традиційні обмеження – часові, кадрові, інформаційні – та перейти від реактивного реагування на порушення до проактивного прогнозування, моделювання сценаріїв та формування рекомендацій. Серед основних результатів дослідження можна перелічити:

– розробку методу двонаправленого пошуку для побудови повних причинно-наслідкових ланцюжків шляхом об’єднання мереж, розширених від проблеми та від мети;

– створення системи промпт-керованого зондування для структурованого виявлення сутностей, зв’язків, прогалин та конфліктів у законодавстві;

– формалізацію виводу ГШІ у вигляді структурованих даних (CSV, JSON), що забезпечує відтворюваність, інтеграцію та підзвітність;

– впровадження концепції віртуальних експертів – спеціалізованих агентів, які виконують ролі аналітика, контролера, інтерпретатора норм;

– розробку нової наукової парадигми – конфліктології штучного інтелекту, яка систематизує джерела, рівні та механізми виникнення конфліктів, пов’язаних із застосуванням ШІ;

– демонстрацію практичної ефективності запропонованих методів на прикладах контролю за податковою системою, аналізу скарг громадян та оцінки виконання міжнародних зобов’язань.

В монографії запропоновано інтегровану, формалізовану та масштабовану методологію застосування ГШІ в парламентському контролі, яка ґрунтується на принципах семантичного інжинірингу та логічної формалізації. Вона використовує промпт-кероване програмування як інструмент побудови виконуваних модулів, передбачає візуалізацію знань у вигляді мереж для

інтерпретації та прийняття рішень, формує основу для інтелектуальної координації між людиною та машиною.

Вперше запропонована концепція конфліктології ШІ як самостійної наукової дисципліни, яка виходить за межі технічного аналізу і охоплює правові, етичні та інституційні аспекти взаємодії людини та ШІ. Незважаючи на значний потенціал, запропонований підхід має серйозні обмеження:

– відсутність постійної пам'яті у LLM робить неможливим створення автономних, навчаються агентів;

– обмеження контекстного вікна не дозволяє аналізувати великі документи повністю;

– ризик галюцинацій вимагає обов'язкової верифікації виводів;

– відсутність стандартизації у форматі промптів, ролей агентів та інтерфейсів взаємодії;

– потреба в людському нагляді для інтерпретації результатів та забезпечення їх достовірності.

Ці обмеження підкреслюють, що ГШІ є інструментом підтримки рішень, а не повноцінною заміною людського експерта.

Майбутній розвиток досліджень повинен бути спрямований на:

– створення платформ для оркестрації віртуальних експертів, які зможуть працювати як «рій» із центральним координатором;

– розробку стандартів для опису ролей, повноважень та інтерфейсів агентів;

– інтеграцію з внутрішніми системами парламенту (наприклад, через API) для автономного моніторингу та аналізу;

– розвиток інструментів для прогнозування наслідків реформ на основі каузальних мереж;

– впровадження систем пояснення рішень для підвищення прозорості та підзвітності.

Генеративний штучний інтелект не є фантастикою майбутнього – він вже сьогодні може бути використаний для підвищення ефективності, об'єктивності та стратегічності парламентського контролю. Ця монографія демонструє, що шлях до інтелектуальної координації між людиною та машиною вже відкритий. Завдання полягає не в тому, щоб боятися технологій, а в тому, щоб навчитися керувати ними, використовуючи їх для підтримки демократичних інституцій, забезпечення підзвітності влади та захисту прав громадян. Автори сподіваються, що ця праця стане відправною точкою для подальших досліджень, розробок та впроваджень, які зроблять парламентський контроль більш інтелектуальним, прозорим та ефективним у епоху цифрової трансформації.

Наукове видання

Дмитро Володимирович ЛАНДЕ
Володимир Миколайович ФУРАШЕВ
Юрій Григорович ДАНИК
Леонард Львович СТРАШНОЙ

ІНТЕЛЕКТУАЛЬНА СИСТЕМА ПАРЛАМЕНТСЬКОГО КОНТРОЛЮ

Від семантичного аналізу до інтелектуальної координації в державному управлінні з використанням штучного інтелекту

Монографія

В авторській редакції

Підписано до друку 20.10.2025. Формат 60x84/16. /16. Папір офс. Гарнітура Times. Спосіб друку – ризографія. Ум. друк. арк. 15,5. Наклад 120 пр. Зам. № 15-200. _____
ТОВ "Інжиніринг"
ISBN 978-617-8180-03-4