DOI: https://doi.org/10.23939/csn2024.02.281
УДК 004.8

ТЕХНІКИ ПРОМПТИНГУ ДЛЯ ПОКРАЩЕННЯ ВИКОРИСТАННЯ ВЕЛИКИХ МОВНИХ МОДЕЛЕЙ

І.Ю. Юрчак, О.О. Кичук, В.М.Оксентюк, А.О.Хіч

Національний університет "Львівська політехніка",
кафедра систем автоматизованого проектування
E-mail: Iryna.Y.Yurchak@lpnu.ua, Olha.Kychuk.knm.2020@lpnu.ua, Vira.M.Oksentyuk@lpnu.ua, Andrii.O.Khich@lpnu.ua

Робота присвячена дослідженню базових технік складання запитів для підвищення ефективності використання великих мовних моделей. Значну увагу приділено питанню інженерії запитів (промптингу). Детально розглянуто різноманітні техніки: промптинг без зразка, зі зворотним зв'язком, з кількома прикладами, ланцюжкове мислення, дерево думок, інструкція для налаштування. Значну увагу приділено технологіям Реакція та Дія (Reaction & Act Prompting) та Доповнена пошукова генерація (Retrieval Augmented Generation, RAG) як критично важливих чинників забезпечення ефективної взаємодії з ВММ. Висвітлено особливості застосування цих технік та їхній вплив на результат. Однак, використання повного потенціалу вимагає ретельного підходу та врахування особливостей застосування.

Здійснено огляд параметрів великих мовних моделей, таких як температура, тор P, максимальна кількість токенів, стоп-послідовності, штрафи за частоту та присутність тощо. Зазначено, що розроблення запитів є ітеративним процесом, який передбачає послідовне випробування різних варіантів для досягнення оптимальних результатів. Всі наведені у дослідженні техніки підкріплено наочними прикладами з отриманими результатами. Зазначено, для яких типів дана техніка буде більш доречною. У результатах дослідження наведено порівняння як базових технік, так і складніших технологій ReAct та RAG.

Інженерія запитів — це ключова технологія ефективного використання великих мовних моделей. Вона актуальна у зв'язку зі зростанням застосування штучного інтелекту у всіх сферах діяльності людства, і її роль лише збільшуватиметься з розвитком технологій. Вміння правильно формулювати запити стає важливою навичкою, необхідною для роботи з сучасними великими моделями, особливо в умовах їхньої універсальності та складності..

Ключові слова: великі мовні моделі, інженерія запитів, промптинг, техніки запитів, генерування контенту.

1. Вступ

Інженерія запитів (Prompt Engineering) є критично важливою практикою для ефективної роботи з великими мовними моделями (ВММ). Це галузь, що займається розробкою та вдосконаленням підходів до формулювання текстових запитів (промптів), які подаються на вхід мовної моделі з метою отримання бажаних результатів.

Досвідчені інженери запитів проектують вхідні дані, які оптимально взаємодіють з іншими вхідними даними в мовних моделях. Ці вхідні дані допомагають отримувати якісніші відповіді, та забезпечують краще виконання різних завдань, такі як написання художніх текстів, складання

програмного коду, витягування анотації з великого обсягу тексту, розлогі діалоги, створення цифрового мистецтва, розпізнавання та генерація зображень.

2. Огляд літературних джерел

Prompt Engineering можна визначити як процес розробки та оптимізації текстових запитів Завданням промтингу полягає в тому, щоб правильно формулювати запити до моделі, враховуючи контекст завдання та очікування від результату. У деяких випадках невеликі зміни у формулюванні підказки можуть суттєво змінити висновок моделі, тому вміння працювати з підказками стає важливою навичкою [1].

Глибоке навчання забезпечило багато останніх досягнень у сфері великих мовних моделей (ВММ) – наприклад ChatGpt або Gemini [4-6]. Базові моделі представляють значну еволюцію в межах глибокого навчання, оскільки можуть обробляти масивні та різноманітні набори неструктурованих даних. ВММ, навчені на цих моделях, можуть виконувати багато завдань, таких як відповідати на питання, класифікувати, редагувати, узагальнювати та створювати новий вміст [2].

Мета правильних запитів до мовних моделей полягає в тому, щоб розробити ефективні методики формулювання промптів, які допоможуть максимально розкрити потенціал ВММ та отримувати змістовні, очікуванні та безпечні результати. Для досягнення цієї мети застосовується низка підходів та технік [9].

Вивчення та застосування технік інженерії запитів дозволяє користувачам налагодити більш змістовну та контрольовану взаємодію з великими мовними моделями [7-8]. Це допомагає уникнути небажаної поведінки моделі, підвищити якість та достовірність результатів, а також розширити діапазон завдань, які можна вирішувати за допомогою ВММ.

Інженерія запитів відіграє ключову роль у забезпеченні етичного та відповідального використання великих мовних моделей. Належно сформульовані промпти дозволяють звести до мінімуму ризики генерування шкідливого, незаконного чи упередженого контенту [3], а також спрямовують моделі на толерантну та неупереджену поведінку.

3. Постановка задачі

Враховуючи, що безпосереднє налаштування ВММ під конкретні завдання для більшості користувачів і розробників є непрактичним або недосяжним, звернено увагу на оптимізацію запитів до ВММ. Техніка інженерії промптів, яка передбачає створення точних, специфічних для завдання інструкцій природною мовою вручну стала центральною сферою даного дослідження.

Метою є полегшення розуміння основних концепцій формулювання питань до мовних моделей, дослідити їхні можливості та покращити обізнаність користувачів щодо поведінки моделей при використанні різних промптів. Проведено ґрунтовні експерименти з Gemini 1.0 Pro, ChatGPT-4o та Claude 3 Sonnet для перевірки ефективності запропонованих принципів у розробці запитів та промптів [4-6].

Для цього створено різні запити, кожен з яких відповідає одному певному напрямку. Ці запити відформатовано згідно з низкою технік промптингу. Отримані результати порівнюються між собою за встановленими метриками, після чого зроблено висновки щодо ефективності використаних промптів для конкретної теми запиту.

Наприкінці узагальнено висновки дослідження та надано рекомендації щодо оптимального використання технік промптингу для покращення взаємодії з ВММ у різноманітних контекстах. Результати цього дослідження допоможуть користувачам максимізувати потенціал сучасних мовних моделей та отримувати від них більш релевантні, змістовні та зрозумілі відповіді.

4. Налаштування великих мовних моделей

Під час розробки та тестування запитів для взаємодії з ВММ зазвичай використовується API (Application Programming Interface). Є багато параметрів, які можна налаштувати, щоб отримати різні результати для запитів. Потрібно експериментувати, щоб визначити правильні налаштування для різних сценаріїв використання, оскільки таке налаштування важливе для підвищення надійності та бажаності відповідей [3].

Нижче наведено загальні параметри, з якими можуть зіткнутися люди, які використовують різні типи ВММ (рис.1).

Рис.1. Параметри великих мовних моделей

Температура – це параметр, який визначає ступінь передбачуваності відповіді, чим нижча температура, то більш передбачуваним буде результат. При низькій температурі генерація відповіді завжди вибиратиме токен з найбільшою ймовірністю. З підвищенням температури вибір токенів стає менш передбачуваним, що призводить до більш різноманітних або креативних відповідей, оскільки зростає можливість вибору інших, менш очевидних токенів [7-8].

На практиці низьку температуру можна використовувати для завдань, що потребують точності, наприклад, для отримання відповідей, де важливі достовірність та стисліть. Для творчих завдань, таких як складання художніх текстів чи віршів, більш високе значення температури може бути корисним, оскільки воно збагачує відповіді.

Top P – визначає кількість слів, які можна розглянути для вибору наступного слова в генерації тексту. Показник обмежує кількість слів, з яких модель може вибирати наступне випадковим чином. Це сприяє створенню відповідей, в яких використовуються різні слова. Відмінності між температурою і Top P полягає у тому, що розмір набору слів, з якого обирається наступне слово, контролюється параметром Top P, а температура регулює різноманітність і випадковість вибору слів із цього набору [8].

Рекомендується змінювати або Top P або температуру, але не обидва.

Максимальна довжина ланцюжка – допомагає керувати кількістю токенів, які генерує модель під час відповіді. Визначення максимальної довжини допомагає запобігти отриманню довгих або нерелевантних відповідей і контролювати витрати, оскільки розгорнутий результат може збільшити час виконання виклику API.

Стоп-послідовності - це рядок, який повідомляє моделі, коли їй варто зупинити генерацію токенів. Такий параметр спроможний контролювати довжину і структуру відповіді моделі. Користувач може встановити конкретні фрази чи стоп-слова, після яких система повинна зупинити генерацію відповіді. Наприклад, можна вказати моделі генерувати списки, що містять не більше 10 елементів, додавши «11» як стоп-послідовність.

Штраф за частоту - застосовує до наступної лексеми штраф, на основі того, скільки разів ця лексема вже з'являлася у відповіді та запиті. Чим вищий штраф, тим менша ймовірність того, що вживане слово з'явиться знову, а замість нього система використає менш поширені вирази. Призначаючи більший штраф для токенів, які з'являються частіше, цей параметр допомагає моделі відповідати з меншою кількістю повторюваних слів, а менше значення, навпаки, сприяє застосуванню поширених фраз.

Штраф за присутність - подібно до штрафу за частоту, ця змінна накладає обмеження на повторювані токени і є постійною. В залежності від даного атрибуту модель реагує на певну кількість

повторень одного твердження. Збільшення значення цього параметра зменшує кількість повторень, тоді як використання меншого значення навпаки збільшує кількість повторень.

Рекомендується змінювати штраф за присутність або частоту, але не обидва.

Окрім загальних параметрів налаштування можна відмітити інші ключові елементи, а саме:

- Інструкція – запит або завдання від користувача, які модель повинна виконати, щоб на виході згенерувати бажаний результат.
- Вхідні дані – інформація або питання, на які він очікує відповідь.
- Контекст – додаткові дані для кращого розуміння мовної моделі, які результати очікуються на виході.
- Вихідний показник – формат або тип вихідних даних, що мають бути згенеровані.

Розробка промптів є ітеративним процесом, який передбачає послідовне випробування різних варіантів для досягнення оптимальних результатів. Починаючи з простих промптів і поступового додавання до них додаткових елементів та контексту, можна досягти кращих результатів. Протягом процесу важливо систематично вдосконалювати постановку запитів. Ефективним підходом є розділення завдання на менші складові частини та поступове вдосконалення результатів кожної з них. Цей підхід спрощує процес розробки промптів з самого початку.

Можна розробляти ефективні запити для різних простих завдань, використовуючи команди для вказівки моделі на те, що потрібно досягти, наприклад: «Впорядкувати», «Написати», «Підсумувати», «Класифікувати», «Перекласти» тощо. Інші рекомендації включають розміщення інструкцій на початку промпта. Додатково рекомендується використання чіткого роздільника, наприклад «###», для відокремлення інструкцій і контексту [1].

5. Техніки промптингу

Покращення формулювання запитів сприяє кращому виконанню різних завдань. Нижче розглянуто основні техніки промптингу, що є ключовим елементом в оптимізації використання ВММ.

Промптинг без зразка (Zero-Shot Prompting)

Промптинг без зразка (Нульовий промптинг) — це підхід, в якому мовна модель виконує завдання без попереднього навчання на прикладах. Вона просто отримує інструкцію у вигляді запиту та генерує відповідь, ґрунтуючись на своїх вже існуючих знаннях. У наведеному прикладі модель відразу виконує завдання (переклад), не отримавши до цього навчальних прикладів чи інструкцій, як потрібно перекладати. (рис.2).

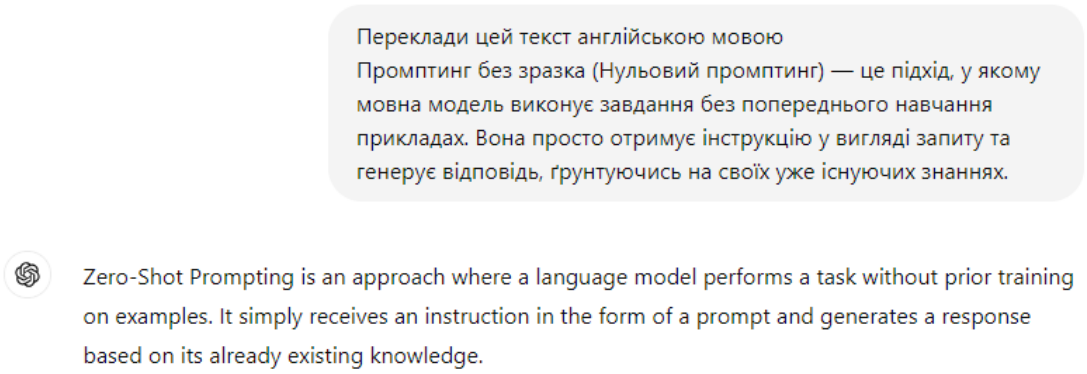


Рис.2. Приклад використання техніки нульового промптингу

У нульовому промптингу модель використовує раніше набуті знання для виведення вимог завдання та генерації відповідних відповідей, демонструючи свою здатність узагальнювати та застосовувати вивчені концепції в нових ситуаціях [5]. Цей підхід має особливе значення в сценаріях, де непрактично навчати модель для кожного можливого завдання або де діапазон потенційних завдань занадто широкий для повного передбачення.

Використання нульового промптингу дозволяє взаємодіяти з мовною моделлю більш природньо, не надаючи конкретних деталей для кожної можливої теми розмови. Замість цього, можна надати моделі деякі ключові слова або загальну ідею, і вона використовуватиме свої наявні знання для генерації відповіді.

Техніки промптингу для покращення використання великих мовних моделей

Нульовий промптинг добре працює, коли завдання не надто спеціалізоване і не потребує детального контекстного розуміння. Відповіді, що отримано такою технікою можуть не задовольнити потреб у більш складних завданнях, оскільки розуміння моделі окремих завдань є обмеженим.

Промптинг зі зворотним зв'язком (Feedback-Based Prompting)

Промптинг зі зворотним зв'язком – це техніка, яка використовується для покращення якості взаємодії з мовними моделями через постійне надання зворотного зв'язку. Основна ідея полягає в тому, щоб отримати зворотний зв'язок від користувача на кожному етапі взаємодії з моделлю, та використовувати цей зв'язок для коригування подальших запитів. Такий підхід дозволяє поступово покращувати відповіді моделі та робити їх більш релевантними.

Якщо модель відхилилася від теми, зворотний зв'язок допомагає привернути її увагу до ключових аспектів запиту. Модель може не враховувати контекст належним чином, але зворотній зв'язок допомагає уточнити, який контекст має бути включений у відповідь. Користувач може вказати, що стиль відповіді був надто формальним або неформальним, що допоможе коригувати модель.

Промптинг зі зворотнім зв'язком є ітераційним процесом. Він працює як свого роду "навчання" моделі на ходу, при якому користувач формує уточнені запити, щоб поступово покращити якість взаємодії. Ця техніка важлива у випадках, коли потрібна максимальна точність або модель повинна відповідати певним стилям та вимогам.

Промптинг з невеликою кількістю прикладів (Few-Shot Prompting)

Користувачі не завжди можуть бути задоволені відповідями великих мовних моделей, якщо вони вводять лише один приклад або жодного. Це пов'язано з тим, запит може не містити всіх важливих деталей, необхідних для правильної роботи моделі. Промптинг з невеликою кількістю прикладів - це техніка, де моделі надається кілька прикладів бажаного формату відповіді перед основним питанням. Це допомагає моделі краще зрозуміти контекст та очікуваний формат відповіді.

Few-shot можна використовувати для забезпечення контекстного навчання, коли надаються демонстрації в запиті, щоб спрямувати модель на вищу продуктивність. Демонстрації слугують як контекст для наступних прикладів, в яких потрібно, щоб модель генерувала відповідь. Користувачі можуть керувати моделлю, надаючи їй кілька прикладів, зазвичай від двох до п'яти [5]. Все, що перевищує цей діапазон, може стати надто складним для чат-бота.

На лістингу 1 наведено приклад генерації тексту у певному стилі за наступним завданням.

Лістинг 1.

```
Завдання: Написати короткий опис погоди у депресивному стилі.
Приклад 1:
Тема: Сонячний день
Опис: Безжальне сонце випалювало тротуар, перетворюючи місто на розпечену сковорідку. Тіні зіщулилися, сховавшись по кутках, наче злочинці від закону.
Приклад 2:
Тема: Дощовий вечір
Опис: Дощ барабанив по дахах, як пальці нервового піаніста. Вулиці блищали, відбиваючи неонові вогні, приховуючи таємниці у своїх темних калюжах.
Приклад 3:
Тема: Туманний ранок
Опис: Туман огорнув місто савану таємниці. Силуети будинків ледь проступали, як привиди минулого, зберігаючи свої секрети у молочному серпанку.
Тепер опишіть наступну тему у депресивному стилі:
Тема: Снігова ніч
Опис:
```

За таким запитом надано наступну відповідь (рис.3).

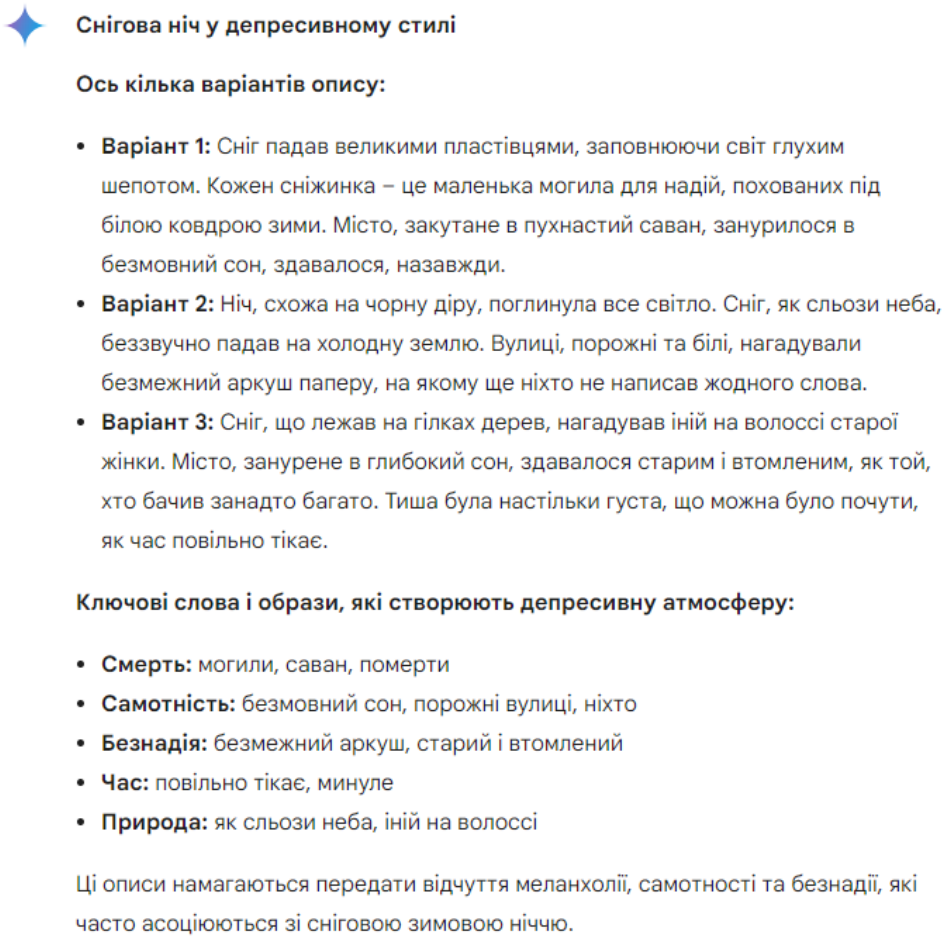


Рис.3. Приклад виконання завдання за технікою Few-Shot

Загалом, створення прикладів є корисним для вирішення конкретних проблем, але іноді інформації, яка була вивчена моделлю недостатньо для успішного виконання завдання. В такій ситуації слід почати розглядати налаштування моделі або експерименти зі складнішими методами формулювання запитів.

Цей метод можна застосовувати до різних завдань, таких як:

- Класифікація емоційного забарвлення тексту.
- Генерація тексту у певному стилі.
- Витягування анотації з великої кількості інформації.
- Переклад на іншу мову із збереженням певного тону чи стилю.
- Вирішення математичних завдань в певний спосіб.

Ланцюжкове мислення (Chain-of-Thought Prompting)

Техніка ланцюжкового мислення передбачає структурування запитів таким чином, щоб модель слідувала поетапному процесу міркувань, аналогічному тому, як людина може логічно розкласти проблему на складові. На відміну від методів прямих відповідей, коли модель надає негайні відповіді, запити у формі ланцюжка думок спрямовані на те, щоб робити процес мислення ВММ зрозумілим і чітким.

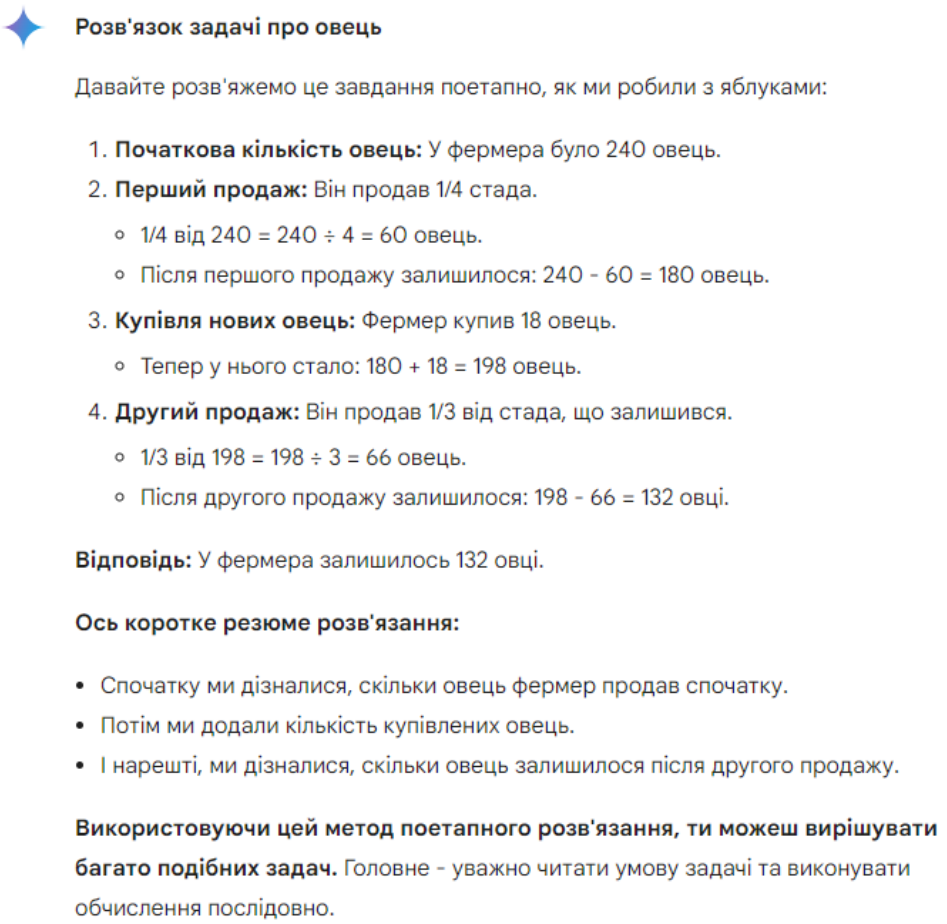
На практиці цей підхід передбачає надання промпта, який не лише ставить питання, але й заохочує модель показати свою роботу. Наприклад, при розв'язанні математичної задачі модель детально описуватиме кожен крок в обчисленнях. У більш складних завданнях, таких як інтерпретація тексту або планування подій, модель викладатиме своє розуміння ситуації, розглядатиме різні фактори, враховуватиме варіанти і робитиме висновок.

На лістингу 2 наведено приклад завдання для вирішення за технікою ланцюжкового мислення. Моделі показано приклад вирішення завдання.

Лістинг 2.

```
Завдання: Розв'язати текстове завдання з математики.
Приклад:
Запитання: У магазині було 120 яблук. Вранці продали 1/3 всіх яблук, а по обіді ще 30 яблук. Скільки яблук залишилось у магазині?
Рішення:
1. Спочатку у магазині було 120 яблук.
2. Вранці продали 1/3 всіх яблук:
 1/3 від 120 = 120 ÷ 3 = 40 яблук продано вранці.
3. Після ранкового продажу залишилося:
 120 – 40 = 80 яблук.
4. Після обіду продали ще 30 яблук:
 80 – 30 = 50 яблук залишилося.
Відповідь: У магазині залишилось 50 яблук.
Тепер вирішіть наступне завдання, використовуючи аналогічний ланцюжок міркувань:
Питання: У фермера було стадо із 240 овець. Він продав 1/4 стада, потім купив 18 нових овець, а потім ще продав 1/3 від стада, що залишився. Скільки овець залишилось у фермера?
Рішення:
```

За таким запитом надано наступну відповідь (рис.4).



Розв'язок задачі про овець

Давайте розв'яжемо це завдання поетапно, як ми робили з яблуками:

1. **Початкова кількість овець:** У фермера було 240 овець.
2. **Перший продаж:** Він продав 1/4 стада.
   - 1/4 від 240 = 240 ÷ 4 = 60 овець.
   - Після першого продажу залишилося: 240 - 60 = 180 овець.
3. **Купівля нових овець:** Фермер купив 18 овець.
   - Тепер у нього стало: 180 + 18 = 198 овець.
4. **Другий продаж:** Він продав 1/3 від стада, що залишився.
   - 1/3 від 198 = 198 ÷ 3 = 66 овець.
   - Після другого продажу залишилося: 198 - 66 = 132 овці.

Відповідь: У фермера залишилось 132 овці.

Ось коротке резюме розв'язання:

- Спочатку ми дізналися, скільки овець фермер продав спочатку.
- Потім ми додали кількість купівлених овець.
- І нарешті, ми дізналися, скільки овець залишилося після другого продажу.

Використовуючи цей метод поетапного розв'язання, ти можеш вирішувати багато подібних задач. Головне - уважно читати умову задачі та виконувати обчислення послідовно.

Рис.4. Приклад виконання завдання за технікою Chain-of-Thought

Використання техніки ланцюжкового мислення буде ефективною для виконання наступних завдань: розв'язання математичних завдань, логічного аналізу, покрокового планування, розбирання складних ситуацій, обґрунтування висновків чи рішень. Ця техніка допомагає моделі краще зрозуміти процес вирішення, робить міркування прозорішими і дозволяє перевірити кожен крок логічного ланцюжка.

Ефективною вважається поєднанням технік Chain-of-Thought та Few-Shot для отримання задовільних результатів у складних завданнях, які вимагають розмірковувань перед відповіддю [5, 7].

На лістингу 3 наведено приклад, що демонструє комбінацію техніки Few-Shot (надається кілька прикладів вирішення завдання) з Chain-of-Thought підходом (надається приклад покрокових

розтлумачень в кожному прикладі).

Лістинг 3.

```
Завдання: Розв'язати логічні завдання, пояснюючи перебіг міркувань.
Приклад 1:
Питання: Якщо всі кішки люблять рибу, а Мурка – кішка, чи любить Мурка рибу?
Рішення:
1. Дано: Всі кішки люблять рибу.
2. Дано: Мурка – кішка.
3. Якщо Мурка – кішка, а всі кішки люблять рибу, то Мурка має любити рибу.
Відповідь: Так, Мурка любить рибу.
Приклад 2:
Питання: Іван вищий за Петра. Петро вищий за Миколу. Хто найвищий?
Рішення:
1. Порівняємо ріст Івана і Петра: Іван вищий за Петра.
2. Порівняємо ріст Петра і Миколи: Петро вищий за Миколу.
3. З цього випливає, що Іван вищий за Петра, а Петро вищий за Миколу.
4. Значить, Іван вищий за Петра і Миколу.
Відповідь: Іван найвищий.
Тепер вирішіть наступне завдання, використовуючи аналогічний ланцюжок міркувань:
Питання: Всі учні у класі люблять або футбол, або баскетбол. 15 учнів люблять футбол, 7
учнів люблять обидва види спорту. У класі лише 20 учнів. Скільки учнів люблять лише
баскетбол?
```

За таким запитом надано наступну відповідь (рис.5).

Розв'язок логічного завдання про учнів

Аналіз умови:

- Усі учні люблять або футбол, або баскетбол (або обидва).
- 15 учнів люблять футбол.
- 7 учнів люблять обидва види спорту (це означає, що вони входять як в групу любителів футболу, так і в групу любителів баскетболу).
- Всього в класі 20 учнів.

Розв'язання:

1. **Визначимо кількість учнів, які люблять тільки футбол:**
   - Оскільки 7 учнів люблять обидва види спорту, а 15 люблять футбол загалом, то тільки футбол люблять: 15 - 7 = 8 учнів.
2. **Визначимо загальну кількість учнів, які люблять баскетбол (включаючи тих, хто любить обидва види):**
   - Всі учні класу або люблять футбол, або баскетбол, або обидва. Отже, кількість учнів, які люблять баскетбол, дорівнює загальній кількості учнів в класі мінус кількість учнів, які люблять тільки футбол: 20 - 8 = 12 учнів.
3. **Визначимо кількість учнів, які люблять тільки баскетбол:**
   - Ми вже знаємо, що 7 учнів люблять обидва види спорту. Отже, з 12 учнів, які люблять баскетбол, тільки баскетбол люблять: 12 - 7 = 5 учнів.

Відповідь: Лише баскетбол люблять 5 учнів.

Рис.5. Приклад виконання завдання за поєднанням технік Chain-of-Thought та Few-Shot

Результати дослідження різняться з попередніми, де використано лише одну техніку промптингу. У даному експерименті отримані результати вказують на ефективність підходу щодо поєднання Chain-of-Thought та Few-Shot, оскільки окрім правильної відповіді модель надає пояснення власних міркувань при розв’язанні задачі (рис.6).

Пояснення міркувань:

Ми розв'язали задачу, поступово виключаючи з загальної кількості учнів ті групи, про які маємо інформацію. Спочатку ми визначили кількість учнів, які люблять тільки футбол, потім - загальну кількість любителів баскетболу і, нарешті, віднявши з цієї кількості тих, хто любить обидва види спорту, дізналися кількість учнів, які люблять тільки баскетбол.

Ключовий момент:

Важливо розуміти, що учні, які люблять обидва види спорту, входять в обидві групи (любителів футболу і любителів баскетболу). Тому, коли ми підраховуємо кількість учнів, які люблять тільки один вид спорту, ми повинні відняти цих "універсальних" учнів.

Цей тип задач часто розв'язують за допомогою діаграм Ейлера-Венна, які допомагають візуалізувати відношення між множинами.

Рис.6. Приклад пояснення міркувань моделі для розв'язання завдання

Цей метод дозволяє досягти успіхів у вирішенні поставлених завдань при наданні лише обмеженої кількості прикладів. Такий підхід не лише прискорює процес розв'язання завдань, але й зменшує обсяг даних, необхідних для навчання моделі. Впровадження методу ланцюгового мислення може значно зекономити ресурси та час при роботі з великими обсягами даних [4,8].

Дерево думок (Tree of Thoughts Prompting)

Техніка дерева думок використовує структуру, подібну до дерева рішень, для систематичного аналізу складних тем на різних рівнях глибини. Цей підхід дозволяє створювати ієрархічну структуру міркувань, де кожна «гілка» представляє окремий напрямок думки або конкретний аспект проблеми. Гілка може поділятися на «підгілки», що дозволяє виконати подальший деталізований аналіз. Загалом, техніка дерева думок містить два компоненти: генерацію думок та їх оцінювання (рис.7).

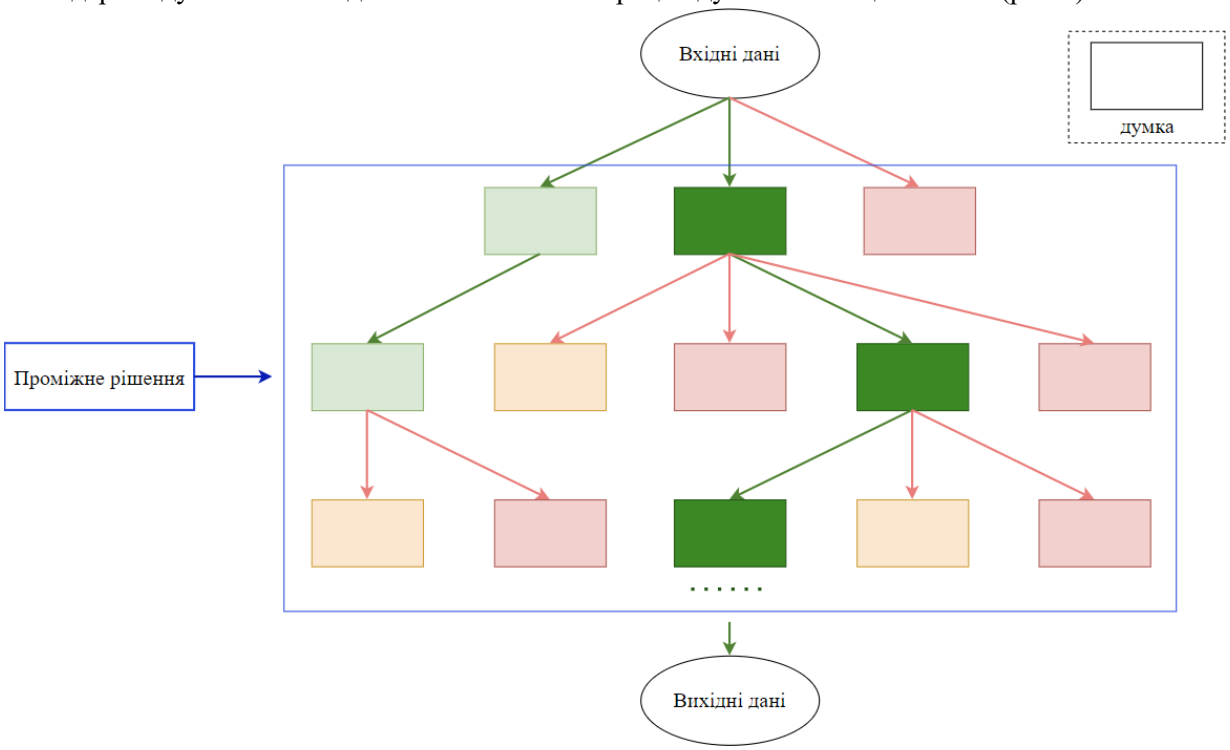


Рис.7. Приклад роботи техніки Tree of Thoughts

Використовуючи дерева думок досліджується кілька шляхів, кожен з яких складається з окремих думок, які представляють потенційні рішення проблеми. Разом кожен з цих шляхів та їхні індивідуальні думки утворюють дерево, яке досліджує простір вирішення проблеми [5].

Техніка дерева думок є розширеною версією ланцюжкового мислення, яка дозволяє моделі

розглядати кілька можливих шляхів вирішення проблеми, оцінювати їх та вибирати найкращий. На лістингу 4 наведено приклад застосування цієї техніки.

Лістинг 4.

```
Завдання: Знайти оптимальний маршрут для туриста, який хоче відвідати 3 визначні пам'ятки в Парижі за один день: Ейфелеву вежу, Лувр та Нотр-Дам. Турист має 8 годин (з 9:00 до 17:00).
Дерево думок:
```

На поставлене завдання модель надала наступну відповідь, що продемонстрована на лістингу 5.

Лістинг 5.

```
1. Початкова точка: Готель (близько до центру)
2. Можливі маршрути:
   2.1 Маршрут А: Ейфелева вежа -> Лувр -> Нотр-Дам
   2.2 Маршрут Б: Нотр-Дам -> Лувр -> Ейфелева вежа
   2.3 Маршрут В: Лувр -> Нотр-Дам -> Ейфелева вежа
3. Оцінка маршруту А:
   3.1 Час на Ейфелеву вежу: 2 години (включаючи черги)
   3.2 Переїзд до Лувру: 30 хвилин
   3.3 Час у Луврі: 3 години (мінімум для основних експонатів)
   3.4 Переїзд до Нотр-Даму: 15 хвилин
   3.5 Час на огляд Нотр-Даму ззовні: 30 хвилин
   Всього: 6 годин 15 хвилин. Залишається час на обід та невеликий запас.
4. Оцінка маршруту Б:
   4.1 Час на Нотр-Дам: 1 година (зовнішній огляд + коротка екскурсія всередину)
   4.2 Переїзд до Лувру: 15 хвилин
   4.3 Час у Луврі: 3 години
   4.4 Переїзд до Ейфелевої вежі: 30 хвилин
   4.5 Час на Ейфелеву вежу: 2 години
   Всього: 6 годин 45 хвилин. Щільний графік, мало часу на обід.
5. Оцінка маршруту В:
   5.1 Час у Луврі: 3 години (відкривається о 9:00, можна почати одразу)
   5.2 Переїзд до Нотр-Даму: 15 хвилин
   5.3 Час на Нотр-Дам: 1 година
   5.4 Переїзд до Ейфелевої вежі: 25 хвилин
   5.5 Час на Ейфелеву вежу: 2 години
   Всього: 6 годин 40 хвилин. Хороший баланс часу, є можливість для обіду.
6. Вибір оптимального маршруту:
   Маршрут В видається найбільш оптимальним:
   - Початок з Лувру дозволяє уникнути ранкових черг
   - Достатньо часу для кожної визначної пам'ятки
   - Є час на обід та невеликі перерви
   - Закінчується на Ейфелевій вежі, що зручно для вечірніх видів
Рекомендований маршрут: Лувр (9:00-12:00) -> обід (12:00-13:00) -> Нотр-Дам (13:15-14:15) -> Ейфелева вежа (14:40-16:40)
```

У цьому прикладі використано техніку дерева думок для аналізу різних варіантів маршруту та вибору оптимального. Модель здійснила наступні етапи розв’язання завдання.

- Для користувача надано три можливі маршрути.
- Для кожного маршруту проведено детальну оцінку часу.
- Модель врахувала різні фактори: час на переїзди, можливі черги, години роботи пам'яток.
- Здійснено оцінку та порівняння основних критеріїв завдання: загальний час та зручність кожного маршруту.
- На основі аналізу обрано оптимальний та найбільш вдалий варіант.
- Надано аргументи та обґрунтування на користь вибраного маршруту.

Ця техніка особливо корисна для складних завдань планування, прийняття рішень та оптимізації, де необхідно враховувати багато факторів та розглядати різні сценарії. Результати роботи моделі забезпечує прозорість процесу ухвалення рішень, глибокий аналіз кожного з наданих варіантів та знаходити неочевидні оптимальні рішення.

Налаштування інструкцій (Instruction Tuning Prompting)

Налаштування інструкцій - це потужна техніка, яка дозволяє мовним моделям покращити

здатність дотримуватися інструкцій шляхом оптимізації або зміни формулювання запитів, що допомагає моделі точніше розуміти та виконувати задані команди. Основна ідея у тому, щоб налаштовувати модель для виконання конкретних завдань, надаючи їй чіткі інструкції як підказки. Це дозволяє моделі краще розуміти, що від неї очікується, і генерувати більш точні та релевантні відповіді.

Модель отримує інструкції, які описують конкретні дії чи завдання. Наприклад, замість простого запиту "Напиши про собак", можна використовувати більш детальну інструкцію: "Опиши основні породи собак та їх характеристики". Додавання контексту у запит допомагає моделі краще орієнтуватися у завданні. Наприклад, "Який вид собак найкраще підходить для сім'ї з дітьми?".

Часто запити містять приклади бажаних відповідей або форматів. Це допомагає моделі зрозуміти, як має виглядати кінцевий результат, адаптуватися до різних стилів запитів та покращувати якість відповідей.

Застосування техніки налаштування інструкцій буде корисною для розв'язання наступних завдань:

- Пояснення складних тем або надання навчальних матеріалів.
- Допомога у написанні статей, блогів або інших текстів, де важливі конкретні вимоги та структура.
- Вирішення проблем або виконання завдань технічної підтримки.
- Створення чат-ботів або віртуальних помічників, здатних вести діалоги на різні теми та виконувати завдання користувачів.

Техніка налаштування інструкцій покращує взаємодію користувачів з моделлю, роблячи його більш інтуїтивним і зрозумілим. Користувачі можуть формулювати свої запити більш природно та отримати якісні та точні відповіді, що адаптовані до потреб користувачів.

На лістингу 6 наведено приклад застосування техніки налаштування інструкцій

Лістинг 6.

```
Напиши пост у соціальних мережах про новий смартфон, акцентуючи увагу на його камері та дизайні.
```

На поставлене завдання модель надала наступні розгалужені відповіді (рис.8)

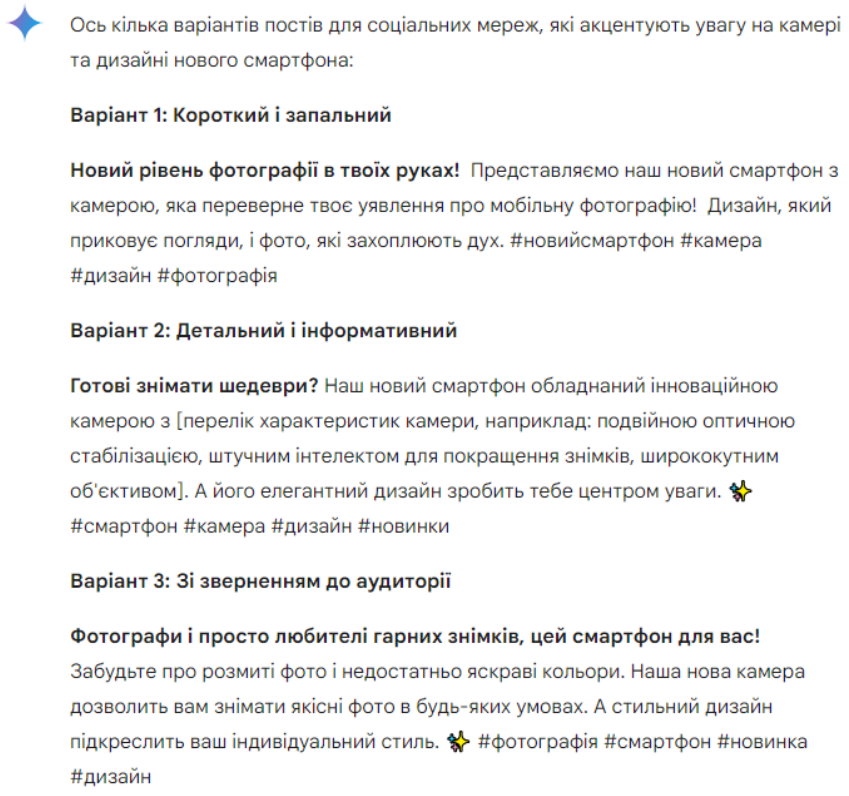


Рис.8. Приклад виконання завдання та технікою Instruction Tuning

За такою технікою модель надає додаткові відомості, які будуть у нагоді для користувача (рис.9).

Щоб зробити пост ще більш привабливим, можна додати:

- **Якісні фотографії** зроблені на цей смартфон, щоб продемонструвати можливості камери.
- **Короткий відеоролик**, який показує основні функції камери та дизайн смартфона.
- **Посилання на детальний огляд** або сторінку продукту.
- **Конкурс** з призами для підписників, які зроблять найкраще фото на цей смартфон.
- **Релевантні хештеги**, щоб збільшити охоплення аудиторії (наприклад, #смартфон, #камера, #новинка, #технології, #дизайн, #мобільнафотографія).

Приклад хештегів для різних соціальних мереж:

- **Instagram:** #newphone #cameraphone #mobilephotography #tech #gadget #design
- **Facebook:** #новийсмартфон #камера #фотографія #технології #дизайн
- **Twitter:** #newphonealert #cameraphone #mobilephotography #tech #gadget

Пам'ятайте: ключові слова повинні бути релевантними і відповідати інтересам вашої аудиторії.

Бажаю успішного просування вашого нового смартфона!

Чи хотіли б ви, щоб я написав пост для конкретної соціальної мережі?

Рис.9. Приклад надання додаткової інформації за технікою Instruction Tuning

Техніка налаштування інструкцій відкриває нові можливості для створення інтелектуальних систем, які можуть розуміти та виконувати людські інструкції. Навчання моделі на парах "інструкція-відповідь" зазвичай потребує менше даних та обчислювальних ресурсів. Модель може бути легко адаптована до нових завдань шляхом додавання нових інструкцій.

Реакція та Дія (Reaction & Act Prompting)

ReAct є технікою, що виходить за межі традиційних методів, таких як ланцюжкове мислення, шляхом інтеграції мислення, планування дій та використання зовнішніх джерел знань. Модель здійснює ітеративний процес: отримує вхідний запит, опрацьовує приклади з кількох запитів, формує ланцюжок міркувань і дій, взаємодіючи при цьому із зовнішніми джерелами знань для збагачення відповіді. Це дозволяє генерувати більш вичерпні, послідовні та фактично коректні відповіді.

Суть підходу полягає у розділенні складної задачі на кілька простих кроків. Це досягається дотриманням шаблону «Завдання – Думка – Дія – Результат» (Task – Think – Action – Result) - структурованого підходу до вирішення проблем та прийняття рішень (рис.10).

- Створюється набір даних, який містить різні типи вхідних даних та відповідні реакції та дії.
- Модель навчається на цьому наборі даних, щоб пов'язувати вхідні дані з відповідними реакціями та діями.
- Після навчання модель може реагувати на різні типи вхідних даних та виконувати відповідні дії.

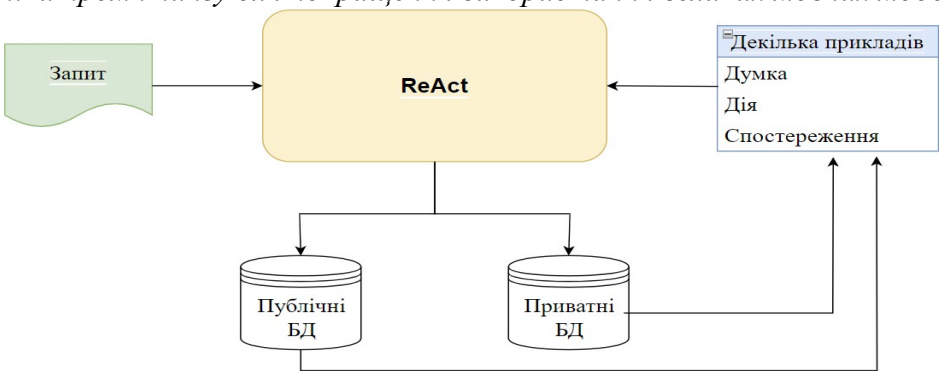


Рис.10. Діаграма компонентів технології ReAct

На лістингу 7 наведено приклад застосування техніки Reaction & Act.

Лістинг 7.

```
Системне повідомлення:
Ви є помічником, який допомагає користувачам з їхніми запитаннями про техніку. Ваша задача
- надавати точні та корисні відповіді. Якщо ви не знаєте відповіді, чесно скажіть про це.
Повідомлення користувача:
Який найкращий спосіб догляду за акумулятором ноутбука?
Повідомлення помічника:
Для догляду за акумулятором ноутбука рекомендується дотримуватися наступних порад:
1. Уникайте повного розрядження акумулятора. Намагайтеся заряджати його, коли рівень заряду
падає до 20-30%.
2. Не залишайте ноутбук підключеним до зарядного пристрою постійно. Це може призвести до
перегріву та зниження ємності акумулятора.
3. Використовуйте оригінальні зарядні пристрої та кабелі.
4. Зберігайте ноутбук у прохолодному місці, уникаючи прямих сонячних променів та високих
температур.
5. Якщо ви не плануєте використовувати ноутбук протягом тривалого часу, заряджайте
акумулятор до 50% перед зберіганням.
Повідомлення користувача:
Чи можна використовувати ноутбук під час зарядки?
Повідомлення помічника:
Так, ви можете використовувати ноутбук під час зарядки. Сучасні ноутбуки розроблені таким
чином, щоб можна було безпечно користуватися ними під час зарядки. Однак, для оптимального
догляду за акумулятором, намагайтеся не залишати ноутбук підключеним до зарядного пристрою
постійно.
```

У цьому прикладі системне повідомлення надає загальні інструкції для помічника, а повідомлення користувача містять конкретні запити. Помічник реагує на ці запити, надаючи детальні та корисні відповіді, що відповідають контексту запиту.

Доповнена пошукова генерація (Retrieval Augmented Generation, RAG)

RAG це техніка, при якій модель використовує зовнішню базу знань або документи для отримання інформації (Retrieval) і генерує відповідь на основі цього контексту. Модель не обмежується лише своїми навчальними даними, вона отримує доступ до зовнішніх джерел інформації (наприклад, бази даних або інтернет-ресурсу) для більш точної та актуальної відповіді. Перед генерацією тексту модель здійснює пошук релевантної інформації з зовнішніх джерел даних (рис.10).

RAG забезпечує колаборацію між традиційними великими мовними моделями та зовнішніми базами даних або джерелами інформації. Коли мовна модель зустрічає запит або тему, на яку вона недостатньо навчена, за допомогою технології RAG шукаються кілька подібних до запитання користувача фрагментів тексту із векторної бази даних, після чого вони додаються до вхідних даних мовної моделі. Ця інформація додається у процес формування відповіді. Отримані відповіді вже ґрунтуються не лише на попередніх знаннях моделі, а й збагачені останніми, найбільш актуальними зовнішніми даними (рис.11).

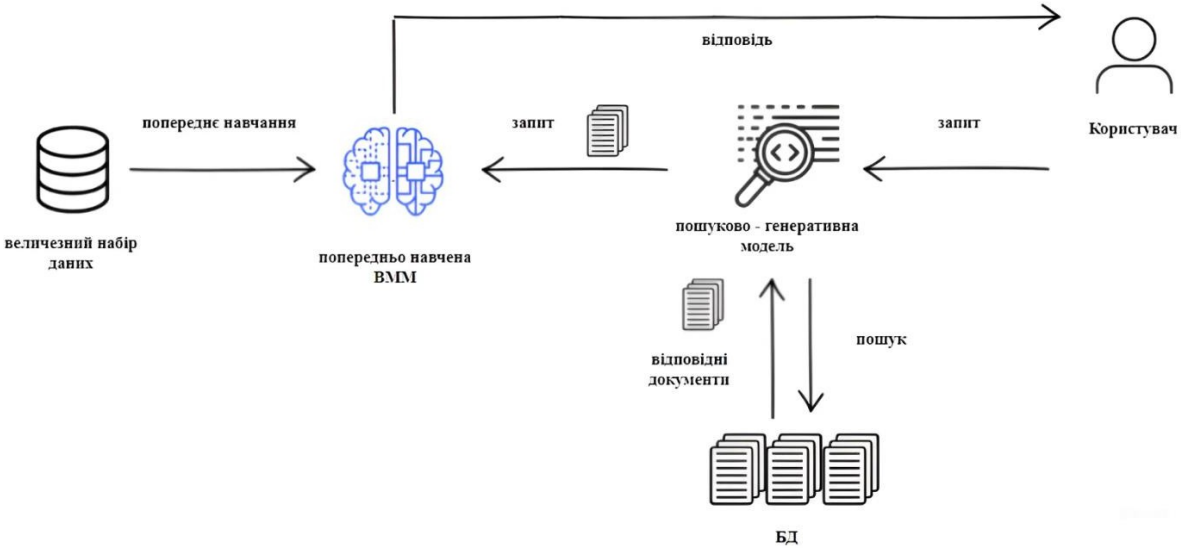


Рис.11. Діаграма послідовності технології RAG

RAG демонструє високу ефективність на кількох тестових наборах даних та генерує відповіді, що є більш фактологічними, конкретними та різноманітними. Останнім часом підходи на основі витягування інформації стали більш популярними і поєднуються з великими мовними моделями, такими як ChatGPT, для підвищення їхніх можливостей та логічної послідовності. Техніка RAG дозволяє моделям динамічно доступатися і інтегрувати зовнішню інформацію, розширюючи їхню базу знань в реальному часі [1, 2].

Техніка RAG підходить для завдань, пов'язаних із запитами фактологічних даних, де потрібна точна інформація, наприклад, у пошукових системах, питаннях по базі документів чи наукових статтях.

6. Результати дослідження

Результатом дослідження стало виявлення особливостей та порівняння поширених технік промптингу, що допомагає краще зрозуміти їх сильні та слабкі сторони, а також надати оптимальні сценарії їх застосування.

За технікою Zero-Shot Prompting запит до моделі формулюється без надання прикладів. Модель одразу намагається відповісти на основі своїх навчених даних, без додаткових контекстів або прикладів. Вона повинна розуміти завдання та генерувати відповідь лише на основі одного запиту. Підходить для простих запитів, таких як переклад тексту, узагальнення або відповіді на відомі питання. Є менш ефективною для складних чи нових завдань, які потребують контексту.

За технікою Feedback-Based Prompting відбувається ітераційний процес, в якому користувач надає зворотний зв'язок для уточнення запитів і відповідей моделі. Техніка заснована на постійній взаємодії та коригуванні запитів, може адаптуватись до конкретних вимог користувача. Добре підходить для складних завдань, що потребують уточнення, наприклад, написання технічної документації, генерація креативних текстів, програмування та персоналізованих завдань.

За технікою Few-Shot Prompting модель отримує кілька прикладів правильного виконання завдання перед тим, як надати остаточну відповідь. Приклади демонструють моделі структуру відповіді та дозволяють їй краще зрозуміти контекст і краще орієнтуватися у завданні. Підходить для складних завдань, таких як аналіз даних, математичні розрахунки, написання художніх текстів або розв'язання задач з конкретними вимогами. Менш ефективною є для завдань, де немає явної потреби у прикладах.

За технікою Chain-of-Thought Prompting модель розтлумачує кроки чи ланцюжок думок, що необхідні для вирішення завдання. Це допомагає їй краще структурувати відповідь. Відповідь моделі фокусується на проміжних кроках міркування, а не лише на кінцевому результаті. Це підвищує якість відповідей для багатоетапних рішень. Ідеально підходить для завдань, що вимагають логічних міркувань, таких як математичні завдання або наукові пояснення. Ефективною є для генерації аргументів чи проведення аналізу.

За технікою Tree of Thoughts Prompting відбувається дослідження різних можливих гілок рішень. Модель розглядає кілька можливих шляхів, оцінює їх та надає оптимальні чи креативні рішення з допомогою вивчення альтернатив. Підходить для творчих та складних проблем, які потребують пошуку найкращих рішень, таких як генерація ідей, стратегічні планування чи розробка складних програмних рішень.

За технікою Instruction Tuning Prompting модель навчається на великому обсязі даних, в яких спеціально описано інструкції, що дозволяє їй краще дотримуватись вказівок у запитах. Може опрацьовувати широкий спектр завдань на основі наданих користувачем інструкцій. Ефективно для завдань, де важлива чіткість і суворе дотримання правил, наприклад програмування, виконання покрокових інструкцій або виконання шаблонних завдань.

Дані дослідження можна звести у порівняльну таблицю 1.

Таблиця 1.

Порівняльна таблиця поширених технік промптингу

Техніка	**Особливості**	**Переваги**	**Типи завдань**
Zero-Shot	Без прикладів, один запит	Простота, швидкість	Найпростіші завдання, переклади, короткі відповіді
Feedback-Based	Ітерації зі зворотним зв'язком	Точність, гнучкість	Складні та персоналізовані завдання
Few-Shot	Приклади у запиті	Найкраща точність	Комплексні завдання, аналіз даних
Chain-of-Thought	Логічний ланцюжок міркувань	Логіка, зрозумілість	Математика, аналіз, завдання з кроками
Tree of Thoughts	Дослідження різних рішень	Креативність, багато варіантів	Творчі завдання, стратегії
Instruction Tuning	Дотримання явних інструкцій	Чіткість, універсальність	Завдання з чіткими вказівками, програмування
Reaction & Act	Адаптація до змін	Реакція на контекст	Динамічні завдання, ігри, підтримка користувачів

Reaction & Act Prompting і Retrieval Augmented Generation (RAG) - це техніки, які використовуються для покращення якості та релевантності генерованих моделей відповідей моделі. Незважаючи на те, що вони працюють по-різному, обидві техніки пов'язані тим, що скеровані на взаємодію з актуальною інформацією, покращення контексту та керування генерацією відповідей на основі зовнішніх даних.

Reaction & Act Prompting

За технікою Reaction & Act Prompting модель реагує на зміни контексту або нових даних, діючи на основі попередніх дій, що допомагає краще керувати процесом у динамічних завданнях. Техніка заснована на реакціях і відповідях на дані, що вводяться в реальному часі. Може коригувати свої дії з появою нових даних або змін контексту. Добре підходить для інтерактивних сценаріїв, таких як підтримка користувачів, чат-боти, ігри, сценарії динамічних даних. У ReAct використовуються дві компоненти:

- Reaction – модель реагує на зовнішні сигнали чи запити, змінюючи свою поведінку залежно від контексту. Наприклад, модель може коригувати свої відповіді, спираючись на контекст попередньої взаємодії.
- Act – модель виконує конкретні дії або функції на основі вказівок користувача, наприклад, запускає пошук, активує інструменти або запитує додаткові дані для покращення відповіді.

Retrieval Augmented Generation

За технікою Retrieval Augmented Generation модель використовує зовнішню базу знань або документи для отримання інформації (Retrieval) і генерує відповідь на основі цього контексту. Модель не обмежується лише власними даними, вона отримує доступ до зовнішніх джерел інформації

(наприклад, бази даних або інтернет-ресурсу) для більш точної та актуальної відповіді. У RAG використовуються дві компоненти:

- Retrieval – модель отримує релевантну інформацію із зовнішніх баз даних або документів, щоб доповнити чи уточнити свою відповідь. Це допомагає вирішити проблему обмежених знань моделі.
- Generation – після отримання релевантної інформації модель використовує її для генерації більш точних та змістовних відповідей. Це допоможе поліпшити як повноту, і актуальність відповідей з врахуванням поточної інформації.

Дані дослідження можна звести у порівняльну таблицю 2.

Таблиця 2.

Порівняльна таблиця технік Reaction & Act Prompting і Retrieval Augmented Generation

Характеристика	**Reaction & Act Prompting**	**Retrieval Augmented Generation**
Основна ідея	Реакція на зміни контексту в реальному часі	Генерація відповіді з використанням отриманої інформації
Механізм роботи	Ітеративний процес, де модель адаптує відповідь на основі нових вводів	Витягування релевантних даних із зовнішніх джерел для генерації точної відповіді
Контекст	Модель реагує на новий контекст після введення нових запитів від користувача	Модель використовує зовнішню базу даних для отримання додаткової інформації
Типи завдань	Динамічні, інтерактивні задачі, що вимагають багаторазової реакції	Питання-відповідь, завдання, які потребують точних даних чи фактів
Переваги	Гнучкість, здатність адаптуватись до змін	Точність, доступ до актуальної інформації
Приклад використання	Підтримка клієнтів, інтерактивні сценарії, ігри	Пошук по базі даних, питання відповіді з доступом до інтернет-ресурсів

Таким чином, обидві техніки допомагають моделям більш точно та динамічно відповідати на запити, використовуючи або отримувати додаткову інформацію, щоб зробити відповіді більш релевантними та інформативними.

Проведені дослідження та аналіз допомагають вибрати відповідну техніку промптингу залежно від типу завдання та потрібних результатів.

7. Висновки

Інженерія запитів є критично важливою галуззю для забезпечення ефективної взаємодії з великими мовними моделями. Належне формулювання промптів, або текстових запитів, що подаються на вхід мовної моделі, дозволяє максимізувати її потенціал, отримувати змістовні, очікувані та безпечні результати, а також мінімізувати ризики небажаної поведінки чи упередженості. Розробка ефективних запитів передбачає ретельне налаштування різних параметрів моделі, таких як температура, top-p, максимальна довжина, стоп-послідовності, штрафи за частоту та присутність, що дозволяє контролювати різноманітність, довжину та структуру згенерованих відповідей.

Ключовими розглянутими техніками промптингу є Zero-Shot, Feedback-Based, Few-Shot, Chain-of-Thought, Tree of Thoughts, Instruction Tuning. Окремої уваги заслуговують техніки Reaction & Act та Retrieval Augmented Generation. Техніка RAG інтегрує зовнішні джерела знань для розширення можливостей моделі, тоді як ReAct є комплексним підходом, що поєднує процеси мислення, планування дій та використання різноманітних джерел знань.

Інженерія запитів є ітеративним процесом, що потребує систематичного вдосконалення та експериментування з різними підходами та технічними рішеннями для досягнення оптимальних результатів у конкретних сценаріях використання ВММ. Належне опанування практики інженерії запитів забезпечує максимально ефективну та відповідальну взаємодію з великими мовними моделями, дозволяючи розкрити їхній величезний потенціал у різноманітних галузях застосування.

Список літератури

1.Prompt Engineering Guide, URL: https://www.promptingguide.ai, (Accessed: 13 September 2024).

2.Zhao, Wayne Xin, et al. (2023) "A survey of large language models." arXiv preprint arXiv:2303.18223 (2023). https://doi.org/10.48550/arXiv.2303.18223

3.Pranab Sahoo, et al (2024) A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications. arXiv:2402.07927. https://doi.org/10.48550/arXiv.2402.07927

4.OpenAI, URL: https://platform.openai.com/docs/introduction, (Accessed: 13 September 2024).

5.Google AI, URL: https://ai.google.dev/gemini-api/docs/model-tuning, (Accessed: 13 September 2024).

6.Anthropic, URL: https://docs.anthropic.com/claude/docs/intro-to-claude, (Accessed: 13 September 2024).

7.Matthew Renze, Erhan Guven (2024) The Effect of Sampling Temperature on Problem Solving in Large Language Models. arXiv:2402.05201. https://doi.org/10.48550/arXiv.2402.05201.

8.*Matthew Renze, Erhan Guven. The Effect of Sampling Temperature on Problem Solving in Large Language Models (2024). arXiv:2402.05201, https://doi.org/10.48550/arXiv.2402.05201*

9.Sander Schulhoff, Michael Ilie, Nishant Balepur et al The Prompt Report: A Systematic Survey of Prompting Techniques (2024) arXiv:2406.06608v1, https://doi.org/10.48550/arXiv.2406.06608.

PROMPTING TECHNIQUES FOR ENHANCING THE USE OF LARGE LANGUAGE MODELS

I.Yu.Yurchak, O.O. Kychuk, V.M. Oksentyuk, A.O. Khich

Lviv Polytechnic National University,
Department of "Computer Design Systems"
E-mail: Iryna.Y.Yurchak@lpnu.ua, Olha.Kychuk.knm.2020@lpnu.ua, Vira.M.Oksentyuk@lpnu.ua, Andrii.O.Khich@lpnu.ua

The work is dedicated to the study of fundamental prompting techniques to improve the efficiency of using large language models (LLMs). Significant attention is given to the issue of prompt engineering. Various techniques are examined in detail: zero-shot prompting, feedback prompting, few-shot prompting, chain-of-thought, tree of thoughts, and instruction tuning. Special emphasis is placed on Reaction & Act Prompting and Retrieval Augmented Generation (RAG) as critical factors in ensuring effective interaction with LLMs. The features of applying these techniques and their impact on results are highlighted. However, leveraging their full potential requires a careful approach and consideration of application specifics.

A review of the parameters of large language models, such as temperature, top P, maximum number of tokens, stop sequences, frequency and presence penalties, etc., is provided. It is noted that prompt development is an iterative process that involves sequential testing of different options to achieve optimal results. All techniques discussed in the study are supported by illustrative examples with obtained results. It is indicated which types of tasks each technique is more suitable for. The study results include comparisons of both fundamental techniques and more advanced technologies such as ReAct and RAG.

Prompt engineering is a key technology for the effective use of large language models. It is relevant due to the increasing application of artificial intelligence in all areas of human activity, and its role will only grow with the development of technology. The ability to correctly formulate prompts is becoming an important skill necessary for working with modern large models, especially given their versatility and complexity.

Keywords: large language models, prompt engineering, prompting technique, content generation.