Министерство науки и высшего образования Российской Федерации
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ЯДЕРНЫЙ УНИВЕРСИТЕТ
«МИФИ»

ИНСТИТУТ ЛАЗЕРНЫХ И ПЛАЗМЕННЫХ ТЕХНОЛОГИЙ
КАФЕДРА №31 ПРИКЛАДНАЯ МАТЕМАТИКА

ОТЧЕТ

по научно-исследовательской работе за весенний семестр 2024 года

на тему:

Разработка метода аугментации запросов к языковым моделям с помощью векторных представлений текстов из базы данных

Автор: Закарян Арсен Арменович
Руководитель НИР: д.ф.-м.н. главный научный сотрудник НИЦ «Курчатовский институт» Сбоев Александр Георгиевич

г. Москва 2024

Содержание

1 Аннотация

Работа посвящена исследованию возможности использования векторных представлений (эмбеддингов) документов для аугментации запросов (в качестве подсказок) к языковым моделям обученным на задачи ответа на вопросы. Проблема языковых моделей, генерирующих ответы только на основе заданного промта (текстовой фразы), заключается в том, что они могут галлюцинировать и выдавать неверную информацию. Цель работы - разработать оптимальный метод векторизации документов и метод внедрения этих веторов в языковую модель с целью использования этих документов в процессе генерации ответов на вопросы. Акутальность данного исследования обусловлена плохой работоспосностью языковых моделей без дополнительных подсказок.

В текущем семестре был выполнен обзор литературы по языковым моделям и методам rerieval augmentated generation (RAG) . Для освоения методов обучения языковых моделей была дооубчена модель ruT5-base для автоматичесской суммаризация текста с целью обработки отзывов на лекрственные препараты. Результаты метрики ROUGE: Rouge1 = 6.56%, Rouge2 = 2.86%, RougeL = 6.51%, модель требует улучшения.

2 Введение

Построение языковых моделей остается актуальным и важным направлением в области искусственного интеллекта. С их помощью улучшается качество автоматического перевода, распознавания речи, генерации текста и других языковых задач.

Проблема языковых моделей, генерирующих ответы только на основе заданного промта (текстовой фразы), заключается в том, что они могут галлюцинировать и выдавать неверную информацию. Кроме того, длина входной последовательности у этих моделей ограничена, что затрудняет использование длинных контекстных подсказок. Существует несколько способов решения этой проблемы. В данной работе предлагается подход, основанный на обогащении запроса векторными представлениями документов.

Цель работы - разработать оптимальный метод векторизации документов и метод внедрения этих веторов в языковую модель с целью использования этих документов в процессе генерации ответов на вопросы.

В данном семестре был выполнен ряд задач:

1) Изучение общих методов обработки естественного языка(NLP). Этот этап включал в себя разбор рекурентных нейронных сетей для решения задач классификации, регрессии. Также были рассмотрены способы постороения векторных представлений для слов и предложений, что предоствило базу для дальнейшей работы.

2) Изучение механизма внимания (attention), который является ключевым элементом, позволяющим модели сосредоточиться на определенных частях входных данных при выполнении задач NLP. Этот механизм позволяет модели выделить наиболее важные элементы в тексте и использовать их для более точного принятия решений.

3) Постановка задачи языкового моделирования и изучение моделей решения этой задачи. Языковые модели используются для прогнозирования вероятности последовательности слов в тексте. Они обучаются на больших корпусах текстов и способны предсказывать следующее слово в предложении на основе контекста. Языковые модели играют важную роль в различных задачах NLP, таких как машинный перевод, суммаризация текста и другие, помогая моделям понимать и генерировать естественный язык более точно и эффективно.

4) Разработка программы на языке Python для автоматической суммаризации текста в контексте анализа отзывов о лекарственных препаратах.

5) Изчение идеи RAG (Retrieval Augmented Generation) для более качественного ре-

шения задачи суммаризации отзывов

3 Обзор современного состояния области и существующих методов

3.1 Механизм attention

При работе с естественным языком, каждому слову в словаре ставится в соответствие вектор с вещественными компонентами (эмбединг слова). Эти векторы поступают на вход нейронной сети с той или иной архитектурой. Компоненты векторов являются параметрами, которые подбираются в процессе обучения так, чтобы близкие по смыслу слова имели схожие векторы. В качестве меры близости обычно выступает косинус угла между векторами, который определяется их скалярным произведением.

Одна из проблем, возникающая при построении эмбедингов слов, - это неоднозначность естественного языка (table - это стол или таблица и т.д.). При обучении на корпусе текстов, вектор эмбединга каждого слова получает фиксированные компоненты, которые не зависят от того смысла, в котором это слово встретилось в тексте.

Механизм внимания (точнее в этом случае самовнимания) модифицирует вектор эмбединга каждого слова, "подмешивая"к нему векторы его окружения (контекста) с некоторыми весами. Пусть, например, в предложении "The table has a lot of data"словам соответствуют векторы $v_1, ..., v_7$. Для модификации вектора $v_{table} = v2$, вычислим его скалярные произведения с другими словами:

$$w_1, w_2, ..., w_7 = v_2v_1, v_2v_2, ..., v_2v_7$$

(включая его же). Полученные веса отнормируем при помощи функции $softmax$, так чтобы их сумма равнялась единице, а значения лежали в диапазоне [0...1]: $w_i = softmax(w1, ..., w7) = \frac{e^{w_i}}{(e^{w_1}+...+e^{w_7})}$. Чем ближе к единице веса w_i, тем сильнее слово v_2 похоже на слово v_i. Построим теперь новый эмбединг слова table в виде взвешенной суммы: $v2 = w_1v_1 + w_2v_2 + ... + w_7v_7$. Так как артикли и предлоги в векторном пространстве находятся далеко от векторов table и data, мы получим что-то типа:

$$v_{table} \approx 0.9v_{table} + 0.1v_{data}$$

Для предложения "There is a plate on the table"

$$v_{table} \approx 0.8v_{table} + 0.2v_{plate}$$

. Для текста с другим контекстом (второй пример выше) получатся иные компоненты. В первом случае "внимание"слова table фокусируется на слове data, а во втором - на слове plate. В результате единый для всех смыслов вектор v_table расщепляется на два вектора, которые имеют тенденцию перемещаться к своим смысловым кластерам (row, digits,... в первом случае и chair, furniture, ... - во втором).

3.2 Языковые модели: эволюция и архитектура. BERT, GPT и T5

Языковые модели — это алгоритмы, способные продолжать тексты. Последней и наиболее успешной с точки зрения качества оказалась архитектура трансформеров. Она состоит из двух частей: encoder (на изображении слева) и decoder (на изображении справа).

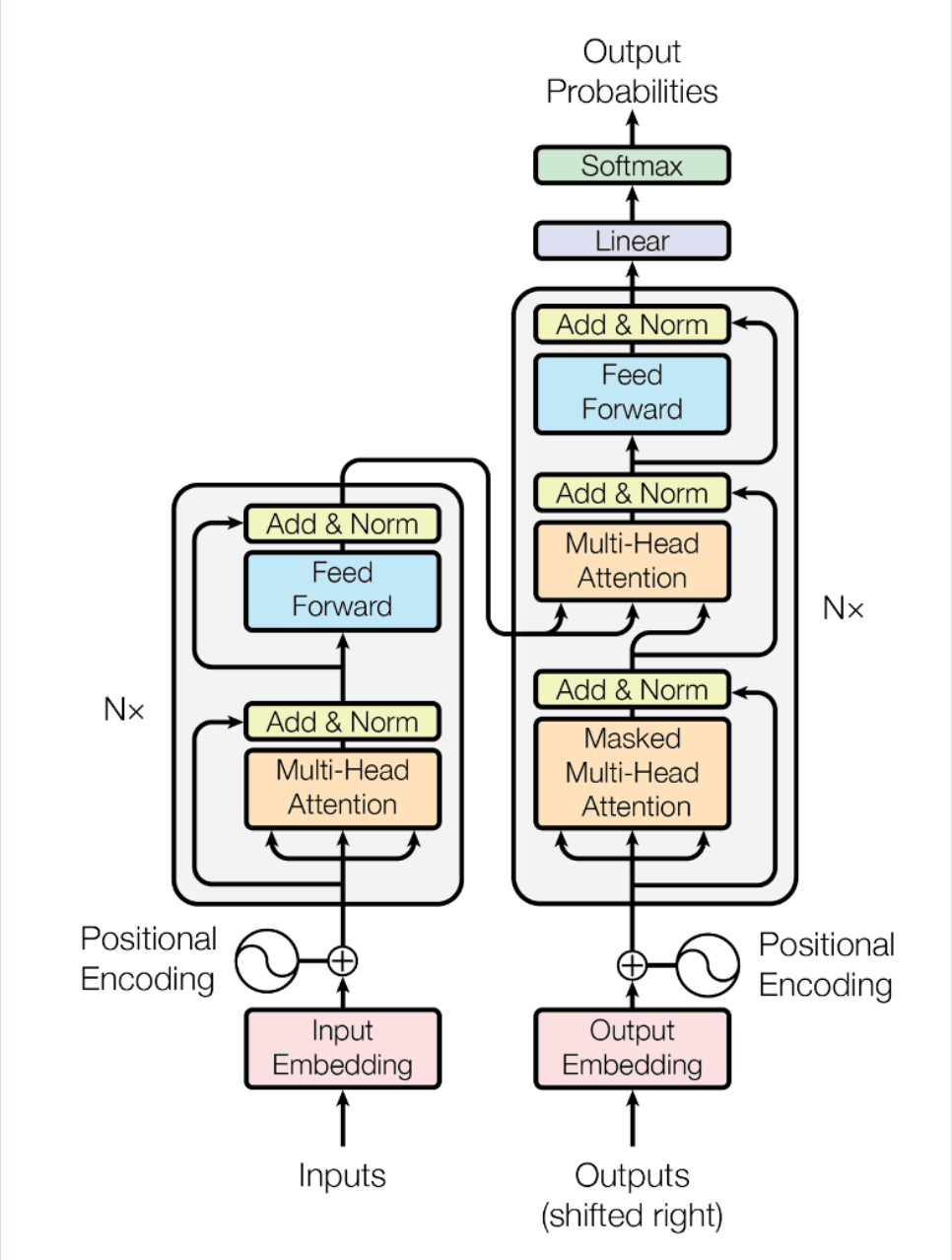

Рис. 1: Архитектура трансформера

Изначально был популярен подход обучать части отдельно. Так на базе encoder-блоков были построены BERT-модели. Идея обучения звучит несложно: давайте из

входного текста замаскируем токеном MASK 15% имеющихся токенов и обучим модель угадывать, какие именно токены были скрыты. Тогда, если модель обучится это делать, она сможет очень хорошо понимать текст.

Таким образом, энкодеры обладают следующими особенностями:

Анализируют входной текст и связи между токенами. Выделяют важные токены для определённой задачи. Ничего не генерируют.

На базе декодеров сделаны GPT-модели. Они обучаются предсказывать следующий токен на основе предыдущих. На инференсе, когда очередной токен сгенерирован, он добавляется в контекст, и уже на основе него выбирается новый токен. Таким образом модель:

генерирует токен за токеном. смотрит на весь контекст, архитектурно, нет забывания токенов. имеет возможность (как и BERT-модели) обучаться параллельно. обладает достаточно высокой вычислительной стоимостью инференса.

Контекст в случае трансформеров определяется числом токенов, которые они могут обработать за раз. Архитектурно за понимание контекста отвечает блок Attention, и размеры матриц в нём как раз определяют размер контекста.

Размер матриц конечен: чем они больше, тем сложнее вычислять блок внимания, поэтому контекст существенно ограничен.

Одним из нейросетевых алгоритмов, позволяющих проводить суммаризацию текста является T5 (Text-To-Text Transfer Transformer) - это модель на основе трансформеров, которая представляет собой универсальный фреймворк для обработки естественного языка. T5 отличается от других моделей трансформеров тем, что она решает широкий спектр задач NLP, преобразуя их в единый формат "текст-к-тексту".

Архитектура T5 состоит из энкодера и декодера, которые обучаются вместе на больших объемах текстовых данных. Энкодер преобразует входной текст в скрытое представление, а декодер генерирует целевой текст или ответ. Обучение T5 происходит в два этапа: предобучение на большом корпусе текста и дообучение на конкретных задачах.

3.3 RAG: идея в обработке естественного языка для лучшей генерации

Одним из методов улучшения модели являяется RAG (Retrieval Augmented Generation) - это метод, который использует большие языковые модели, в котором пользователь задает вопросы, и дополнительная информация из внешних источников программно добавляется к запросу перед подачей его на вход языковой модели. Это позволяет языковой модели предоставить более полный и точный ответ на вопрос пользователя.

В простом примере, если пользователь спрашивает модель: "Каков текущий обменный курс доллара?"языковая модель сама по себе может не иметь актуальных данных о курсах валют. Добавляя информацию из первого результата поиска Google по запросу "курс доллара к рублю"к запросу пользователя, языковая модель может сгенерировать более точный ответ.

В более сложном сценарии можно рассмотреть создание автоматизированной системы технической поддержки на основе языковой модели. Вместо повторного обучения модели на изменяющейся базе знаний вопросов и ответов, RAG может динамически извлекать соответствующую информацию из базы знаний на основе запроса пользователя для генерации точных ответов.

Концепция Retrieval Augmented Generation можно разделить на три основных компонента:

- Retrieval: Поиск и извлечение соответствующей информации. Компонент, отвечающий за поиск и извлечение информации, называется ретривером. - Retrieval Augmented: Усиление запроса пользователя с помощью полученной соответствующей информации. - Retrieval Augmented Generation: Генерация ответа пользователю, включающего полученную дополнительную информацию.

При реализации RAG возникают несколько сложностей, таких как алгоритмы нечеткого поиска, определение оптимального размера текстовых блоков для подачи на вход языковой модели, обработка нескольких и больших статей в базе знаний и эффективное объединение или сжатие полученной информации.

Эти основные проблемы должны быть решены при разработке системы RAG. Хотя есть стандартный подход к созданию систем RAG, часто требуется тонкая настройка и настройка, чтобы достичь желаемых результатов.

Основной алгоритм для системы RAG включает сегментацию базы знаний на

более мелкие текстовые блоки, которые затем используются для усиления запросов пользователей и генерации ответов. Точная настройка и усовершенствование этой начальной структуры являются необходимыми для индивидуального подхода системы RAG к конкретным требованиям.

Именно с этой технологией и будет поставлена задача на следующий семестр: Реализовать языковую модель с использованием RAG, в которой дополнительной информацией будут являться не целиком документы, а только их эмбеддинги

4 Реализация программы автоматической суммаризации

Постановка задачи: есть модель нейронной сети с архитектурой T5, принимающая $T = w_i$ - текст состоящий из слов w_i, где i от 1 до N. Необходимо обучить веса модели так, чтобы ее ответ $t = w'_j$ - текст состоящий из слов w'_j, где j от 1 до M отражал главную мысль исходного текста.

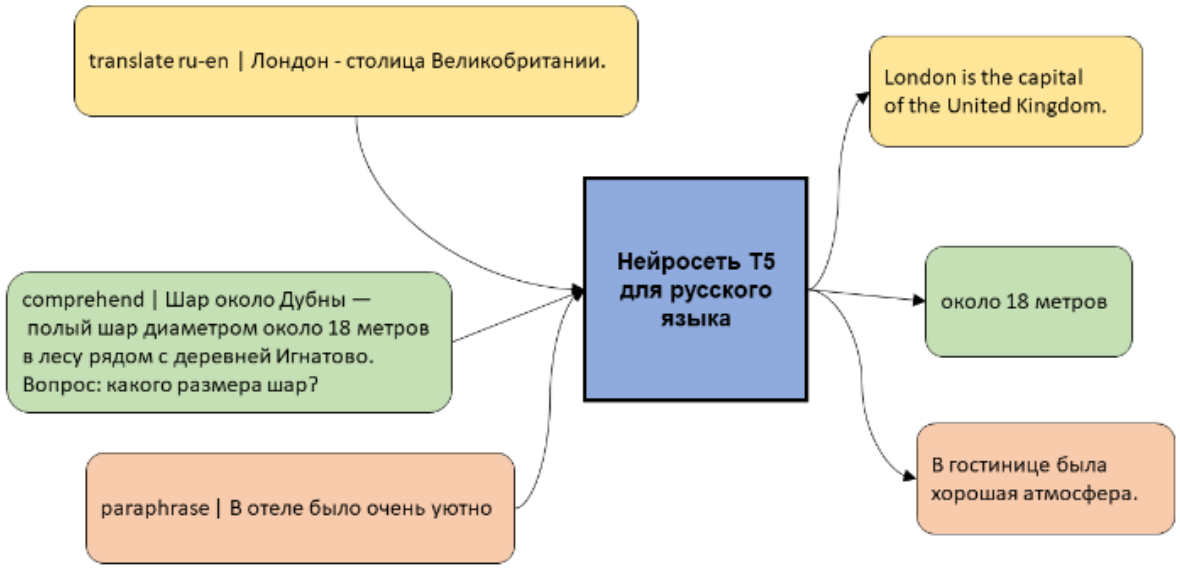

Рис. 2: T5

Для экспериментов используется корпус RDRS `https://github.com/sag111/RDRS`. Корпус содержит 3800 отзывов про мед. препараты

Для решения задачи была взята конкретная модель ruT5-base из библиотеки transformers.

Решать задачу будем в 2 этапа:

1) Подготовим данные к виду, в котором можно будет подавать на вход модели.

2) Дообучим модель ruT5-base для решения нашей конкретной задачи

Изначально отзыв имеет следующий вид:

url

TITLE

заголовок

TEXT

текст отзыва

В конце текста всегда (или почти) есть предложение: "Общее впечатление: ..."

RATING

Вот пример отзыва, который предстоит обработать:

"http://otzovik.com/review_2500038.html \nTITLE \nОтзыв: Препарат Имунорикс (детям) - не советую никому\nTEXT\nМоя девятилетняя дочь болела часто, но не тяжело. Решили воспользоваться новыми достижениями в медицине и по совету врача пропили в два этапа это средство. Итог ужасный: переболели всеми видами ОРВИ, но самое страшное за полгода два раза обструктивным бронхитом и теперь нам почти подтвердили бронхиальную астму. Берегите своих детей! Год выпуска/покупки: 1970
Общее впечатление : не советую никому\nRATING\n1\nNOTE\n"

Нам необходимо избавиться от всех лишних деталей и оставить только сам отзыв и заголовок В результате выполнения этого куска кода имеем следующий комментарий

```
for i in range(len(review)):
    #delete general_impression from review
    main_review[i] = review[i]
    general_impression_idx = main_review[i].find('Общее впечатление')
    if general_impression_idx != -1:
        main_review[i] = main_review[i][:general_impression_idx]

    #delete everything before review
    review_idx = main_review[i].find('Отзыв: ')
    if review_idx != -1:
        main_review[i] = main_review[i][review_idx + len('Отзыв: '):]

    #target
    target_idx = main_review[i].find('\n')
    target[i] = main_review[i][:target_idx]

    #delete "TEXT" and "\n" if they exist
    TEXT_idx = main_review[i].rfind('\n')
    if TEXT_idx != -1:
        main_review[i] = main_review[i][TEXT_idx + len('\n'):]
```

Рис. 3: Препроцессинг

"Моя девятилетняя дочь болела часто, но не тяжело. Решили воспользоваться новыми достижениями в медицине и по совету врача пропили в два этапа это средство. Итог ужасный: переболели всеми видами ОРВИ, но самое страшное за полгода два раза

обструктивным бронхитом и теперь нам почти подтвердили бронхиальную астму. Берегите своих детей! Год выпуска/покупки: 1970 "

а так же заголовок на него

'Препарат Имунорикс (детям) - не советую никому'

После этого разделяем нашу выборку на train и test и разделим наш исходный текст на токены с помощью этой функции

```
def preprocess_function(examples):
    type(examples)
    model_inputs = tokenizer(examples["text"], max_length=max_intput_length,
                             truncation=True, padding=True)

    labels = tokenizer(text_target=examples["summary"], max_length=max_target_length,
                       truncation=True)

    model_inputs["labels"] = labels["input_ids"]
    return model_inputs
```

Рис. 4: Токенизция данных

Если в исходной строке вдруг появится символ, который не лежит в нашем словаре, мы заменяем его на специалный токен <pad>

После всех преобразований мы наконец-то может дать данные на обучение модели

```
training_args = transformers.Seq2SeqTrainingArguments(
        output_dir="./results",
        evaluation_strategy="epoch",
        learning_rate=2e-5,
        per_device_train_batch_size=1,
        per_device_eval_batch_size=1,
        weight_decay=0.01,
        save_total_limit=3,
        num_train_epochs=6,
    )
```

```
trainer = transformers.Seq2SeqTrainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_data["train"],
    eval_dataset=tokenized_data["test"],
    tokenizer=tokenizer,
    data_collator=data_collator,
)
```

```
trainer.train()
```

Обучение модели

В результате обучения ошибка и на тренирововчной и на тестовой выборках значительно упала.

Epoch	Training Loss	Validation Loss
1	1.964900	1.861135
2	2.070300	1.793932
3	1.906800	1.731060
4	1.820700	1.707372
5	1.783700	1.701762
6	1.694800	1.699989

Ошибка обучающей модели

Рассмотрим пример отзыва вместе с правильным ответам, а так же ответ нашей модели

Отзыв:

Давно страдаю запорами, не знаю причину. Обычно не обращала на это особого внимания, пока не появился геморрой, который стал следствием нарушения регулярности стула. Сделали операцию и предупредили, что стул должен быть регулярным иначе будут другие проблемы. За неимением желания тратить много денег на слабительные решила попробовать уладить свою проблему при помощи Бисакодила. Стоит он чуть больше 20 рублей, что несомненно порадовало, но возникло сомнение в его эффективности. Выпив одну таблетку на ночь, я не ощутила никакого эффекта вечером, но утром обалдела, он реально действенный. Однако днем следующего дня побаливал живот терпимо конечно, но неприятно. После 2 таблетки у меня было ощущение будто меня выворачивает изнутри, боли были сильные, но и эффект тоже через чур. Решила, что нужно поменять данное средство на более мягкое в действии.

Правильный ответ:

Слабительное Бисакодил-Хемофарм - Сильный, но...

Ответ модели:

Таблетки Биотики "Бисакодил Хороший слабительный препарат.

Осталось посмотреть на метрику качества ROUGE.

$ROUGE_1$:

$$ROUGE_1 = \frac{\sum_{g \in C \cap R} Count_{\text{match}}(g)}{\sum_{g \in C} Count_{\text{total}}(g)}$$

где $Count_{\text{match}}(g)$ - количество вхождений n-граммы g как в сгенерированном, так и в эталонном тексте, а $Count_{\text{total}}(g)$ - общее количество вхождений n-граммы g в сгенерированном тексте.

$ROUGE_2$:

$$ROUGE_2 = \frac{\sum_{g \in C_2 \cap R_2} Count_{\text{match}}(g)}{\sum_{g \in C_2} Count_{\text{total}}(g)}$$

где C_2 и R_2 - множества биграмм (n=2) в сгенерированном и эталонном тексте соответственно.

$ROUGE_L$:

$$ROUGE_L = \frac{\sum_{lcs \in LCS} |lcs|}{\sum_{r \in R} |r|}$$

где LCS- множество наибольших общих последовательностей между сгенерированным и эталонным текстами, и $|lcs|$ - длина наибольшей общей последовательности, а $|r|$ - длина эталонного текста.

Рассмотрим средний по размеру тестовой выбоки результат метрики Rouge для нашей модели

```
{'rouge1': 0.06561567406254055,
 'rouge2': 0.02861035422343324,
 'rougeL': 0.06514207862981704,
 'rougeLsum': 0.06472687167510056}
```

Рис. 5: ROUGE

Эти результаты ROUGE метрик указывают на то, что сгенерированный текст имеет низкое перекрытие с оригинальным текстом. ROUGE1 показывает перекрытие униграмм около 6.56%, ROUGE2 - перекрытие биграмм около 2.86%, а ROUGE-L - пересечение максимальной длины последовательных общих подпоследовательностей около 6.51%. Это может свидетельствовать о том, что сгенерированный текст не полностью передает смысл и модель требует улучшения

5 Заключение

Сформирован обзор литературы по нейросетевым языковым моделям, для дальнейшей работы выбраны модели типа T5, ввиду открытости, хорошего качества решения широкого спектра задач обработки естественного языка.

Сформировани обзор литературы по RAG для дальнейшего улучшения работоспособности языковой модели.

Реализована программа на языке для автоматической суммаризации на языке Python с использованием архитектуры T5.

Настоящее исследование нацелено на разработку метода аугментации запроса к языковым моделям с помощью векторных представлений текстов с целью улучшения качества генерируемых ответных последовательностей и при этом сокращении размера входной последовательности по сравнению с классическими методами RAG. В текущем семестре были проведены подготовительные работы и решались учебные задачи

Дальнейшие работы предполагают поиск метода кодирования текста векторами и разработку метода внедрения векторных представлений текстов в большую языковую модель

Список литературы

[1] https://habr.com/ru/articles/779526/

[2] https://education.yandex.ru/handbook/ml/article/yazykovye-modeli

[3] https://qudata.com/ml/ru/NN$_A$*ttention.html*