

Отчет по проекту
Калинин Владислав Алексеевич
ШАД
Новосибирск

Здесь я распишу полную работу над моим проектом. Разделим описание на

1 Концепция

2 DL обучение и выбор модели

3 fortend

Уточню, что точных критериев по проекту не было и я реализовал много идей и оставил их в описании. В итоге на инференс ушло лучшее решение.

1 Концепция

За основу взята идея с определения типа научной статьи по arXiv. Моё приложение работает со статьями на 2 языках: русский, английский. На вход приложению дается название и аннотация статьи. Модель рассчитывает вероятности класса статей и выдает список и график вероятностей для пользователей. Более того мое приложение может подобрать наилучшее название к статье по ее описанию. Если автор не придумал названия к своему тексту, то данная опция ему поможет.

Теперь перейдем к деталям реализации такого проекта. Стоит учесть, что я много экспериментировал с сервисом и оставил полное описание всего, что делал. Из основного опишу, что

1. Проводился парсинг данных

2. Проводилось обучение 3 моделей, отличающихся по размеру от DistillBERT до bloom

3. Приложение реализовано на фреймворках strimlit (основное приложение) и flask. Данный выбор сделан по причинам, что huggingface плохо дружит с flask. Полное приложение работает и там, но с меньшим интерфейсом для юзера. Более того на flask реализован сбор инференс метрик, что нельзя делать на strimlit.

2 DL обучение

2.1 Сбор данных

Основные данные взяты с Kaggle и преобразованы для обучения в файле data_preparation. Я провожу чистку данных: удаляю лишние столбцы,

преобразую текста, удаляя лишнее (пунктуацию, пробелы, непонятные токены) для обучения. Стоит отметить, что по канонам Kaggle там данные представлены грязно и сложно, поэтому пришлось написать несколько функций для полной чистки и извлечения таргета.

Поскольку в области науки существуют много разделов то таргет будет выглядеть в виде 'Область науки/ Раздел области'. Более того в данных arxiv области науки определены сокращено, например cs.gt-hp, что не совсем понятно пользователям, поэтому я заменяю сокращения классов на полное названия по сайту arxiv.

Данных для обучения мало не бывает поэтому я реализовал парсинг arxiv в extra_data_install. Я комбинирую библиотеки MyHTMLparser и BeautifulSoup. Класс MyHTMLparser пробегает по сайту arxiv и ищет определенные паттерны ссылок на статьи внутри сайта. Далее BeautifulSoup заходит на ссылки и ищет аннотацию и область статьи и качает их. Так я увеличу мои данные в несколько раз. Я взял список url ссылок с различных областей науки от конденсированных состояний до психологии. Все статьи взяты с 2023 года

2.2 Обучение и выбор модели определение типа статьи

Тут я решил поэкспериментировать и опишу все мои результаты и действия. Для задачи я использовал различные модели и подходы к обучению. Пойду описание работы от легкого к сложному

A) DL для девочек

Начнем с простого, возьмем модель DistillBERT и зафайтуним модель на наших данных. Не будем никак преобразовывать обучения, а просто возьмем Trainer, поменяем немного условия тренировки и прогоним его. Так я примерно понимаю время обучения и метрики. За метрику я взял ассигасу. После оценок понимаю, что модель обучается несколько часов и метрика равна 69. После я увеличивал эпохи обучение, но улучшений особых не было. Вся реализация в simple_train_model

B) DL для мальчиков

Тут я взял BERT и полностью реализовал процесс обучения и для эффективности я взял AdamW, потому что он более точно считает и работает с weight decay чем Adam. Более того я реализовал Mixed precision train для эффективности обучения. Обучение происходит заметно быстрее и скор вырос, что очень хорошо. Вся реализация в medium_train_model

C) DL для мужиков

Ну тут я решил уже поэкспериментировать с огромными моделями, в частности, я взял bloom (а чтобы скучно не было в жизни). При обычном

обучении у меня уже падает по памяти на forward. Я проводил много экспериментов по обучению данной модели и пришел к решению. Во-первых, понизим размер батча до 2. Реализуем аналогично mixed precision train и используем параллелизм с помощью библиотеки tensor_parallel. При таких условиях можно зафайтчить данную модель, но обучения одной эпохи шло 2 дня. Это был хороший опыт для меня по работе с большими моделями, но у меня не хватило ресурсов для полного файтчинга модели экспериментов с ней. Более того чтобы засунуть ее для инференса на сайт надо будет квантизовать модель, а квантизация работает только на сру, что займет неприлично много времени. Конечно, можно сделать еще дистилляцию, но из-за размеров модели это займет очень много времени.

В итоге основная модель на сервис была BERT, как самый оптимальный выбор по времени и железу, которое мне доступно.

2.3 Выбор моделей на генерацию название статей и работы с языками.

Как упоминалось выше в моем сервисе можно работать с 2 языками, а также генерировать название к тексту. Для данных были выбраны готовые pretrained модели с hugging face: Helsinki-NLP/opus-mt-ru-en и Callidior/bert2bert-base-arxiv-titlegen.

2.4 Efficient DL

Для инференса я применил квантизацию BERT в файле. Трансформеры сами по себе огромные и я решил ужать модель, реализовав метод квантизации. В частности, проводиться дополнительный прогон модели для подбора параметров для квантизации и лучшего свертывания модели. Я так же хотел попробовать дистилляцию, но поскольку квантизация выполняется на сру, то она заняла у меня 38 часов и времени оставалось мало уже для нового. Так же реализован переход в общий вид записи модели ONNX. В идеале было бы преобразовать далее код на C++, что требует больше времени на изучения.

3 Реализация самого сайта

Задача реализована на Flask и strimlit. Такой выбор пал, потому что huggingface space полностью не работает с Flask и для полного развертывания проекта я реализовал все на strimlit, но есть отдельно готовый проект и на Flask. Отличие в реализациях на данных проектах практически не отличается. Я сделал упор на красоту и интерфейс в strimlit с отрисовками графиков и большим взаимодействием с юзером. А на Flask я упростил взаимодействие с юзером изменил дизайн и добавил метрики prometheus для отслеживания работы сервиса (strimlit не дружит с prometheus и реализовать на нем подобное сложнее)

Разберем немного по отдельности проекты.

3.1 Strimlit

Strimlit проще Flask и он является основным, потому что именно он используется в продакшене в hugging face. Сервис разбит на несколько страниц во вкладке pages. Основная страница приложения app.py. Для нее я реализовал шапку профиля, которую задизайнил сам. Позже юзера встречает выбор языка. Если он выберет русский язык я дополнительно подгружаю трансформер для перевода и работы с текстом на русском, иначе не делаю этого, что экономит время запросов и память.

Выглядит это так

После выбора языка появиться две вкладки для ввода названия статьи и аннотации к ней. Юзер вводит нужные данные и нажимает на кнопку после выводиться список данных и график. Причем если юзер выберет язык русский и напишет все на английском, то все будет тоже работать.

Happy ML 59 team

Home

Select a language/ default is English

Russian

Write the title of the article

Write an abstract of the article

Click

Click

The article is written in the field of science:

Math.Functional Analysis

Computer Science.Social and Information Networks

Condensed Matter.Strongly Correlated Electrons

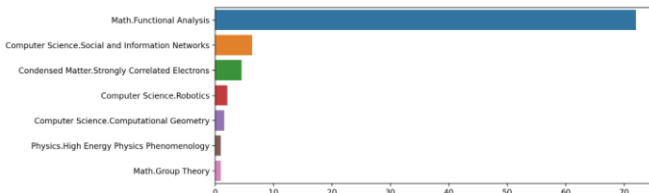
Computer Science.Robotics

Computer Science.Computational Geometry

Physics.High Energy Physics Phenomenology

Math.Group Theory

Probability graph that an article of a given field of science



Так же я поставил ограничение на выдачу значений в 5 классов. Такое решение взято, поскольку при вводе всякой чуши или статьи на различные темы. При таких вводах вероятности классов небольшие, а их много, что засоряет сайт и график.

Так же есть страница 2 где можно написать текст для генерации названия

Page for generating articles and titles

Write summary of your article or dont write anything

most outstanding challenge that confronts heliophysics. This paper argues the scientific case for novel out of ecliptic observations of the Sun's polar regions, in conjunction with existing, or future multi-vantage point heliospheric observatories. Such a mission concept can revolutionize the field of Heliophysics like no other mission concept has - with relevance that transcends spatial regimes from the solar interior to the heliosphere.

Generation title

modeling and quantifying the solar evolution of the sun's polar regions

И третья страница с милой гифкой

The application was created to help work with scientific articles

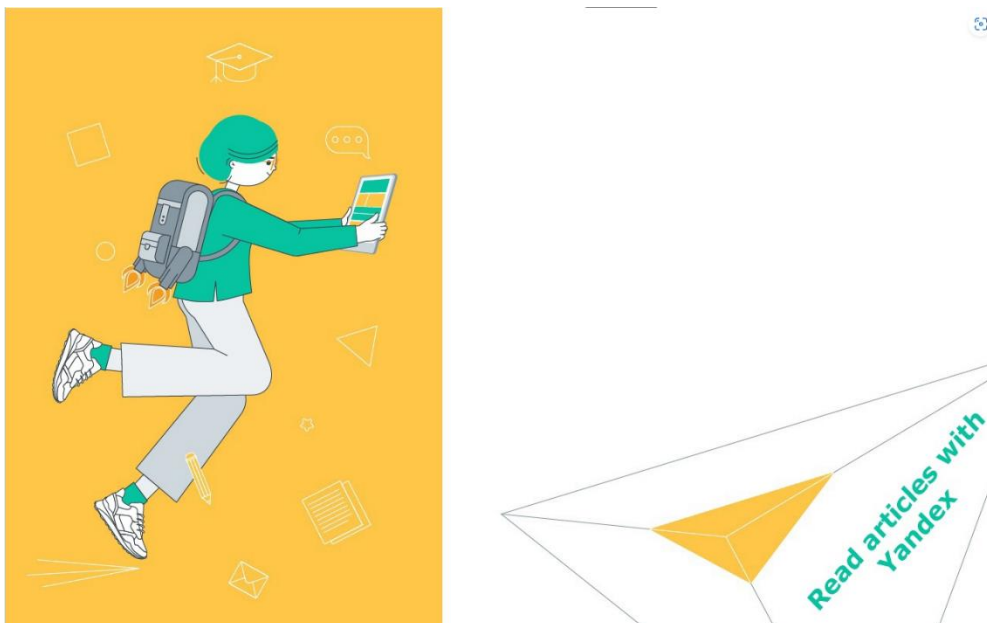
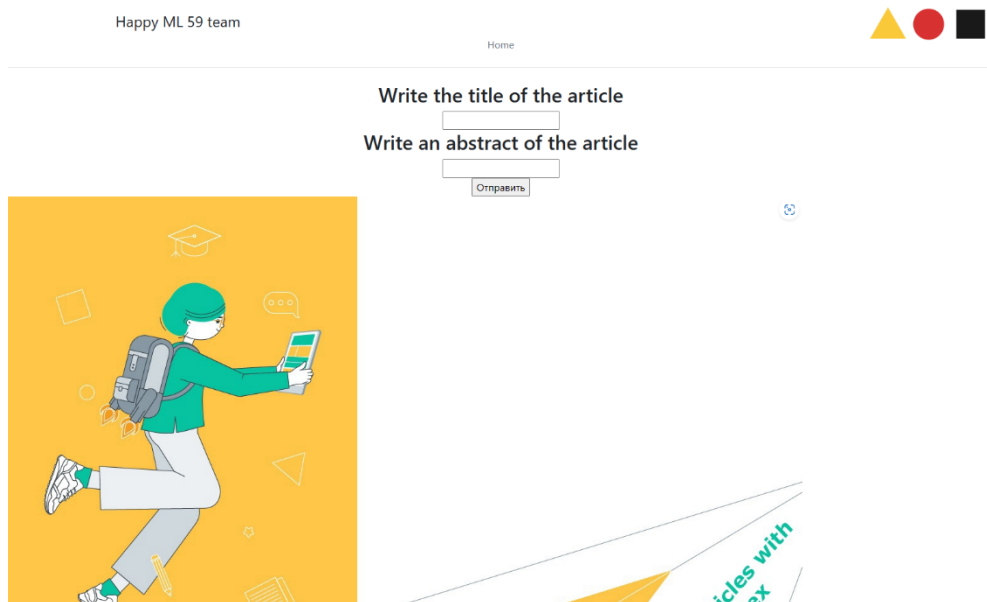
I hope the app helped you



The work was performed by Kalinin Vladislav

3.2 Flask

Подобный интерфейс реализован и во flask, но чуток проще. Поскольку основная работа шла в strimlit, flask уже предполагается мною для более серьезного выката сервиса. Там я реализовал внутренний сайт с интерфейсом ШАДа но чуть другим. Я решил поэкспериментировать с дизайном. Тут юзер сразу заходит и видит 2 вкладки текста и работа идет сразу на 2 языках. Выглядит это примерно так



Юзер вводит данные статьи и его бросает на страницу с ответом данных.

Topic of the article

Quantitative Finance.Portfolio Management

- [Home](#)

Так же он может пройти по ссылке генерации и ему подберут название статьи по его абстракту, и пользователь увидит примерно такое

Title for the article

applications of the language technology to the study of modern chinese thoughts and literature

- [Home](#)

И так же добавлена страница about, которая желает удачи юзерам с милой картинкой.

3.3 Метрики

Куда нам без метрик и тут. Для этого я подключил prometheus и настроил подсчет метрик на нужный порт. То есть у меня есть отдельный prometheus сервис для сбора метрик. Поскольку prometheus только хранит мои метрики я реализовал папку grafana для визуализации метрик (all.yaml). В папке telegraf собирает данные и дает prometheus. Я собираю метрики каждые 5 секунд с моего сервиса (время указывал в prometheus yaml). Я выбрал базовые метрики: кол-во запросов в процессе обработки на сервере, кол-во запросов всего было, время по выдаче предиктов. Стоит уточнить, что данные метрики реализованы на приложении на Flask т. к. они с ним дружат вместе.