Этот документ описывает реализацию умного помощника по комплексному подбору инвестиционных площадок.

Скрипт помощника (общий бот): https://github.com/Sebastina14593/lct/tree/main/Общий бот/main.py.

Ссылка на контейнер докера: https://hub.docker.com/repository/docker/iroman1402/chatbot/general

Публичный IP адрес сервиса: 51.250.115.217

Используемые модели решения и причины использования

LLM-модель GigaChat и RAG-модель

RAG-модель (Retrieval-Augmented Generation) используется для улучшения качества и точности ответов, объединяя возможности генеративных моделей и систем поиска. GigaChat с RAG-моделью может сначала извлекать релевантную информацию из внешних источников данных или базы знаний (в данном случае это данные по «Помещениям и сооружениям», «Региональным мерам поддержки в Москве», а также заранее подготовленные вопросы по «Особым экономическим зонам»), а затем использовать эту информацию для генерации ответов. Векторы документов сохраняются в индексе, что позволяет быстро находить релевантные документы при получении запроса. Это позволяет GigaChat не только полагаться на собственные знания, но и динамически обновлять контекст и содержание ответов, основываясь на актуальных данных, что значительно повышает эффективность и полезность системы в задаче подбора помещений и мер поддержки для инвесторов.

Реализация данного подхода представлена, например, в скрипте бота по поиску помещений: https://github.com/Sebastina14593/lct/tree/main/Бот по поиску помещений/app.py.

В скрипте используются такие библиотеки как langchain и gigachat.

Алгоритм поиска парето-оптимального решения для поиска наиболее подходящих вариантов помещений

Поиск Парето-оптимального решения является мощным инструментом при выборе наилучших вариантов помещений для инвесторов, учитывая разнообразные и зачастую противоречивые критерии. В нашем проекте данный метод позволяет находить наиболее подходящие варианты, которые удовлетворяют запросам инвестора по разным критериям, не жертвуя одним параметром ради другого. Это особенно важно для пользователей с индивидуальными предпочтениями и потребностями, так как каждый вариант в Парето-множестве представляет сбалансированное решение по ключевым параметрам: цена, расположение, площадь, инфраструктура и другие важные характеристики.

Реализация данного подхода представлена в следующем скрипте: https://github.com/Sebastina14593/lct/tree/main/Бот по поиску помещений/pareto_optimum.py.

Используемые методы обработки данных

API Яндекс.Карт

При общении с пользователем умный помощник задает ему вопрос о районе, в котором бы инвестор хотело приобрести помещение/земельный участок. Для последующей реализации алгоритма поиска Парето-оптимума данные, предоставленные в его ответе, обрабатываются с помощью API Яндекс Карт и хранятся у нас в виде точки с координатами широты и долготы. В дальнейшем по данным точкам будут строиться расстояния до ближайшего помещения/земельного участка.

Функции, необходимые для подключения API Яндекс карт:

from geopy.geocoders import Yandex # для вычисления координат

from geopy.distance import geodesic # для расчета расстояния между точками

Файл «Помещения и сооружения» (https://github.com/Sebastina14593/lct/tree/main/Бот по поиску помещений/Помещения и сооружения.xlsx)

В данном файле представлены данные о помещениях и сооружениях, на основе которых находится наилучшее помещения для пользователя.

Поскольку в исходном файле отсутствовали координаты по некоторым помещениям/земельным участкам, поле с координатами было дозаполнено с помощью API Яндекс карт на основе поля «Адрес объекта».

Для генерации последующего pdf-файла с оптимальными вариантами было обработано поле «Фотографии объекта», в котором представлены ссылки на фотографии объектов.

В поле «Стоимость объекта, руб. (покупки или месячной аренды)» были объединены с полями «Стоимость, руб./год за га», предназначенное для земельных участков, а также «Стоимость, руб./год за кв.м.».

Для последующей реализации RAG-модели для различных сценариев умного помощника данные переводятся в эмбединги и хранятся в векторной БД. Для этого используются следующие функции:

from langchain_huggingface import HuggingFaceEmbeddings

from langchain_community.vectorstores import FAISS

from langchain_community.document_loaders.csv_loader import CSVLoader # для перевода документа.

Условия и ограничения внутри решения

Основным ограничением для решения выступает переключение бота с одного функционала («Подбор меры поддержки для бизнеса») на другой («Подбор инвестиционной площадки») и наоборот, а также нарушение сценария его поведения при «специфическом» ответе пользователя (например, вопросы, касающиеся отвлеченных тем). Для обхода данных ограничений было принято решение использовать интерактивные кнопки для сохранения сценария поведения умного помощника.

Техническая документация API

Эндпоинт: «/ask»: Получение ответов на вопросы, задаваемых умному помощнику, относительно особых экономических зон.

Пример запроса:

POST /search

{

'question': 'Привет! Какие меры поддержки в ОЭЗ в Москве существуют?'

}

Пример ответа:

{

"answer": "Здравствуйте! В ОЭЗ Технополис Москва действует ряд мер поддержки для резидентов, например Льготы по налогу на прибыль и имущество организаций. Рассказать ли поподробнее?"

}