DOI 10.20535/2411-1031.2025.13.2.344712
УДК 004. 89:004.9

ДМИТРО ЛАНДЕ,
ОЛЕКСАНДР РИБАК

БЕЗКОДОВИЙ ПІДХІД ДО ПОБУДОВИ СЕМАНТИЧНИХ МЕРЕЖ ЗАСОБАМИ ПРОМПТ-ІНЖИНІРИНГУ

У статті запропоновано безкодовий підхід до побудови семантичних мереж засобами промпт-інжинірингу з використанням великих мовних моделей (LLM). Розроблено фреймворк, у якому базові примітиви – умова, цикл і функція – поєднуються у композиційні структури, що забезпечують автоматизоване виділення понять, встановлення між ними зв'язків і побудову формалізованих графів знань. Запропонований метод спирається на принцип no-code, який дозволяє описувати алгоритмічну логіку природною мовою без залучення програмного коду. Це робить можливим використання великих мовних моделей не лише як генераторів тексту, а як повноцінних інструментів для побудови структур знань.

У межах дослідження LLM розглядається як рушій для автоматизованої онтологічної інженерії. Модель інтерпретує природномовні інструкції як формалізовані дії, що дає змогу ітеративно виділяти ключові концепти, визначати типи відношень і формувати графи знань із заданою логічною послідовністю. Особливу увагу приділено галузі кібербезпеки, де швидке створення й актуалізація онтологій загроз має вирішальне значення для своєчасного реагування на нові вектори атак.

Практична реалізація підходу здійснена на прикладі побудови семантичної мережі у тематиці фішингових атак. У ході експерименту мовна модель GPT-5 обробила 48 новинних повідомлень, автоматично сформувавши близько 70 пар пов'язаних понять. Отриманий граф знань відобразив цілісну структуру домену, де центральне поняття "фішинг" поєднано з численними похідними термінами: кібератака, соціальна інженерія, підроблена сторінка, шкідливе ПЗ тощо. Результати експерименту доводять, що запропонована методика забезпечує релевантність міжпоняттєвих зв'язків і збагачення базової термінології семантично спорідненими поняттями.

Інтеграція великих мовних моделей у процес онтологічного моделювання спрощує створення структур знань, знижує поріг входження для користувачів без досвіду програмування та відкриває перспективи розвитку нейросимволічних систем, що поєднують генеративні можливості моделей із формальними методами представлення знань. Запропонований підхід має високий потенціал практичного застосування в галузях, які потребують динамічного оновлення знань, – передусім у кібербезпеці, медицині, фінансових технологіях і аналітиці даних.

Ключові слова: семантичне моделювання знань, великі мовні моделі, кібербезпека, OSINT, текстова аналітика, формальні примітиви управління, промпт-інжиніринг, онтологічне моделювання.

Вступ. Семантичні мережі – фундаментальна форма представлення знань у сучасних системах: вони дозволяють структурувати інформацію та підтримувати експертні системи. Проєктування таких мереж складне й ресурсомістке – воно потребує глибоких галузевих знань та програмних навичок [1]. У кібербезпеці, де постійно з'являються нові види атак і вразливостей, традиційні методи онтологічного моделювання не забезпечують своєчасного оновлення знань. Це створює нагальну потребу в нових підходах, що поєднують швидкість обробки природної мови з формальною структурою знань.

Великі мовні моделі дозволяють автоматизувати аналіз тексту і генерувати семантичні пари “поняття – відношення”, але їхні текстові відповіді слід перетворювати у формальні структури для подальшого аналізу. У цій роботі запропоновано безкодовий фреймворк на основі промпт-примітивів (умова, цикл, функція), який формалізує послідовність дій моделі без програмування. Такий підхід робить побудову семантичної мережі доступною навіть нефахівцям.

На прикладі фішингових атак показано, що цей механізм ефективно формує релевантну онтологію загроз із можливістю швидкого оновлення. Таким чином, розроблений метод є необхідним для автоматизованих систем у різних галузях, де потрібне швидке створення і підтримка актуальних моделей знань.

Аналіз останніх досліджень і публікацій. Активний розвиток промпт-інжинірингу (prompt engineering) та поява потужних великомовних моделей зумовили виникнення концепції програмування поведінки LLM через послідовності підказок (prompts). У роботі [2] було запропоновано формальний підхід до представлення складних завдань для LLM на основі набору базових “*примітивів*” промпта, таких як умова, цикл, функція, мітка та перехід. Ці формальні примітиви дозволяють структурувати процес обробки тексту великою мовною моделлю та роблять його багаторазово відтворюваним. Зокрема, використання конструкцій на кшталт умовних операторів і циклів у текстових інструкціях до моделі забезпечує логічно розгалужені сценарії роботи LLM, а не лінійну послідовність. Такий підхід можна розглядати як різновид “*безкодової*” програмної інженерії, де замість традиційного коду використовуються структуровані фрагменти інструкцій природною мовою. Три базові примітиви – умова, цикл та функція – утворюють основу для безкодового створення складних систем засобами промпт-інжинірингу, дозволяючи застосовувати природну мову для управління логікою AI-систем. У розширеному варіанті фреймворку додаються також примітиви “*мітка*” (для маркування позицій) та “*перехід*” (для зміни порядку виконання інструкцій), що наближає модель до процедурного програмування і дає можливість явно керувати потоком виконання. Таким чином, сформовано теоретичне підґрунтя для *промпт-програмування* – написання алгоритмізованих підказок, які LLM спроможна виконувати послідовно, наче програму.

Паралельно в академічній спільноті з’явилися роботи, що розглядають узагальнення промпт-інжинірингу до рівня повноцінного програмування. Зокрема, Beurer-Kellner та ін. (2023) запропонували концепцію Language Model Programming (LMP) – поєднання текстових промптів зі скриптовою логікою, яка включає контрольні структури та обмеження на відповіді моделі [3]. В рамках LMP було реалізовано мову LMQL – спеціалізовану мову запитів до LLM, що дозволяє вбудовувати конструкції if / for (умови і цикли) та інші обмеження безпосередньо в шаблони підказок. Автори показали, що такий підхід спрощує проєктування інтерактивних діалогів і складних багатокрокових запитів, зберігаючи або навіть підвищуючи точність виконання завдань, а також зменшуючи витрати обчислювальних ресурсів. Отже, сучасні дослідження підтверджують, що підхід “*промпт = програма*” є перспективним напрямом, який забезпечує більш гнучке та контрольоване використання можливостей LLM.

В контексті побудови семантичних структур знань з’являються також роботи, що інтегрують LLM для автоматизації суто онтологічних задач. Lippolis та ін. (2025) досліджують можливості GPT-моделей у генерації чернеток OWL-онтологій з неформальних описів вимог (історій користувача та компетенційних питань). Вони запропонували дві нові техніки промптингу для автоматизованого розвитку онтологій – Memoryless CQbyCQ та Ontogenia – і показали, що LLM можуть генерувати онтології достатньо високої якості, перевершуючи результати новачків-онтологів [1]. При цьому автори наголошують на необхідності багатовимірної оцінки якості таких автоматично згенерованих онтологій та відзначають

типові помилки моделей, які потребують уваги при практичному використанні. Shimizu і Hitzler (2024) окреслили нову область досліджень – *LLM-асистована інженерія знань*, зазначивши, що великі мовні моделі здатні суттєво прискорити онтологічне моделювання, розширення та узгодження онтологій, а також наповнення графів знань (виконуючи, зокрема, усунення неоднозначностей сутностей) [4]. Dehal та ін. (2025) у оглядовій роботі підкреслюють, що LLM і графи знань мають взаємно підсилюючий зв'язок: LLM, завдяки розумінню природної мови і генеративним можливостям, підтримують автоматизацію побудови *Knowledge Graph* (KG), тоді як самі графи знань можуть збагачувати базу знань моделі і підвищувати пояснюваність її відповідей [5]. Практичним підтвердженням цього є результати Mavridis та ін. (2025), які застосували GPT-4 та інші сучасні моделі для зіставлення медичних термінів зі стандартною онтологією *SNOMED CT*. Їхній LLM-агент зміг автоматично зіставити 108 медичних термінів із точністю 93,75% і F1-мірою 96,26%, що наочно демонструє потенціал LLM у подоланні розриву між неструктурованими даними та формальними семантичними моделями [6]. Окремо слід відзначити напрям використання LLM для автоматичного розширення тезаурусів і галузевих словників. Так, Xu та ін. (2025) запропонували багатоетапний підхід для побудови галузевого лексикону в енергетичному домені, де LLM відіграє роль генератора синонімів для спеціалізованих термінів. У їхньому підході спершу класичними методами TF-IDF відбирається базовий словник, а далі розробляються ітеративні шаблони промптів для моделі з механізмом самоперевірки. В результаті було згенеровано словник із 3426 термінів та 10745 синонімів, причому середня косинусна схожість пар синонімів досягла 0,86, а експертна оцінка підтвердила ~89% точності [7]. Це свідчить, що LLM можуть успішно виконувати автоматизовану екстракцію понять та пошук семантично близьких термінів у спеціалізованих галузях знань – особливо якщо інтегрувати їх у структурований безкодовий робочий процес із проміжними перевірками (для зменшення ризику "галюцинацій" моделі).

Існуючі дослідження закладають основу для нашої роботи. Формалізація примітивів промпт-програмування створює своєрідні "блоки Lego" для побудови складних запитів до LLM. Паралельно, успіхи LLM у генеруванні онтологій та графів знань підтверджують доцільність їх застосування для автоматизації побудови семантичних мереж. Однак новизна нашого підходу полягає в тому, що ми не лише визначаємо примітиви, а й композиційно поєднуємо їх для побудови семантичних структур (таких як ієрархічна категоризація понять, виділення синонімічних рядів, автоматична побудова онтології тощо) у середовищі великої мовної моделі. Іншими словами, ми пропонуємо методику, що дозволяє перетворити здатність LLM до генерації тексту у здатність до генерації структурованого графа знань, використовуючи промпти як алгоритмічні інструкції.

Методологія безкодової побудови семантичної мережі. У запропонованому фреймворку [2] складений промпт розглядається як послідовність базових примітивів промпт-програмування, описаних вище. Кожен примітив задає конкретну логічну операцію над вхідними даними і реалізує окремий крок алгоритму. Зокрема, використовуються такі примітиви:

Примітив "*Умова*" (If-Else): забезпечує розгалуження сценарію залежно від булевого предикату. Формально: нехай X –вхідний об'єкт, $P(X)$ – булевий предикат (True/False), A_1, A_2 – дві альтернативні дії. Тоді логіка примітива визначається як функція розгалуження:

$$f(X) = \begin{cases} A_1(X), \textit{якщо } C(X) = True \\ A_2(X), \textit{інакше} \end{cases}. \quad (1)$$

Ця конструкція відповідає умовному оператору if...else у класичних мовах програмування. У контексті LLM "промпт-програма" містить відповідну інструкцію

природною мовою. Наприклад: *"Якщо для вхідного тексту виконуються певні умови, то виконай підзавдання 1; інакше – підзавдання 2."* – тим самим визначається два альтернативні сценарії обробки залежно від змісту тексту.

Примітив "*Цикл*" (For-loop): забезпечує багаторазове повторення підпромпту для множини вхідних об'єктів. Нехай $S = \{x_i\}$ – множина вхідних елементів (наприклад, список текстових документів), а $g(x)$ — функція-підпромпт, яку застосовують до кожного елемента. Тоді результат виконання циклу можна подати як об'єднання результатів для всіх $x \in S$

$$Result = \bigcup_{x \in S} g(x). \tag{2}$$

Тобто примітив "*Цикл*" об'єднує виходи функції g для всіх елементів із набору S. Як приклад, для аналізу набору текстів можна сформулювати промпт: *"Для кожного з наведених текстів знайди до N пар пов'язаних з темою понять."* – модель послідовно обробить кожен текстовий блок і поверне сукупність знайдених пар для всіх документів.

Примітив "*Функція*": дозволяє оформити підпромпт як параметризований блок, який можна багаторазово використовувати з різними параметрами. *Формально*, введемо оператор F, що приймає на вхід об'єкт x і набір параметрів θ, та повертає результат застосування шаблону промпта з підставленими параметрами:

$$F(x, \theta) = P(x, \theta), \tag{3}$$

де $P(x, \theta)$ – визначений шаблон промпта (текст інструкції) із вставленими значеннями параметрів θ.

Це дозволяє, наприклад, створити повторно використовувану промпт-функцію, яка знаходить пари пов'язаних понять за заданою темою і з обмеженням на максимальну кількість результатів.

Таким чином, складений промпт у цілому можна розглядати як композицію зазначених примітивів, подібно до абстрактного синтаксичного дерева програми. Вихід одного примітиву стає вхідними даними для наступного, формуючи своєрідну "програму" з природномовних інструкцій, яку LLM виконує крок за кроком.

Для обробки складних аналітичних, прогностичних або рекурсивних завдань фреймворк можливо розширити новими примітивами:

– Мітка – для позначення точок входу / виходу;
– Перехід – для керування порядком виконання.

Мітка виступає як логічний маркер або контрольна точка, яка дозволяє позначати місця входу, виходу або проміжні етапи виконання "*промпт-програми*". Це створює можливість реалізації складних сценаріїв із кількома гілками чи підпрограмами – аналогічно до ідентифікаторів у графах виконання чи підпрограм у традиційних мовах.

Перехід є оператором, який визначає послідовність руху між цими мітками, тобто керує потоком виконання. Він може бути умовним (залежним від результатів проміжних кроків) або безумовним (викликає інший блок незалежно від попереднього стану).

Порівняння підходів до аналізу новин. У ході дослідження було розглянуто два можливі варіанти обробки корпусу новин для побудови семантичної мережі на тему фішингу:

1. *Аналіз всього корпусу новин разом*. Спроба опрацювати відразу весь корпус (20–100 новин) одним викликом моделі виявила низку суттєвих обмежень:

– Великий обсяг тексту може перевищити контекстну довжину LLM або знизити якість відповіді через надмірну кількість інформації в одному запиті.
– Відбувається змішування контекстів різних статей: модель починає пов'язувати між собою поняття з різних новин, навіть за відсутності прямого зв'язку між ними. Це ускладнює

виокремлення чітких і контекстуально релевантних пар понять, що безпосередньо стосуються теми фішингу.

– Жорстке обмеження на вихід (наприклад, 70 пар понять на весь корпус) не гарантує повного охоплення теми. Модель може зосередитися лише на найбільш помітних новинах, тоді як менш очевидні, але важливі зв'язки залишаться поза увагою. Внаслідок цього різноманітні випадки фішингових атак, представлені у новинах, можуть залишитися непропрацьованими.

2. *Аналіз кожної новини окремо*. Другий підхід передбачає ітеративну обробку кожної новини як окремого текстового блоку: для кожного документа запускається промпт-функція, яка витягує пов'язані з фішингом поняття (із лімітом 3-4 пари на новину, щоб сконцентруватися на найбільш релевантних зв'язках).

– Цей підхід було застосовано у практичному експерименті: при послідовній обробці 48 документів модель сформувала близько 70 пар концептів, забезпечивши внесок кожної новини до побудови семантичної мережі.

– Менш насичені інформацією тексти дали 1-2 пари або жодної (якщо у новині взагалі не згадувалося про фішинг), тоді як більш змістовні – до 4 пар.

– У результаті вдалося отримати ширший спектр пов'язаних із фішингом понять, не втрачаючи контекст окремих повідомлень.

– Додатковою перевагою такого підходу є масштабованість: зі збільшенням кількості новин (наприклад, до 50 чи 100) модель обробляє їх послідовно, що запобігає перевантаженню контекстного вікна й зберігає якість результатів.

Отже, порівняльний аналіз показав, що ітеративний підхід (опрацювання кожної новини окремо) є більш доцільним. Він узгоджується з принципами промпт-програмування: спочатку здійснюється цикл по документах, далі – цикл по знайдених у них поняттях, і на завершальному етапі результати об'єднуються в єдиний граф. Таким чином формується великий граф понять, побудований із локальних зв'язків високої якості.

Автоматизована генерація промптів на основі інструкцій користувача. Для підвищення зручності роботи з фреймворком було розроблено механізм автоматизованої побудови промптів на основі простої природномовної інструкції користувача.

У цьому підході використовується зовнішній файл "Meth.docx", який містить формалізований опис примітивів безкодового промпт-програмування – "*Умова*", "*Цикл*", "*Функція*", а також допоміжних конструкцій "*Мітка*" і "*Перехід*".

Такий файл фактично виступає словником або бібліотекою базових блоків, якими LLM може оперувати під час автоматичного складання складного промпта.

Користувач завантажує цей файл у контекст моделі, після чого формулює завдання звичайною природною мовою – наприклад:

"Проаналізуй кожну новину про фішинг, знайди ключові поняття, пов'язані з темою, і побудуй семантичну мережу понять."

Модель, маючи доступ до описів примітивів у файлі "*Meth.docx*", самостійно будує промпт-програму, поєднуючи базові операції (умову, цикл, функцію) у логічну послідовність.

Таким чином, навіть користувач без навичок програмування може створювати складні аналітичні сценарії, задаючи лише ціль і загальні вимоги.

LLM автоматично формує структурований промпт, у якому кожен крок відповідає певному примітиву, а взаємозв'язки між ними утворюють логіку обробки даних — від перевірки релевантності тексту до побудови графа понять.

Наприклад, у відповідь на вищенаведену інструкцію модель генерує промпт-програму, що складається з таких етапів:

– Ітеративна обробка кожного тексту (примітив "*Цикл*").

– Перевірка наявності у тексті цільових термінів (“*Умова*”).
– Витяг пар пов’язаних понять і побудова локальних зв’язків (“*Функція*”).
– Злиття локальних результатів у загальну структуру знань.

У результаті формується повноцінна промпт-програма, яку LLM може виконати крок за кроком і яка забезпечує логічну, відтворювану обробку даних без написання коду.

Водночас отриманий промпт є гнучким артефактом, який користувач може редагувати, уточнювати або доповнювати — наприклад, змінювати параметри циклів, умови фільтрації чи формати вихідних даних. Такий ітеративний процес корекції дозволяє адаптувати промпт до конкретних аналітичних завдань, підвищуючи точність і стабільність результатів, що є ключовою перевагою безкодового підходу у поєднанні з LLM.

Математична модель семантичної мережі, генерованої LLM. Нехай задано корпус текстових документів $D = \{d_1, d_2, \ldots, d_N\}$, де кожен документ d_i є послідовністю токенів. Мета – побудувати семантичну мережу у вигляді неорієнтованого графа

$$G = (V, E),$$

де $V \subset T$ – множина вершин (концептів), отриманих із D;

$E \subseteq V \times V$ – множина ребер, що відображають семантичні зв’язки між концептами.

Формалізація промпт-примітивів як операторів

Введемо три базові оператори, що відповідають примітивам промпт-інжинірингу:

1. Оператор циклу L:

Для функції $f : D \to 2^{V \times V}$, яка видобуває пари концептів із одного документа,

$$L(f, D) = \bigcup_{d \in D} f(d). \tag{4}$$

2. Оператор умови C:

Нехай $\chi : D \to \{0,1\}$ – індикаторна функція релевантності (наприклад, $\chi(d) = 1$, якщо d містить ключове слово “фішинг”). Тоді

$$C(f, \chi, d) = \begin{cases} w, \text{ якщо } \chi(d) = 1, \\ \varnothing, \text{ інакше.} \end{cases} \tag{5}$$

3. Оператор функції F_θ:

Параметризоване відображення, що залежить від шаблону промпта та параметрів θ:

$$F_\theta : \text{Text} \to 2^{V \times V}.$$

Внутрішньо це стохастичне відображення, але для аналізу розглядаємо його очікуване значення:

$$E[F_\theta(d)] = \{(c_i, c_j) \in V^2 \mid \Pr_{LLM}((c_i, c_j) \mid d, \theta) > \tau\}, \tag{6}$$

де τ – поріг достовірності;

$\Pr_{LLM}((c_i, c_j) \mid d, \theta)$ – це умовна ймовірність, з якою модель, отримавши на вхід текст d і промпт, параметризовані θ, згенерує або підтвердить пару (c_i, c_j) як семантично пов’язану.

Двоетапна конструкція графа

Процес побудови G складається з двох етапів:

Етап 1 (локальна екстракція):

$$E^{(1)} = L(d \to C(F_{01}, \chi, d), D). \tag{7}$$

Етап 2 (глобальне розширення):

Нехай $E^{(1)} = L\left(d \to C\left(F_{\theta 1}, \chi, d\right), D\right)$. Для кожного $v \in V^{(1)} \setminus \{c\}$, де c – центральний концепт (наприклад, “фішинг”), застосовується оператор розширення:

$$E^{(2)} = v \in V^{(1)} \setminus \{c\} \bigcup F_{02}(v), \tag{8}$$

де $F_{02}(v)$ повертає одну пару (v, v'), таку що v' – семантично близький термін до v.

Підсумковий граф:

$$V = V^{(1)} \bigcup \{v` | (v, v`) \in E^{(2)}\}, \quad E = E^{(1)} \bigcup E^{(2)}. \tag{9}$$

Ймовірнісна інтерпретація та оцінка якості

Кожне ребро $e = (u, v) \in E$ характеризується вагою:

$$w(e) = \Pr_{LLM}(e \mid \text{контекст}), \tag{10}$$

що може бути оцінено через логіти або методи самоперевірки (self-consistency).

Для оцінки структурної цілісності мережі вводиться метрика покриття тематичного ядра:

$$\kappa(G, c) = \frac{\left|\left\{v \in V \mid (c, v) \in E\right\}\right|}{\left|V_{ref}\right|}, \tag{11}$$

де V_{ref} – еталонний набір понять з експертної онтології.

Співвідношення з теорією графів: формальні метрики та властивості

Отримана мережа $G = (V, E)$ аналізується за допомогою класичних метрик теорії графів. Нехай $n = |V|$, $m = |E|$.

Ступінь вершини (degree centrality):

$$\deg(v) = \left|\left\{u \in V \mid (v, u) \in E\right\}\right|.$$

Високий ступінь у вершини $c =$ "*фішинг*" свідчить про її роль як хаба.

Посередництво (betweenness centrality):

$$C_B(v) = \sum_{s \neq v \neq t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}, \tag{12}$$

де σ_{st} – кількість найкоротших шляхів між s і t, а $\sigma_{st}(v)$ – кількість таких шляхів, що проходять через v. Ця метрика виявляє "мости" між піддоменами (наприклад, "соціальна інженерія" як зв'язок між "фішингом" і "психологічними маніпуляціями").

Коефіцієнт кластеризації (local clustering coefficient):

$$C(v) = \frac{2 \cdot \left|\left\{(u, w) \in E \mid u, w \in N(v), u \neq v\right\}\right|}{\deg(v) \cdot (\deg(v) - 1)}, \tag{13}$$

де $N(v)$ – множина сусідів вершини v. Високе значення $C(v)$ вказує на наявність щільної локальної тематичної групи (наприклад, "фішингова розсилка", "спам", "електронний лист").

Щільність графа:

$$D(G) = \frac{2m}{n \cdot (n-1)}. \tag{14}$$

Для розріджених семантичних мереж $D(G) << 1$, що характерно для доменів із чіткою ієрархією понять.

Діаметр графа:

$$diam(G) = \max_{u, v \in V} dist(u, v), \tag{15}$$

де $dist(v, u)$ – довжина найкоротшого шляху між u і v. Невеликий діаметр свідчить про високу зв'язність мережі, що є бажаним для онтологій.

Центральність за близькістю (closeness centrality):

$$C_C(v) = \frac{n-1}{\sum_{u \in V \setminus \{v\}} dist(v,u)}. \quad (16)$$

Високе значення $C_C(v)$ означає, що v швидко «досягає» інших понять, що важливо для центральних термінів.

Ці метрики дозволяють кількісно оцінювати якість побудованої семантичної мережі, порівнювати її з еталонними онтологіями та виявляти структурні аномалії (наприклад, ізольовані вершини, що можуть бути наслідком “галюцинацій” LLM).

Алгоритм побудови семантичної мережі (псевдокод). Для реалізації обраного підходу була розроблена оптимізована структура промптів, що поєднує примітиви “*Цикл*”, “*Умова*” та “*Функція*”. Нижче наведено узагальнений псевдокод, який відображає послідовність дій моделі (GPT-5 або аналогічної LLM) при генеруванні семантичної мережі на основі колекції новин про фішинг:

```
Цикл(
  S = {текстові блоки з файлу (окремі новини)},
  F1: Функція(
    x = текстовий блок,
    параметри = {тема = "фішинг", max_пари = 4},
    Інструкція:
      1. Якщо текстовий блок містить тему "фішинг" або споріднені
слова — оброби цей блок.
      2. Виділи до max_пари пов’язаних з темою "фішинг" понять
(кожне поняття — не більше ніж 3 слова), які безпосередньо стосуються
теми.
      3. Для кожного знайденого поняття виконай перевірку:
        Умова(
          Input = поняття,
          C = чи належить поняття до стоп-списку слів (наприклад:
"і", "та", "або", "але", "для", "щоб", "якщо", "при", "над", "під", "у",
"в", "з", тощо)?,
          A1 = skip,  // пропустити поняття, якщо це стоп-слово
          A2 = include // включити поняття, якщо воно не в стоп-
списку
        )
      4. Поверни результат як список пар у форматі
"поняття1;поняття2".
      5. Для кожного знайденого поняття додай окрему пару:
"фішинг;поняття" (зв’язок знайденого поняття з центральним поняттям
"фішинг").
  )
)

Цикл(
  S = {усі унікальні поняття, знайдені на Кроці 1},
  F2: Функція(
    x = поняття,
    параметри = {тип = "смислова близькість", формат =
"поняття;близьке_поняття"},
    Інструкція:
      1. Знайди для даного поняття одне найближче за змістом
(семантично близьке) поняття.
```

```
          2. Поверни результат як пару у заданому форматі:
"початкове_поняття;близьке_поняття".
      )
    )

    Функція(
      x = {список всіх пар з Кроку 1 та Кроку 2},
      параметри = {формат = GEXF},
      Інструкція:
        "Сформуй семантичну мережу на основі наданого списку пар понять
(вершин та зв'язків між ними)
         та експортуй її у форматі .gexf для подальшого аналізу."
    )
```

Основні кроки запропонованого методу:

1. *Ітерація по документах*. Застосовується примітив "*Цикл*" до набору текстових документів на вибрану тему (наприклад, новини про фішингові атаки). Для кожного текстового блоку викликається підпромпт-функція F_1. Спершу модель виконує перевірку за допомогою примітиву "*Умова*", щоб обробляти лише релевантні тексти (тобто "*якщо текст містить слово 'фішинг' або споріднені терміни, то ...; інакше – пропусти цей текст*"). Далі, для відібраного тексту LLM виділяє ключові зв'язки між поняттями: інструкція для цього кроку може бути сформульована як "*Виділи до 4 пар пов'язаних з темою 'фішинг' понять з даного тексту. Поверни пари у форматі 'поняття1; поняття2'. Крім того, для кожного знайденого поняття додай пару 'фішинг; поняття'*." У результаті цього етапу модель аналізує кожен текстовий блок і повертає множину пар пов'язаних концептів для кожного документа. Ці пари включають як знайдені в тексті дві взаємопов'язані між собою сутності, так і додаткові зв'язки кожної з них із центральним поняттям "фішинг".

2. *Розширення списку понять*. З усіх пар, отриманих на кроці 1, формується множина унікальних понять (окрім центрального терміна "фішинг"). Для кожного такого поняття застосовується примітив "*Цикл*" з викликом функції F_2. Інструкція для цього етапу: "*Знайди для даного поняття один найближчий за змістом споріднений термін. Поверни результат як пару у форматі "поняття; близьке поняття"*." Таким чином, модель пропонує для кожного введеного поняття синонім або узагальнююче поняття (тобто термін вищого рівня абстракції), що є семантично близьким до нього.

3. *Формування графу знань*. Усі зібрані на кроках 1 і 2 пари об'єднуються у фінальний граф. Застосовується примітив "*Функція*" для формування підсумкового запиту: модель отримує на вхід згрупований список всіх знайдених пар і виконує інструкцію типу "*Сформуй семантичну мережу на основі наведеного списку пар понять та експортуй її у форматі .gexf*." Формат GEXF (Graph Exchange XML Format) обрано з міркувань зручності подальшого аналізу: отриманий граф можна легко завантажити у зовнішні інструменти візуалізації та аналізу мереж (зокрема, Gephi, Cytoscape, Pajek чи бібліотеку NetworkX у Python). На виході LLM повертає текстове представлення графу – файл у форматі GEXF з переліком вершин та ребер мережі.

На рисунку 1 схематично показана архітектура побудови семантичної мережі за допомогою описаних етапів (послідовності промпт-примітивів).

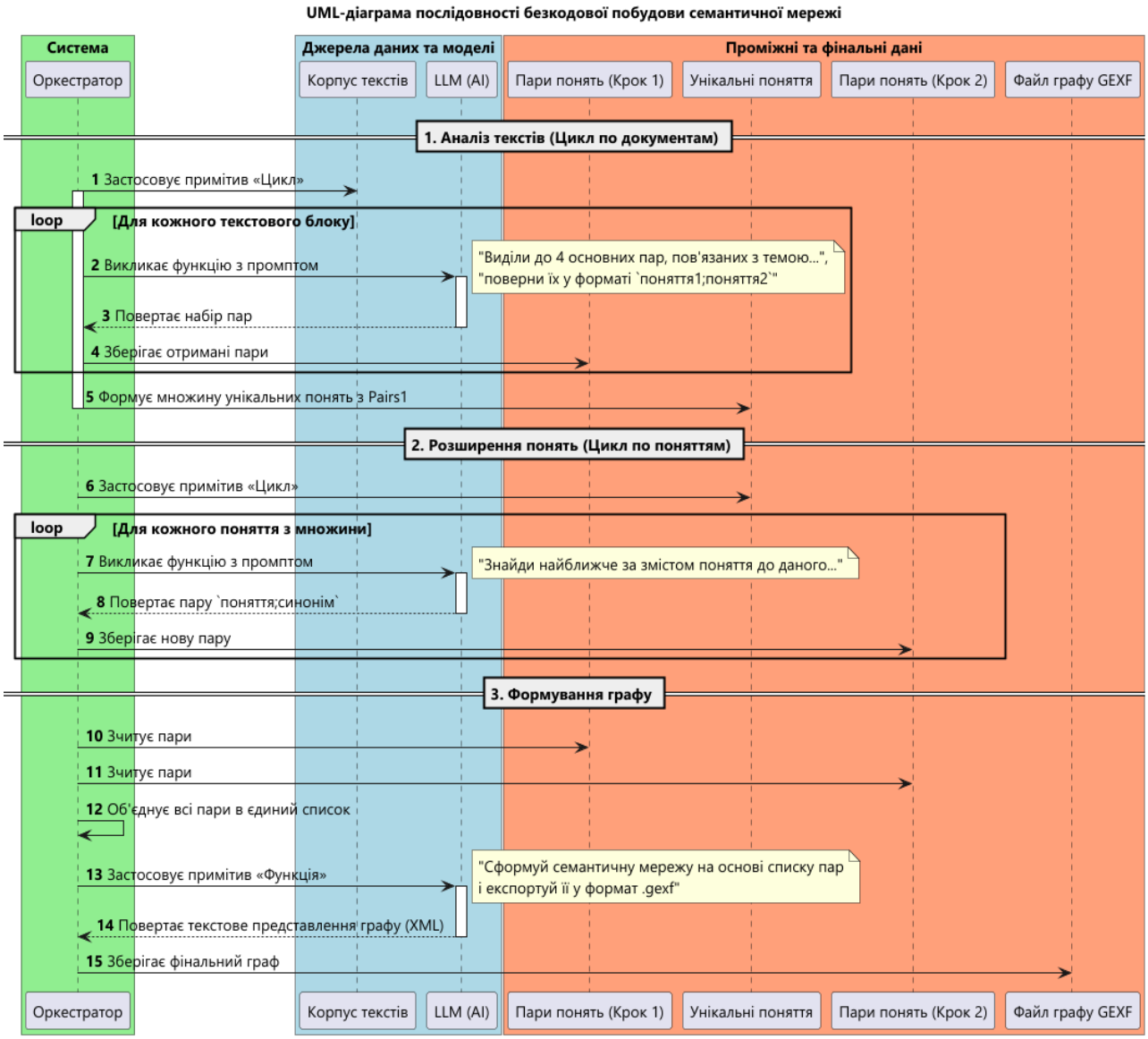

Рисунок 1 – UML-діаграма послідовності побудови семантичної мережі із використанням промт-примітивів

Формально, отримана семантична мережа описується неорієнтованим графом. Нехай S_1 і S_2 – множини унікальних понять, знайдених на Кроках 1 і 2 відповідно, а c – центральне поняття (у нашому прикладі $c =$ “*фішинг*”). Тоді множина вершин цього графа визначається як:

$$V = S_1 \cup S_2 \cup \{c\}. \tag{17}$$

Множина ребер складається з усього набору знайдених зв’язків: E_1, отриманих на Кроці 1, та E_2, отриманих на Кроці 2. Іншими словами:

$$E = E_1 \cup E_2 \tag{18}$$

Отже, семантичну мережу формалізовано як неорієнтований граф

$$G = (V, E), \tag{19}$$

де: V – сукупність усіх залучених понять;

E – сукупність встановлених між ними зв’язків.

Приклад застосування та результати

Для демонстрації підходу було обрано тематику фішингових атак – актуальну кібербезпекову загрозу з власною розгалуженою термінологією (терміни: *фішинг, спарфішинг, фішингові сайти, соціальна інженерія* тощо). За допомогою системи контент-моніторингу соціальних медіа *CyberAgreghator* було зібрано колекцію текстових описів реальних фішингових кейсів, новин і статей українською та російською мовами (рис. 2). Загалом опрацьовано 48 документів, зібраних за останню добу. До цих текстів застосовано описаний вище промпт (з трьома кроками); в якості LLM використано мовну модель GPT-5.

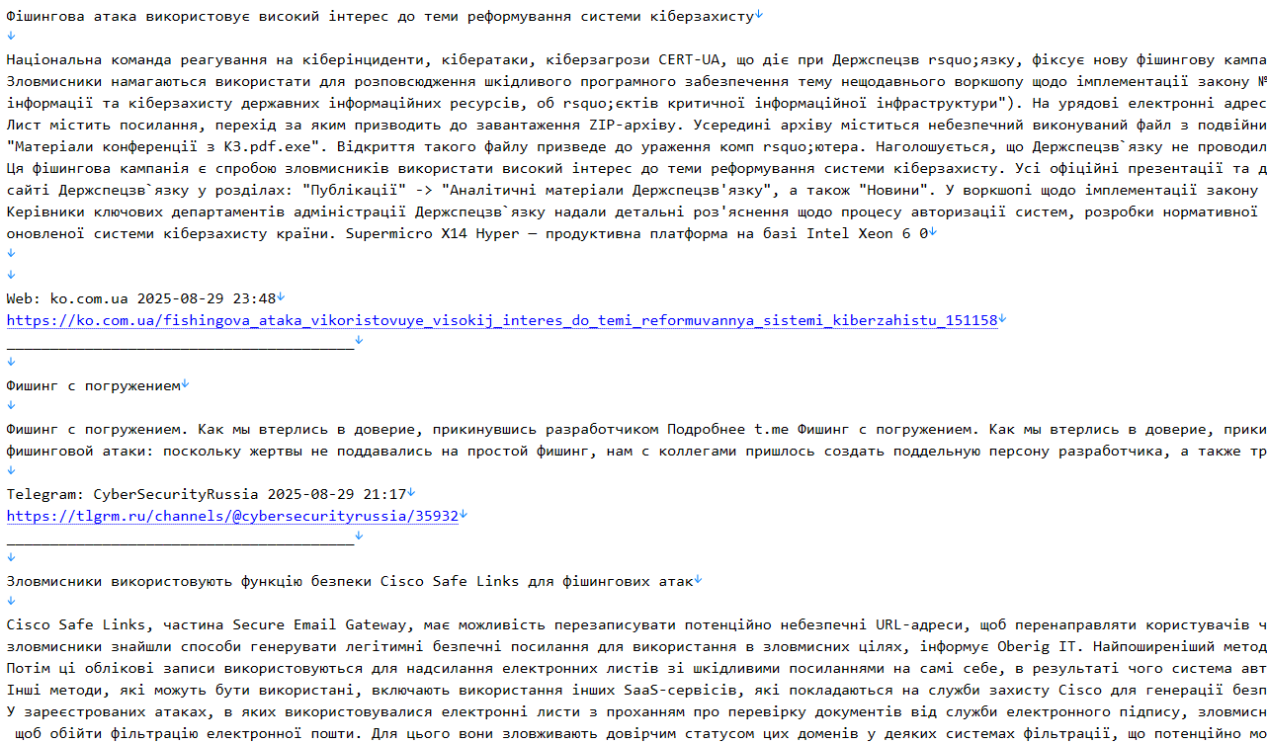

Рисунок 2 – Зібрані системою CyberAgreghator текстові описи фішингових кейсів

На Кроці 1 (цикл по текстах) модель проаналізувала кожен текстовий блок та витягла з нього до чотирьох пар пов’язаних понять. У результаті було отримано близько 70 пар концептів (зв’язаних термінів). Серед прикладів можна навести такі виявлені парні зв’язки:

– “фішинговий сайт; підроблена сторінка” – фішинговий вебсайт імітує сторінку справжнього ресурсу.

– “фішингова розсилка; спам-кампанія” – розсилка фішингових листів є різновидом спам-кампанії.

– “підозрілий файл; зловмисний скрипт” – підозрілий файл у фішингу часто містить шкідливий скрипт.

– “підозрілий URL; небезпечне посилання” – URL-адреса фішингового сайту розглядається як небезпечне посилання.

– “фішингова атака; кібератака” – фішинг є різновидом кібератаки.

Додатково, для збагачення результату, був використаний невеликий довідниковий список термінів, пов’язаних із фішингом. Модель виявила, зокрема, що поняття “електронний лист” часто асоціюється зі “спамом”. Серед сгенерованих нею пар із цього додаткового словника були такі:

- “посилання, фальшивий сайт”;
- “вкладення, вірус”;
- “вірус, шкідливе ПЗ”;
- “шахрайство, соціальна інженерія”;
- “код підтвердження, OTP”;
- “двохфакторна автентифікація, мобільний токен”.

На Кроці 2 (цикл по знайдених поняттях) модель для кожного знайденого на першому етапі поняття запропонувала синонім або узагальнений термін згідно з логікою функції F_2. Наприклад, для поняття “підроблена сторінка” було повернуто синонімічний термін “фейковий вебсайт”, для “спам-кампанія” – більш загальне “масова розсилка”, для “шкідливе ПЗ” – синонім “Malware” тощо. Таким чином відбулося автоматичне розширення початкового списку концептів за рахунок пов’язаних термінів.

На Кроці 3 всі зібрані пари зв’язків з Кроків 1 і 2 були об’єднані у фінальний граф (рис. 3). Модель повернула граф у вигляді фрагмента XML-файлу формату GEXF, який було імпортовано в програму Gephi для візуалізації та перевірки структури.

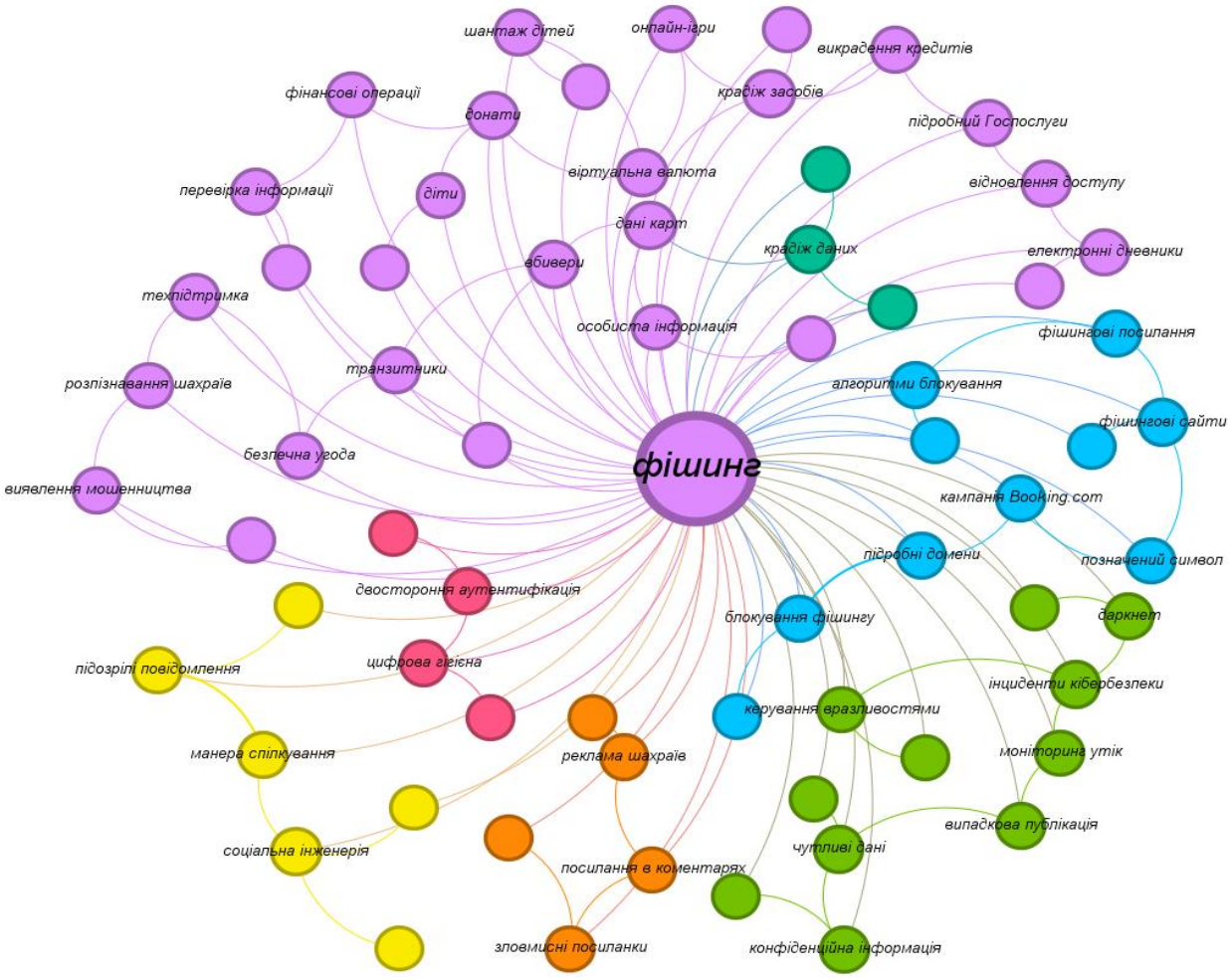

Рисунок 3 – Фрагмент семантичної мережі у домені фішингових атак

Отримана семантична мережа демонструє, що центральне поняття “фішинг” сформувало навколо себе кластер ключових термінів, які описують різні аспекти цього поняття – кібершахрайство, сповіщення, атака, вірус, розсилка, соціальна інженерія тощо. Така

структуризація вказує на природну здатність великих мовних моделей виокремлювати смислові центри та групувати поняття за семантичною близькістю. Мережа, згенерована автоматично, не лише відтворює вже відомі зв'язки, а й виявляє нові контекстні асоціації, що свідчить про глибоке розуміння латентних відношень у текстах.

Цей результат узгоджується з сучасними дослідженнями, які підкреслюють, що LLM дедалі частіше застосовують не лише для генерації текстів, а й для розширення баз знань і формування онтологій на основі неструктурованих даних. У даній роботі модель виконала роль інтелектуального агрегатора: вона автоматично вилучила з текстового корпусу нові факти, терміни та взаємозв'язки, перетворивши їх у формалізовану структуру знань, придатну для подальшої аналітики, візуалізації та побудови експертних систем.

Таким чином, створена семантична мережа підтверджує ефективність підходу до автоматизованого онтологічного моделювання, що поєднує генеративні можливості LLM із логічною строгістю структур знань.

Виявлені зв'язки між поняттями дозволяють моделі фактично реконструювати предметну область у вигляді когнітивної карти. Такий підхід відкриває перспективу переходу від статичних онтологій до динамічних знаннєвих мереж, здатних самостійно оновлюватися на основі потоку нової інформації, що робить його особливо цінним для сфер із швидкозмінним контекстом, зокрема кібербезпеки та інформаційного аналізу.

Висновки. У статті запропоновано та експериментально перевірено безкодовий підхід до побудови семантичних мереж на основі промпт-інжинірингу з використанням великих мовних моделей (LLM). Наукова новизна роботи полягає в композиційному поєднанні формальних примітивів промпт-програмування – умови, циклу та функції – для створення алгоритмічної логіки обробки неструктурованих текстів без застосування традиційного програмного коду. Це дозволяє трансформувати LLM із генератора тексту в інструмент автоматизованої онтологічної інженерії, здатний виділяти поняття, встановлювати між ними семантичні зв'язки та формувати структуровані графи знань.

Практична новизна підходу продемонстрована на прикладі побудови семантичної мережі у предметній області фішингових кібератак. У ході експерименту модель GPT-5 обробила 48 новинних повідомлень і сформувала близько 70 релевантних пар пов'язаних понять, що відобразили цілісну структуру домену: центральне поняття "фішинг" було пов'язане з такими похідними термінами, як "соціальна інженерія", "підроблена сторінка", "шкідливе ПЗ", "спам-кампанія" тощо. Додаткове розширення мережі за допомогою семантично близьких термінів підвищило її повноту та термінологічну глибину, що підтверджує здатність методу до ефективного збагачення базової онтології.

Запропонований фреймворк забезпечує масштабованість, відтворюваність та доступність: користувач без навичок програмування може формулювати аналітичне завдання природною мовою, а LLM автоматично генерує відповідну «промпт-програму», що реалізує логіку обробки. Такий підхід суттєво знижує поріг входження в онтологічне моделювання й дозволяє оперативно оновлювати знання в динамічних галузях, зокрема в кібербезпеці, медицині, фінансових технологіях та аналітиці OSINT-даних.

Подальші дослідження планується спрямувати на адаптацію методу до різних архітектур LLM, розробку механізмів верифікації згенерованих фактів (зокрема для мінімізації галюцинацій) та інтеграцію з існуючими формальними системами представлення знань (наприклад, OWL-онтологіями). У довгостроковій перспективі запропонований підхід може стати основою для розвитку нейросимволічних інтелектуальних систем, які поєднують генеративну гнучкість LLM із строгою логікою символічного моделювання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] A.S. Lippolis et al., “Ontology generation using large language models”, in *Proc. 22nd Eur. Semantic Web Conf. (ESWC 2025)*, Portoroz, Slovenia, 2025, pp. 321-341. doi: https://doi.org/10.1007/978-3-031-94575-5_18.

[2] D. Lande, and L. Strashnoy, “Semantic AI framework for prompt engineering”, *SSRN*, 14 p., 2025. doi: https://doi.org/10.2139/ssrn.5172867.

[3] L. Beurer-Kellner, M. Fischer, and M. Vechev, “Prompting is programming: A query language for large language models”, in *Proc. ACM on Programming Languages*, vol., iss. PLDI, art. 186, pp. 1946-1969, 2023. doi: https://doi.org/10.1145/3591300.

[4] C. Shimizu, and P. Hitzler, “Accelerating knowledge graph and ontology engineering with large language models”, *J. Web Semantics*, vol.85, art. 100862, 2025. doi: https://doi.org/10.1016/j.websem.2025.100862.

[5] R.S. Dehal, M. Sharma, and E. Rajabi, “Knowledge graphs and their reciprocal relationship with large language models”, *Mach. Learn. Knowl. Extraction*, vol. 7, iss. 2, art. 38, 2025. doi: https://doi.org/10.3390/make7020038.

[6] A. Mavridis, S. Tegos, C. Anastasiou, M. Papoutsoglou, and G. Meditskos, “Large language models for intelligent RDF knowledge graph construction: Results from medical ontology mapping”, *Frontiers Artif. Intell.*, vol. 8, art. 1546179, 2025. doi: https://doi.org/10.3389/frai.2025.1546179.

[7] Y. Xu et al., “LLM-Enhanced framework for building domain-specific lexicon for urban power grid design”, *Appl. Sci.*, vol. 15, iss. 8, art. 4134, 2025. doi: https://doi.org/10.3390/app15084134.

Стаття надійшла до редакції 14.09.2025.

REFERENCE

[1] A.S. Lippolis et al., “Ontology generation using large language models”, in *Proc. 22nd Eur. Semantic Web Conf. (ESWC 2025)*, Portoroz, Slovenia, 2025, pp. 321-341. doi: https://doi.org/10.1007/978-3-031-94575-5_18.

[2] D. Lande, and L. Strashnoy, “Semantic AI framework for prompt engineering”, *SSRN*, 14 p., 2025. doi: https://doi.org/10.2139/ssrn.5172867.

[3] L. Beurer-Kellner, M. Fischer, and M. Vechev, “Prompting is programming: A query language for large language models”, in *Proc. ACM on Programming Languages*, vol., iss. PLDI, art. 186, pp. 1946-1969, 2023. doi: https://doi.org/10.1145/3591300.

[4] C. Shimizu, and P. Hitzler, “Accelerating knowledge graph and ontology engineering with large language models”, *J. Web Semantics*, vol.85, art. 100862, 2025. doi: https://doi.org/10.1016/j.websem.2025.100862.

[5] R.S. Dehal, M. Sharma, and E. Rajabi, “Knowledge graphs and their reciprocal relationship with large language models”, *Mach. Learn. Knowl. Extraction*, vol. 7, iss. 2, art. 38, 2025. doi: https://doi.org/10.3390/make7020038.

[6] A. Mavridis, S. Tegos, C. Anastasiou, M. Papoutsoglou, and G. Meditskos, “Large language models for intelligent RDF knowledge graph construction: Results from medical ontology mapping”, *Frontiers Artif. Intell.*, vol. 8, art. 1546179, 2025. doi: https://doi.org/10.3389/frai.2025.1546179.

[7] Y. Xu et al., “LLM-Enhanced framework for building domain-specific lexicon for urban power grid design”, *Appl. Sci.*, vol. 15, iss. 8, art. 4134, 2025. doi: https://doi.org/10.3390/app15084134.

DMYTRO LANDE,
OLEKSANDR RYBAK

NO-CODE APPROACH TO BUILDING SEMANTIC NETWORKS BY MEANS OF PROMPT ENGINEERING

The article proposes a no-code approach to building semantic networks by means of prompt engineering using large language models (LLMs). A framework is developed in which the basic primitives – condition, loop, and function – are combined into compositional structures that ensure automated extraction of concepts, establishment of links between them, and construction of formalized knowledge graphs. The proposed method relies on the no-code principle, which makes it possible to describe algorithmic logic in natural language without involving program code. This enables the use of large language models not only as text generators but as full-fledged tools for constructing knowledge structures.

Within the study, an LLM is considered as a driver for automated ontology engineering. The model interprets natural-language instructions as formalized actions, which makes it possible to iteratively extract key concepts, determine types of relations, and form knowledge graphs with a given logical sequence. Particular attention is paid to the field of cybersecurity, where rapid creation and updating of threat ontologies is crucial for timely response to new attack vectors.

The practical implementation of the approach is carried out on the example of building a semantic network in the topic of phishing attacks. In the course of the experiment, the GPT-5 language model processed 48 news reports, automatically forming about 70 pairs of related concepts. The resulting knowledge graph reflected an integral structure of the domain, where the central concept “phishing” is combined with numerous derivative terms: cyberattack, social engineering, spoofed page, malicious software, etc. The results of the experiment prove that the proposed methodology ensures the relevance of inter-concept relations and the enrichment of the basic terminology with semantically related concepts.

The integration of large language models into the process of ontological modeling simplifies the creation of knowledge structures, lowers the entry barrier for users without programming experience, and opens up prospects for the development of neuro-symbolic systems that combine the generative capabilities of models with formal methods of knowledge representation. The proposed approach has high potential for practical application in fields that require dynamic knowledge updating – primarily in cybersecurity, medicine, financial technologies, and data analytics.

Keywords: semantic knowledge modeling, large language models (LLM), cybersecurity, OSINT, text analytics, formal control primitives, prompt engineering.

Ланде Дмитро Володимирович, доктор технічних наук, професор, завідувач кафедри, Навчально-науковий фізико-технічний інститут Національного технічного університету України “Київський політехнічний інститут імені Ігоря Сікорського”, Київ, Україна, ORCID 0000-0003-3945-1178, dwlande@gmail.com.

Рибак Олександр Олегович, аспірант, Інститут спеціального зв’язку та захисту інформації Національного технічного університету України “Київський політехнічний інститут імені Ігоря Сікорського”, Київ, Україна, ORCID0009-0004-1033-1599, rybak.oleksandr01@gmail.com.

Lande Dmytro, doctor of technical sciences, professor, chair of the department, Educational and scientific physico-technical institute at the National technical university of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”, Kyiv, Ukraine.

Rybak Oleksandr, postgraduate student, Institute of special communication and information protection at the National technical university of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”, Kyiv, Ukraine.