

Практичне завдання №12: Автоматична генерація промптів (APE) та мета-промптинг

Мета: Зрозуміти логіку роботи алгоритмів Automatic Prompt Engineering (APE) через ручну симуляцію; навчитись використовувати LLM як «оптимізатора» для покращення власних промптів (Meta-Prompting).

Ключові концепції: Automatic Prompt Engineering (APE), Generation step, Evaluation step, Meta-Prompting, Iterative Optimization.

Завдання 1: Ручна симуляція APE (сленг -> офіційна мова)

Ідея: Відтворити логіку роботи алгоритму APE (генерація кандидатів -> оцінка -> вибір найкращого) без написання коду.

Завдання: Будемо вирішувати задачу перекладу молодіжного сленгу на суху бюрократичну мову. Ваша задача — знайти ідеальну інструкцію для цього, використовуючи сам ШІ.

Етап 1: Підготовка даних (Dataset)

Використайте наступні 5 пар прикладів (Input -> Output) як "навчальну вибірку" (або напишіть власні приклади):

1. «Зашквар» -> «Ситуація, що викликає сором або осуд»
2. «Крінж» -> «Відчуття незручності за дії іншої особи»
3. «На ізі» -> «Без докладання значних зусиль»
4. «Вайб» -> «Емоційна атмосфера або настрої»
5. «Лівнути» -> «Покинути приміщення або вийти з чату»

Етап 2: Генерація кандидатів (Generation Step)

Зверніться до LLM (ChatGPT/Gemini тощо) з мета-промптом, щоб вона згенерувала варіанти інструкцій.

Промпт (приклад):

Я хочу, щоб ШІ перекладав сучасний сленг на суху, офіційно-ділову мову.

Ось приклади того, що я хочу отримати:

[Вставити 5 прикладів з Етапу 1]

Напиши мені 5 абсолютно різних варіантів системної інструкції (System Prompt), які можна дати моделі, щоб вона виконувала це завдання ідеально.

Варіанти мають відрізнятися за стилем (наприклад, одна – рольова, інша – через приклади, третя – через суворий опис правил).

Зафіксуйте 5 отриманих інструкцій-кандидатів.

Етап 3: Оцінка (Evaluation Step)

Тепер протестуйте кожну з 5 інструкцій на нових даних (тестова вибірка).

Нові слова: «Токсик», «Пруфи», «Чілити».

Запустіть кожну інструкцію з цими словами. Оцініть результати за шкалою 1-5 (точність + стиль бюрократа).

Оцініть:

1. Яка інструкція впоралася найкраще?
2. Чи виявилася найкращою та, яку ви інтуїтивно вважали найсильнішою?

Завдання 2: Експеримент з мета-промптингом («промпт-оптимізатор»)

Ідея: Використати техніку мета-промптингу, де одна сесія LLM виступає експертом-консультантом для виправлення помилок іншої.

Завдання:

Крок А: Створення «поганого» промпта.

Візьміть відому математичну задачу, де моделі часто помиляються (наприклад, задача на логіку або уважність).

Приклад: «У кімнаті 3 коти, 2 качки і 1 курка. Скільки ніг у кімнаті, якщо зайшов господар з собакою?» (Модель може забути порахувати ноги господаря або меблів, якщо не уточнити).

Напишіть максимально простий, «лінивий» промпт: «Порахуй ноги.» і отримайте (ймовірно, неправильну або неповну) відповідь.

Крок Б: Створення «оптимізатора».

Відкрийте новий чат. Це буде ваш «експерт». Введіть наступний системний промпт:

Ти – експерт з промпт-інжинірингу та архітектури LLM. Твоє завдання – аналізувати невдалі промпти та результати їх виконання.

Ти маєш:

1. Проаналізувати, чому модель помилилася (не зрозуміла контекст, галюцинувала, була неуважна).
2. Запропонувати покращену версію промпта, використовуючи техніки CoT (Chain of Thought) та чіткі делімітери.
3. Пояснити, що саме ти змінив і чому.

Крок В: Ітерації.

3. Надішліть «оптимізатору»: «Мій промпт: *[ваш промпт]*. Відповідь моделі: *[помилкова відповідь]*. Виправ це. »
4. Візьміть запропонований «оптимізатором» промпт і протестуйте його в першому чаті.
5. Якщо результат все ще не ідеальний, поверніться до «оптимізатора» з новим результатом: «Стало краще, але *[опис проблеми]*. Спробуй ще раз. »

Оцініть:

1. Як змінилася структура промпта після втручання «експерта»?
2. Які техніки він додав (ролі, кроки, обмеження)?

Завдання для СРС: Вступ до DSPy

Ідея: Ознайомитися з концепцією фреймворку DSPy (Stanford NLP), який змінює парадигму роботи з LLM від «підбору слів» до «компіляції логіки». Зрозуміти різницю між декларативним описом задачі та її виконанням.

Завдання:

1. **Дослідження:** Перегляньте офіційну документацію DSPy (на GitHub або ReadTheDocs) або знайдіть якісний відео-огляд на YouTube (ключові слова: "DSPy explained", "DSPy vs LangChain").

2. **Аналіз ключових понять:** На основі знайденої інформації, дайте письмові відповіді на наступні питання (своїми словами):
- Signatures (**Сигнатури**): В DSPy ми описуємо *ЩО* ми хочемо отримати, а не *ЯК*. Наприклад, InputField: question -> OutputField: answer. Як це відрізняється від звичайного написання текстового промпта?
 - Modules (**Модулі**): Це блоки, які виконують завдання (наприклад, ChainOfThought або Predict). Як модуль використовує сигнатуру?
 - Teleprompters (**Оптимізатори/Компілятори**): Це найцікавіша частина. Як DSPy може автоматично підбирати приклади (few-shot examples) для вашого промпта замість вас?
3. **Порівняльна таблиця:** Запишіть 3 ключові переваги підходу DSPy над звичайним ручним промптингом (підказка: зверніть увагу на стабільність при зміні моделі, автоматичну оптимізацію та систематичність).
4. (**Бонус/за бажанням**): Якщо ви володієте Python, спробуйте запустити найпростіший скрипт DSPy у Google Colab, який реалізує просту QA-систему, і додайте скріншот результату. Майбутнє промпт-інжинірингу — це не «заклинання» словами, а інженерна розробка модульних систем.