Спринт 1

Тема: AI-ассистенты

В рамках данного спринта ведется отработка следующих навыков:

- сформировать базовые представления об инженерии промптинга и обеспечить практическое знакомство с AI-ассистентами для программирования на Python.
- работа с большими языковыми моделями через библиотеку LangChain для решения задач извлечения структурированной информации из неструктурированных текстов в контексте разработки прикладных решений машинного обучения.

Вводная практическая работа

1. Установите GigaCode
2. Установите Ollama + какую-либо модель LLM для ассистента.
3. Установите Continue и сконфигурируйте для ML.
4. Сделайте review кода с практической работы прошлого семестра

В публичном репозитории Github (или Gitverse) создайте аккаунт и залейте в него результат review кода.

Все практические работы раскладывайте папкам:00, 01, 02,….

Обновите main, создайте ветку, кодируйте и коммитьте по практическим работам, создайте Pull Request, разрешите конфликты, слейте и удалите ветку.

Ссылку на репозиторий (ваше портфолио) необходимо один раз опубликовать в СДО, ответив на вопрос типа «Анкета» в разделе «Практики».

Практическая работа № 1

Используя библиотеку LangChain и модель GigaChat, разработайте систему автоматического извлечения общего количества проживающих из текстовых заявок на аренду жилья.

Данные:

- rental_applications_text.csv с полями
    - text_id – уникальный идентификатор
    - text – текст заявки
- rental_applications_amount.csv с полями
    - amount – размеченное общее количество проживающих
    - text_id – уникальный идентификатор
    - text – текст заявки

Основная задача (обязательно всем):

1. Для каждой строки из rental_applications_text.csv извлечь значение total_persons (число проживающих) и сравнить с разметкой из rental_applications_amount.csv.

Дополнительный вариант (выберите только один):

- извлечение count_adults (число взрослых) и count_children (число детей)
- извлечение stay_period (период проживания):
    - start_date – дата заезда (или самая ранняя дата заезда при «плавающих» диапазонах)
    - nights – число дней проживания (или максимальное число дней проживания при «плавающих» диапазонах)
- извлечение price_per_day – жалаемая цена в сутки (при диапазоне брать максимум)

Этапы выполнения:

1. Определите структуру вывода: используйте либо
    - llm.with_structured_output(schema) с JSON-схемой,
    - либо JsonOutputParser/PydanticOutputParser с Pydantic-моделью,
      – так, чтобы в ответе был только JSON с полем total_persons (и полями из дополнительного варианта, если выбран).
2. Создайте ChatPromptTemplate с двумя сообщениями:
    - System: описание задачи («Извлекай общее количество проживающих…»)
    - Human: подстановка {text} и инструкция вернуть JSON по схеме.

3. Напишите скрипт (или Jupyter Notebook), который:
   a. Загружает rental_applications_text.csv и rental_applications_amount.csv.
   b. Для каждой заявки вызывает цепочку и получает JSON-ответ.
   c. Извлекает total_persons (и дополнительные поля, если выбран вариант).
   d. Сравнивает с истинными amount и вычисляет точность (accuracy).
   e. Сохраняет результат в results.csv с колонками:
      - text_id
      - predicted_total_persons
      - true_total_persons
      - correct (True/False)
      - и дополнительных полей при необходимости.
4. Для дополнительных полей проверьте решение на ручной разметке для 10-15 заявок, оцените точность.
5. Подготовьте отчёт (Jupyter Notebook):
   - Описание подхода и приведённый ChatPromptTemplate
   - Пример JSON-ответов для 5-10 заявок
   - Значение точности (accuracy) на полном датасете и на выборке из 10–15 заявок.

Структура проекта:

- rental_applications_text.csv
- rental_applications_amount.csv
- solution.ipynb
- results.csv