ПРОЕКТ МЭРА МОСКВЫ

ДЕПАРТАМЕНТ ПРЕДПРИНИМАТЕЛЬСТВА И ИННОВАЦИОННОГО РАЗВИТИЯ ГОРОДА МОСКВЫ

РАЗВИТИЕ ЧЕЛОВЕЧЕСКОГО КАПИТАЛА

ЛИДЕРЫ ЦИФРОВОЙ ТРАНСФОРМАЦИИ

ХАКАТОН МЭРА МОСКВЫ

ЛИДЕРЫ ЦИФРОВОЙ ТРАНСФОРМАЦИИ 2025

ПРОЕКТ МЭРА МОСКВЫ

ДЕПАРТАМЕНТ ПРЕДПРИНИМАТЕЛЬСТВА И ИННОВАЦИОННОГО РАЗВИТИЯ ГОРОДА МОСКВЫ

РАЗВИТИЕ ЧЕЛОВЕЧЕСКОГО КАПИТАЛА

ЛИДЕРЫ ЦИФРОВОЙ ТРАНСФОРМАЦИИ

Сервис выделения сущностей из поискового запроса клиента в мобильном приложении торговой сети «Пятерочка»

Команда JoJo

Задача №10

КОМАНДА JoJo

О команде

- Saint-Denis, France
- 2 человека
- Капитан команды - Графф Николай

Наименование задачи

Сервис выделения сущностей из поискового запроса клиента в мобильном приложении торговой сети «Пятерочка»

Описание решения

- Модель RuBERT-tiny2
- Обучение с BIO-разметкой, дополнительные пост-обработки
- Поддержка опечаток, сокращений и неполных слов, характерных для реальных пользовательских запросов

Дальнейшее развитие

Адаптация модели под другие сценарии розничной торговли (чат-боты, рекомендации), расширение типов сущностей и внедрение онлайн-обучения на новых пользовательских данных.

Состав команды

Николай Графф

- ML-инженер
- @payrox
- +79516788868

Ангелина Графф

- DevOps специалист
- @angelina_graff
- +79500202868

Информация о команде

Краткая история команды:

- Семейная команда
- Первый хакатон
- Back-end разработчики
- Живем во Франции

Почему вы выбрали именно эту задачу из предложенных на хакатоне?

- Значимость: улучшение поиска может реально помочь миллионам пользователей
- Работа с NER: практическое применение задач анализа текста
- Интересный и полезный опыт: задача требовала не только обучить модель, но и создать и развернуть полноценный сервис

С какими основными сложностями или вызовами вы столкнулись и как их преодолели?

- Неоднородная и противоречивая разметка в train-датасете
- Много времени ушло на настройку пост-обработки
- Изменения в команде - один из участников выбыл по личным причинам

Задача и результат

- Цель: извлечение сущностей TYPE, BRAND, VOLUME, PERCENT из пользовательских запросов (BIO-разметка).
- Результат: модель + веб-сервис POST /api/predict, Docker-образ, документация и тех.отчёт.
- Фокус: устойчивость к опечаткам, неполным словам и числовым формàтам.

Технический стек

01

Python 3.10+

02

HuggingFace Transformers, PyTorch

03

FastAPI, Uvicorn

04

NumPy, Pydantic v2, Docker

05

Модель cointegrated/ rubert-tiny2

Архитектура решения (общая схема)

Поток :

1. Пользователь
2. FastAPI (main.py)
3. predict_word_bio()
4. токенизация
5. модель
6. пост-обработка
7. ответ

Сервер: Uvicorn workers (по CPU), модель загружается на старте.

Endpoint: POST /api/predict принимает {"input": "..."}, отдаёт список спанов с BIO.

Архитектура модели NER

- **Тип**: токен-классификация (IOB2/BIO), 9 меток: O, B/I-TYPE, B/I-BRAND, B/I-VOLUME, B/I-PERCENT.
- **Токенизация**: стандартная у rubert-tiny2.
- **Выход модели**: вероятности по меткам на токен; дальше восстановление в посимвольные/пословные спаны.

Данные и предобработка

- **Источник**: *train.csv* (разметка BIO на уровне символов).
- **Сплит**: случайный 90/10 с фиксированным seed=42.
- **Выравнивание**: сопоставление оффсетов сущностей и токенов; вне сущностей - O

Генерация синтетики

- **Цель**: устойчивость к опечаткам/латинице/вариантам объёмов и процентов.
- **Методы**: удаление/вставка/замена символа; сосед по раскладке; транспозиция соседних букв.

Обучение

- **База**: rubert-tiny2 (дообучение).
- **Гиперпараметры (итог)**: LR=3e-5, batch=16, эпох=12, weight_decay=0.01.
- **Практика**: подбирали параметры эмпирически (множественные запуски), выбирали по F1 на валидации; EarlyStopping(patience≈3).

Пост-обработка

- **Trim punctuation**: обрезка пунктуации на краях спана.
- **Numeric overrides**: регулярные выражения для чисел
- **Margin (per-class)**: понижение в O, если (top1–top2) < delta.
- **Агрегация по словам**: порог доли класса в слове 0.58; при неоднозначности можно унаследовать класс предыдущего слова.

Архитектура сервиса

- **Endpoints**:
  a. GET /healthz - статус и путь к модели;
  b. POST /api/predict - основной endpoint.
- **Производительность**: torch.set_num_threads(1); масштабирование - за счёт *uvicorn*.
- **Развёртывание**: Docker-образ с моделью.

API: формат I/O

- Вход: {"input": "молоко"}
- Выход:
  [{"start_index":0,"end_index":9,"entity":"B-TYPE"}, ...]
- Пустая строка возвращает []; индексы - срезы исходной строки.

Проблемы и решения

- **TYPE - BRAND на границе**: margin (per-class) + словная мажоритарка
- **Шум числовых форматов**: numeric overrides (регулярные выражения).
- **Опечатки/латиница**: решено синтетикой + словной агрегацией.
- **Компромисс скорость/стоимость**: отказ от GPU (дорогая аренда), выбор компактной модели rubert-tiny2.