

Построение системы интеллектуального поиска по portalу СберСпасибо с применением технологий глубокого обучения

✉ Павел Кравченя

Содержание работы. В работе рассматривается построение системы, выполняющей поиск наиболее подходящих ресурсов portalа СберСпасибо по запросу пользователей. Поиск осуществляется по смыслу текста с использованием эмбедингов, полученных с использованием нейронной сети. В работе использовалась предобученная мультилингвистическая сеть на основе дистиллированной трансформерной модели `multilingual-MiniLM-L12`.

Архитектурно решение представляет собой несколько сервисов, запущенных в системе `Kubernetes` на `Yandex Cloud` (см. рисунок 1), позволяющих решать три взаимосвязанные задачи. Первая задача состоит в получении самих данных – информации о ресурсах portalа СберСпасибо с помощью парсинга (посредством разработанного сервиса `parsing-service`), расчете эмбедингов по их тестовому описанию и размещении полученных векторов в базе данных. В качестве последней используется векторная база данных `Weaviate`, имеющая модульную структуру. Один из ее модулей позволяет прозрачно для пользователя осуществлять получение и обработку эмбедингов. Это приводит к упрощению архитектуры системы за счет делегирования работы с эмбедингами самой базе.

Парсинг осуществляется в автоматическом режиме по нескольким разделам сайта СберСпасибо с использованием библиотеки `Selenium`. В процессе парсинга сервис считывает описание конкретного партнера или продукта. Если описание отсутствует, сервис пытается перейти по ссылке, ведущей на веб-сайт партнера, и, в случае успеха, получает информацию с его главной страницы. Парсинг в системе также запускается в `Kubernetes` по расписанию через равные промежутки времени с помощью инструмента `Apache Airflow`.


Вторая задача заключается в получении HTTP-запроса с поисковой фразой от пользователя, получение ее эмбединг-вектора, осуществление поиска в базе данных ресурсов с близкими к пользовательскому значениям эмбедингов и возврат результатов. Взаимодействие с пользователем осуществляется с помощью разработанного сервиса `search-service`, который формирует запрос к базе с нужными требованиями, дожидается результатов и возвращает ответ. Формирование пользовательских эмбедингов и выполнение поиска осуществляется базой `Weaviate` с вышерассмотренным модулем.

Третья задача состоит в оценке качества поиска. Качество может быть посчитано на основе статистики пользовательских кликов на ссылках, предложенных системой в качестве результатов поиска. Для сбора и агрегации этой статистики был разработан сервис `metrics-service`. Данный сервис получает информацию о действиях пользователя и сохраняет ее базу данных `MongoDB`, развернутую на одноименном сервисе `Yandex Cloud`. По запросу сервис возвращает рассчитанные метрики CTR, АНС (которые, при необходимости, могут быть дополнены другими метриками).

Для удобства взаимодействия пользователя с рассмотренными сервисами был реализован пользовательский интерфейс **frontend-service**. Он предоставляет клиенту веб-страницу, на которой можно выполнить поисковый запрос и просмотреть полученные результаты. Скрипты веб-страницы также отправляют всю информацию о запросах, ответах и кликах пользователя на **metrics-service**.

Тестирование показало работоспособность всей системы. Результаты поиска для большинства поисковых запросов, по здравому смыслу, соответствуют запросам. В процессе работы пользователей с системой рассчитываются метрики и обновляется информация о них.

Основным недостатком работы является ее плохая горизонтальная масштабируемость, что выяснилось уже при проведении нагрузочного тестирования. Проведенное с применением инструмента **wrk** тестирование показало, что в лучшем случае сервис обеспечивает задержку в 0.7 - 1.2 сек и выдерживает 12 - 17 запросов в секунду. При этом различные попытки автора увеличить число реплик сервисов так и не привели к увеличению пропускной способности. Возможно, причина этого явления заключается в нестабильной работе базы данных Weaviate, которая в настоящий момент находится в фазе активной разработки.

Репозиторий проекта:  [kpdphys/MLOps/tree/main/SberProject](https://github.com/kpdphys/MLOps/tree/main/SberProject)

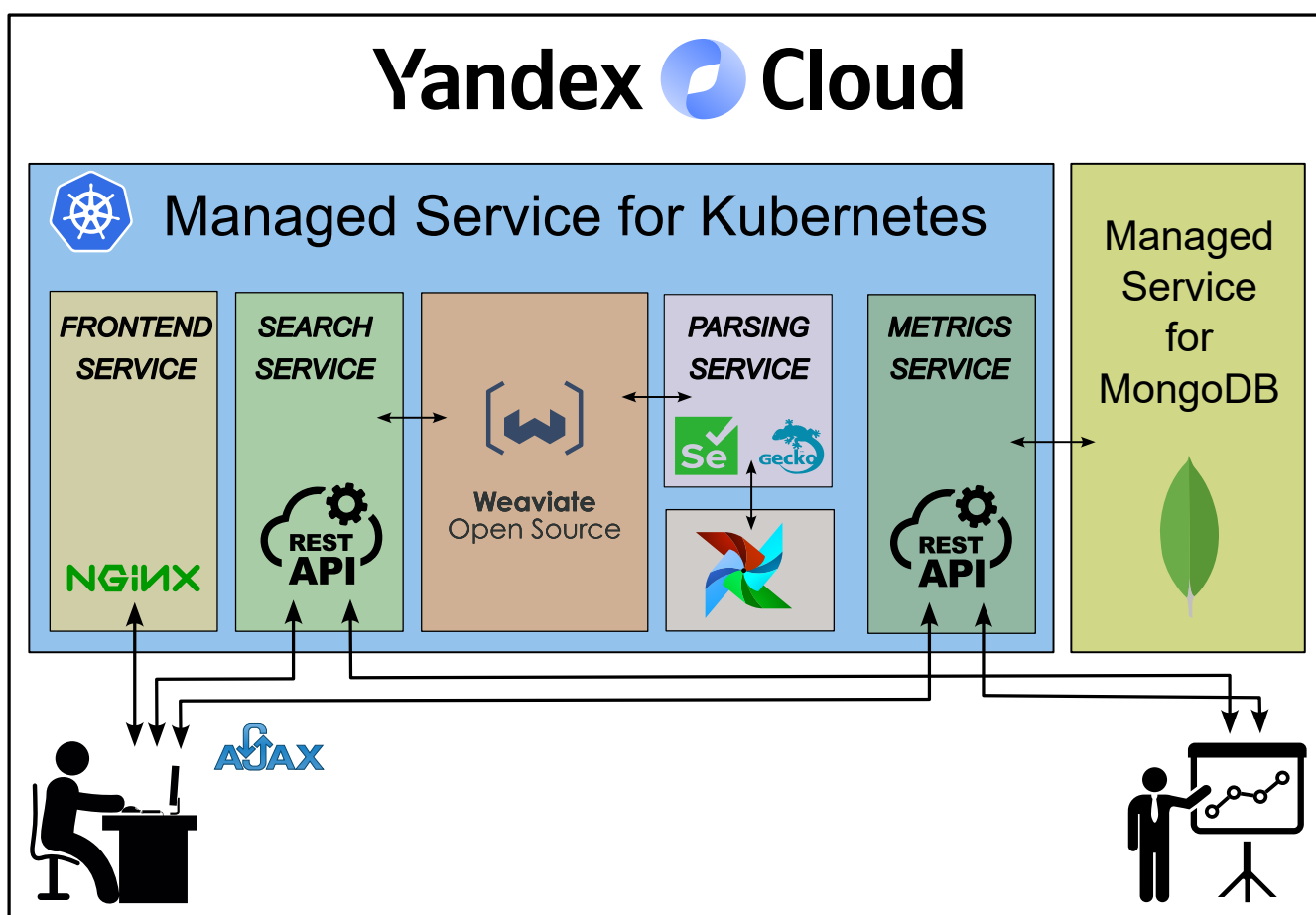


Рис. 1: Архитектура системы