Семантический поиск фраз в текстах

Команда BIDVA, хакатон, апрель-май 2025

Участники команды

01

Лукин Иван

Тимлид, Backend-разработчик

02

Макунин Борис

Data-инженер

03

Пападык Всеволод

Web-разработчик

04

Полянин Алексей

Аналитик данных

05

Белогай Дарья

Аналитик данных

Цель и задачи проекта

Цель проекта — реализовать систему семантического поиска, которая может понимать смысл вопросов и находить наиболее релевантные ответы в базе текстов.

1. **Сравнение**

   Найти и внедрить рабочую ML-модель для смыслового сравнения текстов;

2. **Запуск**

   Подготовить инфраструктуру для локального запуска;

3. **Взаимодействие**

   Создать веб-интерфейс для взаимодействия с пользователем.

Этапы реализации

1. Анализ задачи и данных:

- Изучили условия задачи;
- Проанализировали предоставленные датасеты.

С результатами проведенного анализа можно ознакомиться по ссылке.

Этапы реализации

2. Выбор модели:

- Для решения поставленной задачи, была выбрана нейросетевая модель BERT;
- На Hugging Face отобрали 3 популярных до обученных модели NLP с поддержкой русского языка:
  - intfloat/multilingual-e5-base,
  - Alibaba-NLP/gte-multilingual-base,
  - deepvk/USER-bge-m3.

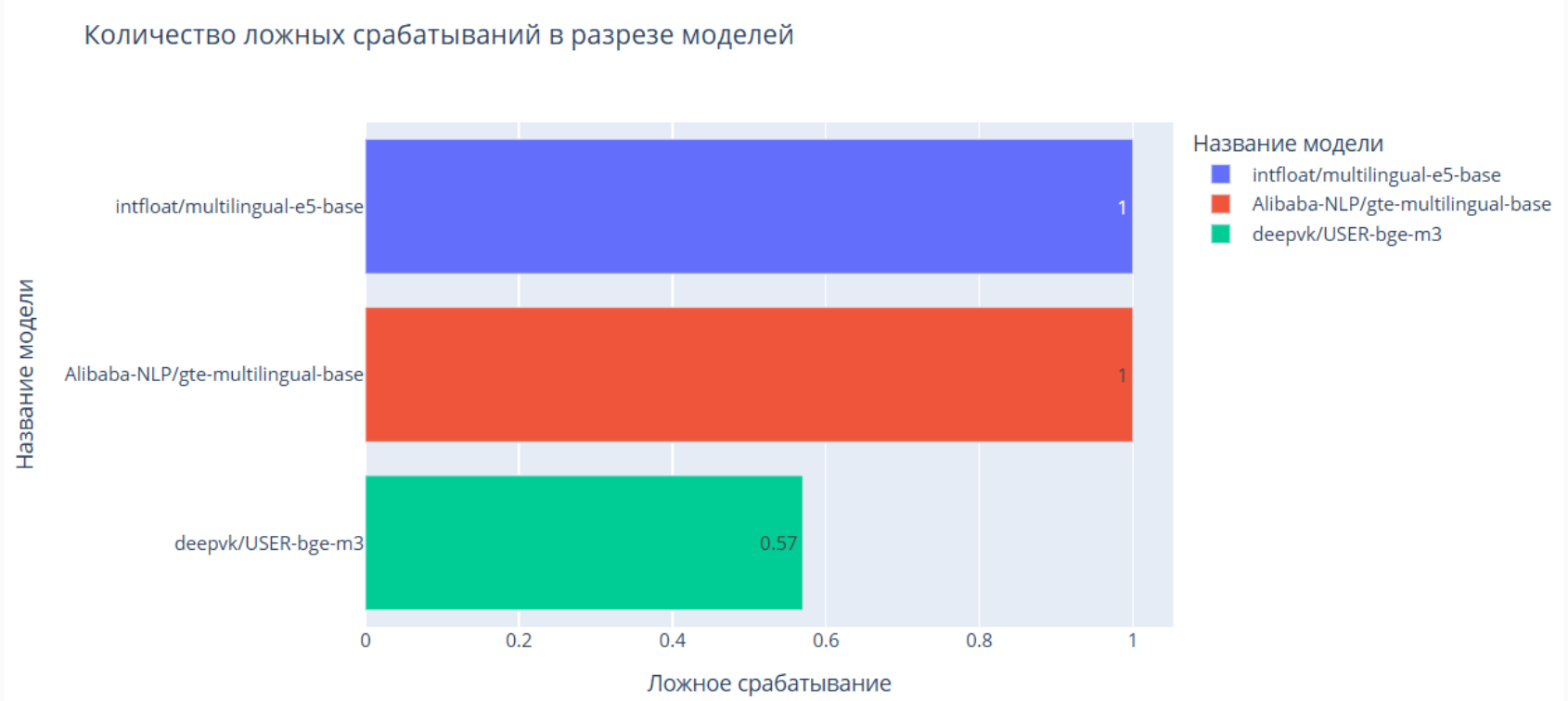
Этапы реализации

3. Тестирование моделей:

- Составили тестовый датасет из 100 примеров;
- Оценивали долю правильных ответов и ложных срабатываний;

С датасетом можно ознакомиться по ссылке.

С результатами тестирования можно ознакомиться по ссылке.


Количество ложных срабатываний в разрезе моделей
Название модели
intfloat/multilingual-e5-base
Alibaba-NLP/gte-multilingual-base
deepvk/USER-bge-m3
intfloat/multilingual-e5-base
Alibaba-NLP/gte-multilingual-base
deepvk/USER-bge-m3
1
1
0.57
0
0.2
0.4
0.6
0.8
1
Ложное срабатывание

Этапы реализации

4. Интеграция модели:

- Создали backend сервер (FastAPI + Uvicorn);
- Упаковали всё в Docker-контейнер.

Все файлы Backend можно найти по ссылке

Этапы реализации

5. Интерфейс пользователя:

- Реализовали web-страницу для взаимодействия с пользователями (HTML-страница на React Framework);
- Разместили страницу на web-сервере Vite;
- Упаковали решение в Docker-контэйнер.

Все файлы Frontend можно посмотреть по ссылке.

Этапы реализации

6. Подготовка всего проекта:

- Для автоматического развёртывания всех контейнеров был создан файл docker-compose;
- При запуске контейнеров открыть web-страницу можно по адресу http://localhost:5173.

Итоговый проект можно посмотреть по ссылке.
Весь проект можно посмотреть в GitHub по ссылке.
Подробное описание решения и инструкции запуска описано в файле README.md.

Алгоритм работы поиска

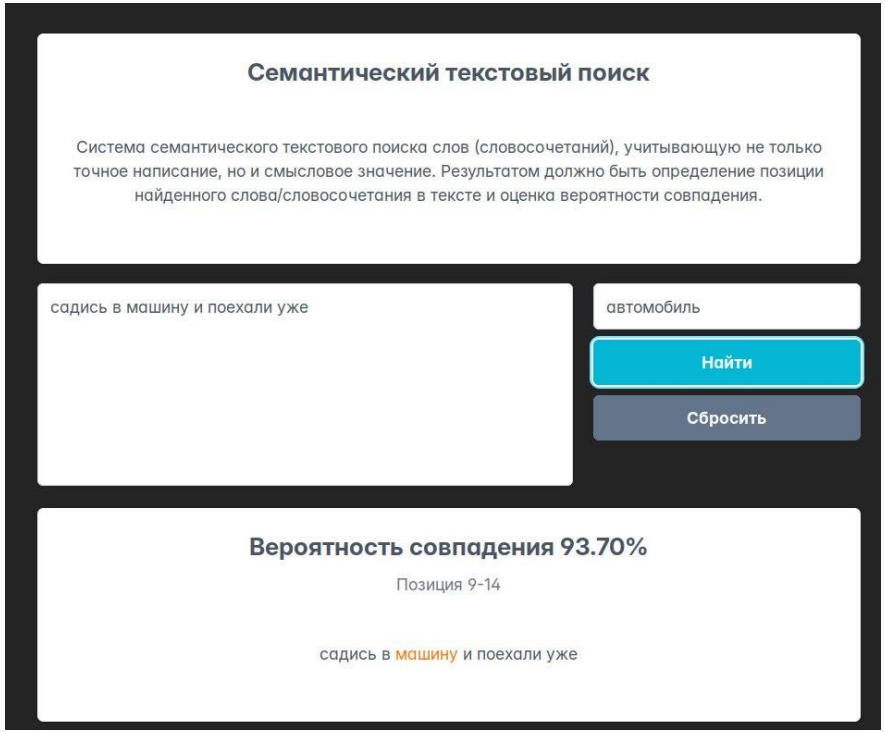
- На вход: предложение и поисковый запрос.
- Чистим предложение от пунктуации и следующих служебных частей речи ['а', 'в', 'но', 'и', 'не', 'из', 'под'] (самые распространённые)
- Разбиваем предложение на пары слов (словосочетания).
- Преобразуем каждый сегмент и запрос в эмбеддинги с помощью модели.
- Сравниваем сегменты с запросом по косинусному расстоянию.
- Если лучший сегмент >67% — сравниваем также отдельные слова.
- Возвращаем либо лучшее словосочетание, либо отдельное слово.
- Если всё <67% — отправляем нулевой результат.

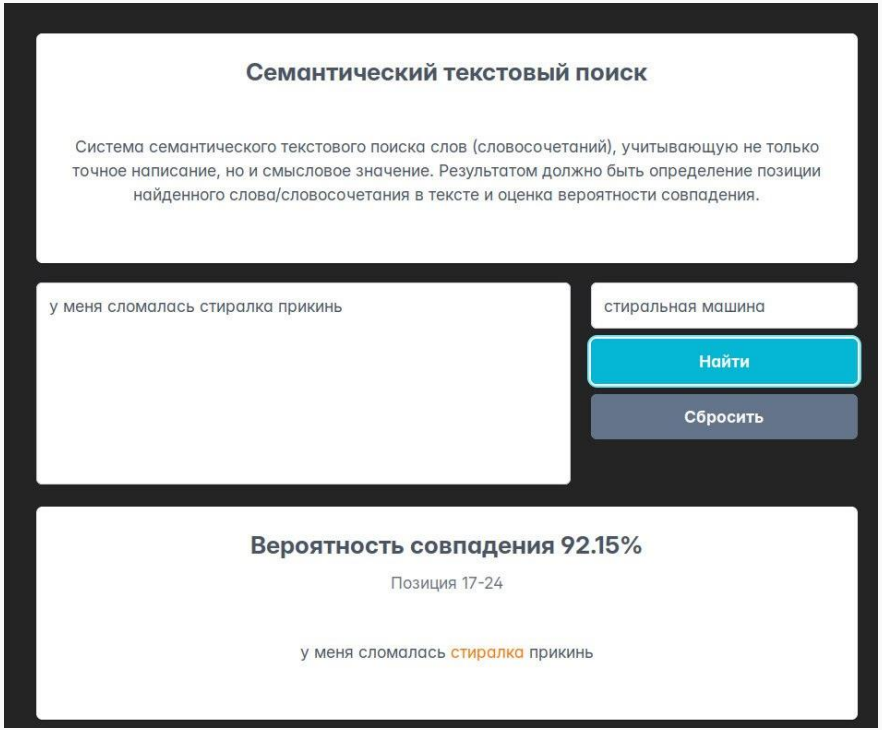
Архитектура сервиса

Сервис реализован по клиент-серверной архитектуре:

Frontend	Vite + HTML-страница на React Framework
Backend	FastAPI-приложение, обрабатывающее запросы и вызывающее модель
Инфраструктура	Всё упаковано в Docker container
Локальный запуск	Локальный запуск через развёртывание Docker compose.

Демонстрация работы





Планы по развитию

- Выполнить сравнительный анализ большего количества обученных моделей для выявления более точных моделей;
- Добавить возможность анализа нескольких текстов одновременно;
- Добавить возможность анализа текстов в файлах;
- Добавить дополнительные проверки на корректность входных данных.

Выводы

В рамках проделанной работы выполнен:

- Анализ текстов в предоставленных датасетах;
- Изучены основные концепции по работе с текстом для выполнения семантического поиска, в частности познакомились с моделью BERT;
- Реализовали Frontend и Backend решения;
- Упаковали клиент-серверное решение в Docker-compose.