нетология

Пояснительная записка
к итоговому проекту на тему:

«Разработка системы семантического поиска аналогичных обращений в службу технической поддержки с использованием методов глубокого обучения»

Автор: Таранов Виктор
Группа: DLL-??

Оглавление

Введение

Цели и краткое описание проекта

Целью проекта является разработка прикладной системы семантического поиска, предназначенной для автоматизированного подбора исторически схожих по смыслу обращений в службу технической поддержки. Система ориентирована на практическое применение в службах Service Desk, где критически важны скорость реакции, воспроизводимость решений и снижение нагрузки на операторов поддержки.

В рамках проекта рассматривается предоставленный заказчиком архив обращений технической поддержки, включающий порядка 15 000 текстовых файлов. Каждый файл соответствует отдельному обращению и содержит исходный текст запроса пользователя, а также историю переписки между пользователем и службой поддержки. Архив является многоязычным и включает тексты на русском, английском и латышском языках, что накладывает дополнительные требования к выбору моделей обработки естественного языка и методам обучения.

Задача проекта заключается в построении системы семантического поиска, способной по новому входящему запросу находить наиболее близкие по смыслу исторические обращения.

1. Постановка задачи

1.1 Формулировка задачи

Исходная задача формулируется следующим образом: по входному тексту обращения пользователя необходимо автоматически определить набор из K наиболее семантически близких обращений из архива технической поддержки.

С формальной точки зрения задача сводится к задаче поиска ближайших соседей в векторном пространстве текстовых представлений. Для каждого текста необходимо построить вектор-представление (эмбеддинг), отражающее его семантический смысл, после чего определить меру близости между эмбеддингами нового запроса и эмбеддингами архивных обращений.

В качестве основной меры близости в проекте используется косинусное сходство, широко применяемое в задачах семантического поиска и информационного извлечения.

Рассматривалась также задача поиска по запросу наиболее подходящего решения проблемы, но так как проверенные решения проблем в предоставленном архиве никак не выделены, было решено ограничиться задачей поиска похожих по смыслу запросов.

1.2 Ограничения и требования

К системе предъявляются следующие требования:

- поддержка многоязычных текстов (для данного архива это русский, английский и латышский языки);
- устойчивость к орфографическим ошибкам и вариативности формулировок;
- приемлемое время ответа при размере архива порядка десятков тысяч документов;

- возможность дальнейшего улучшения качества за счёт дообучения модели на доменных данных;
- воспроизводимость результатов и понятность экспериментальной методологии для заказчика, не имеющего специальных знаний в области машинного обучения.

Также очевидным ограничением стали вычислительные ресурсы, имеющиеся в распоряжении исполнителя: персональный компьютер на базе процессора Apple M2 с ОЗУ 32 Мб, не имеющий графической карты.

1.3 Идея реализации

Задачу поиска k ближайших похожих текстов можно решать традиционными методами машинного обучения (TF-IDF, Word2Vec), но, как было наглядно показано в ходе курса Deep Learning, нейросети имеют ряд явных преимуществ:

- вычисление семантической близости, слабо зависящей от перефразирования, синонимов и словоформ;
- наличие доступных предобученных многоязыковых моделей;
- наличие практических методов и инструментов для дообучения на специализированных наборах данных.

Поэтому с самого начала было принято решение использовать нейросети для решения предложенной задачи. (К тому же это страшно интересно.)

Недостатком выбранного подхода может оказаться недостаточное количество вычислительных ресурсов исполнителя. На этот случай с заказчиком была оговорена возможность разумно ограниченного использования облачных ресурсов Google.

2. Анализ и подготовка данных

Целью данного этапа являлось преобразование неструктурированного архива обращений технической поддержки в пригодный для дальнейшей работы набор данных, обеспечивающий корректное обучение и оценку моделей семантического поиска.

Соответствующий код приведен в ноутбуке 1___prepare_dataset.ipynb.

2.1 Входные данные и формат экспорта

Заказчик предоставил архив ориентировочно из 15 000 обращений. Формат архива:

- list.txt: реестр, где каждая строка содержит три поля: краткую тему обращения, идентификатор инцидента и имя файла с историей переписки;
- отдельные файлы с перепиской по каждому инциденту, на которые ссылается list.txt.

Все файлы поставлены в кодировке UTF-16 и содержат тексты на русском, английском и латышском языках.

Анализ исходного архива выявил ряд проблем, которые могли помешать достижению цели проекта:

- отсутствие разметки языков. В архиве одновременно присутствуют обращения на русском, английском и латышском языках, причём язык обращения заранее не размечен;
- значительную часть текста в каждом файле занимают заголовки писем с адресами электронной почты получателей, как правило, повторяющихся в запросах на различные темы. Эта информация не несет смысловой нагрузки с точки зрения целей проекта и может привести к обучению модели не по схожести запросов, а по близости адресатов;
- аналогичная проблема наблюдается из-за присутствия автоматических подписей и повторяющихся дисклеймеров, размер которых часто многократно превышает размер содержательной части переписки;
- значительная вариативность длины текстов: от нескольких слов до многостраничной переписки;
- необычное расположение сообщений в переписке: в файле они располагаются в обратном хронологическом порядке, то есть в начале самое последнее письмо, а исходный запрос - в самом конце файла.

Для приведения архива в состояние, пригодное для достижения цели проекта, был проведен ряд описанных далее действий.

2.2 Разбор list.txt и первичная валидация

2.2.1 Парсинг реестра

Реестр list.txt был разобран регулярным выражением, рассчитанным на три поля, заключённые в кавычки и разделённые запятыми.

После чтения файла была сформирована таблица pandas с именем df и со столбцами:

- subject: тема обращения;
- id: идентификатор обращения;
- details_filename: имя файла переписки.

По результату парсинга получена таблица. содержащая 15926 строк.

2.2.2 Игнорирование служебных записей CY__*

Уже после передачи архива заказчик попросил игнорировать все записи, у которых идентификатор начинается с CY__. В архиве обнаружилось 699 таких записей. Эти строки были удалены из df.

2.2.3 Проверка целостности и соответствия имён файлов

Проверки целостности включали:

- отсутствие пустых значений в ключевых столбцах (subject, id, details_filename): пропусков не обнаружено;

- соответствие имени файла и идентификатора: проверка details_filename == id + ".txt" показала 0 несовпадений.

2.2.4 Проблема неуникальных идентификаторов и решение

Также была проверена уникальность id. Выявлено наличие дублей, причём у части дублей тема subject отличается. По пояснению заказчика, в исходной системе не было жёсткого контроля уникальности идентификаторов.

После обсуждения было принято решение удалить все записи, для которых id встречается более одного раза, не оставляя ни одной строки из группы дублей, так как нельзя гарантировать корректность выбора «правильной» строки.

После удаления дублей в таблице df осталось 13284 записи.

2.3 Загрузка переписки из файлов

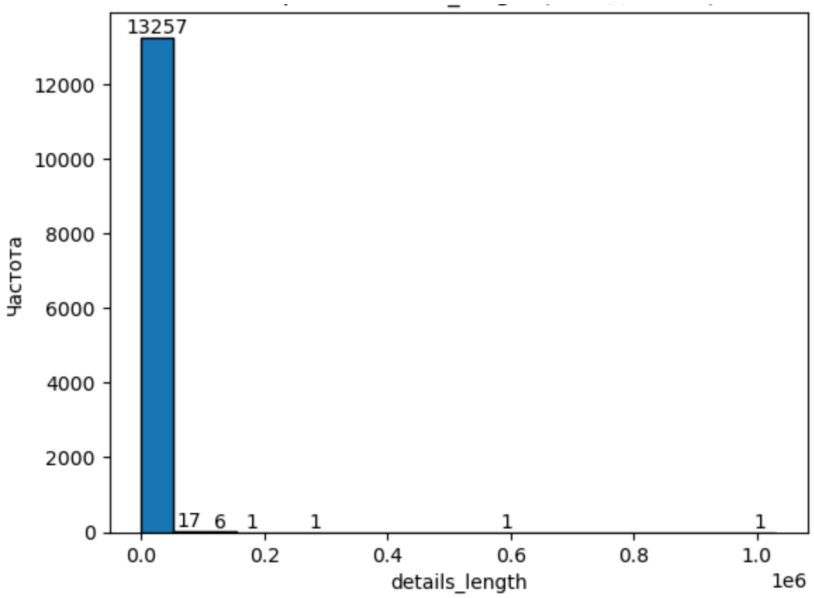
Для каждой строки датафрейма был загружен текст переписки из файла details_filename (UTF-16). Результат сохранен в колонки:

- details: полный текст файла;
- details_length: длина текста.

Время загрузки составило около 2 секунд.

2.4 Анализ распределения длины переписки

Анализ распределения длины показал наличие небольшого числа аномально больших текстов.



Изучение этих текстов показало, что они содержат:

- изображения, закодированные как "Content-Type: image/..."
- тексты, закодированные через base64

- очень длинная кое-где история переписки
- очень длинные исходные запросы (с логами, к примеру)

По согласованию с заказчиком закодированные файлы и файлы с картинками были удалены.

После этого в наборе данных по-прежнему наблюдался "хвост" из небольшого количества аномально длинных текстов, но их было решено оставить, так как это были регулярные тексты, структура которых соответствовала структуре остальных файлов.

2.5 Нормализация и удаление лишнего

2.5.1 Нормализация текста

Для начала была проведена типовая операция по нормализации текста, упрощающая дальнейшую обработку. Для этого реализована функция normalize_text(text), которая:

- удаляет маркеры цитирования вида > в начале строк;
- сводит любые пробельные последовательности к одному пробелу;
- убирает пробел перед знаками пунктуации;
- приводит текст к нижнему регистру.

Действия, характерные для традиционных методов машинного обучения (стемминг, лемматизация, удаление стоп-слов), не проводились.

2.5.2 Подход к удалению подписей и дисклеймеров

Строгого решения для удаления многострочных подписей и дисклеймеров найти не удалось, поэтому пришлось применить эмпирический подход: часто встречающиеся подписи и дисклеймеры были помещены в отдельный файл known_blocks_to_remove.txt, где они разделены специальной строкой-разделителем. Этот файл после передачи системы в эксплуатацию можно будет пополнять новыми образцами.

2.5.3 Удаление блоков из переписки

Реализована функция clean_text(text, block_list), которая:

- нормализует текст;
- ищет в тексте и удаляет каждый блок, содержащийся в known_blocks_to_remove.txt;
- возвращает очищенный текст, его длину и количество удалённых блоков.

Результат применяется ко всем записям и сохраняется в новых колонках df:

- details_no_signs: переписка после удаления известных подписей/дисклеймеров;
- details_no_signs_length: длина;
- removed_blocks_count: сколько блоков было удалено.

2.5.4 Поиск подписей

Для итеративного пополнения known_blocks_to_remove.txt реализована вспомогательная функция find_signatures(df), ведущая поиск подписей по наличию телефонных номеров. Идея: подписи часто содержат телефоны, поэтому наличие номера является хорошим маркером.

Функция выдает отчет, в котором показывает найденные номера телефонов, id файлов и количество упоминаний.

Добавление найденных подписей в known_blocks_to_remove.txt предоставлено пользователю, так как возможности автоматически определить границы подписи найдено не было.

Такой подход к удалению подписей потребовал много ручной работы, поэтому после проведения примерно 200 циклов поиска работа была остановлена. В архиве, вероятно, осталось 2620 редко встречающихся подписей.

2.6 Удаление адресов электронной почты

Еще одним фактором, который может исказить результаты обучения, является большое количество адресов электронной почты, так как в истории сообщений полностью присутствуют заголовки писем, размер которых часто превышает размер запроса.

Особенность данного архива: помимо адресов email в интернет-формате RFC 5321/RFC 5322 встречаются иерархические адреса Lotus Notes по спецификации X.500. Поэтому для поиска адресов применялись два паттерна:

- адреса в формате имя@список-доменов
- адреса в формате имя/подразделение/.../организация

Для реализации поиска создана функция remove_emails(text), результаты применения которой сохранены в колонках df:

- details_no_signs_emails: очищенный текст;
- details_no_signs_emails_length: длина очищенного текста.

Задача поиска адресов оказалась нетривиальной, так как потребовала создания сложных регулярных выражений, которые, с одной стороны, должны были найти все вхождения адресов, а с другой стороны - не привести к удалению полезного контента.

Поэтому для проверки работы функции remove_emails была также написана проверочная функция find_email_snippets, который показывает фрагменты текста, в которых подозревается наличие почтовых адресов.

2.7 Выделение запроса из истории переписки

По изучению архива было выяснено:

- переписка хранится в обратном хронологическом порядке, так что исходный запрос оказывается в самом конце файла;
- сообщения разделены заголовками вида ABC - DD.MM.YYYY или ABC.DD.MM.YYYY, где ABC - инициалы автора латиницей.

Из-за такой необычной структуры данных пришлось решить задачу разбиения истории переписки на отдельные фрагменты и выделение исходного запроса как последнего из фрагментов. Реализация выполнена в виде функции extract_request(text), которая применяется ко всем записям и формирует ключевые для дальнейшей работы колонки.

2.8 Определение языка запроса

Изначально было известно, что техническая поддержка заказчика ведется на русском, английском и латышском языках, и в сообщениях отсутствует атрибут, указывающий на язык. Но для наблюдения за тем, как модель работает с различными языками, такой атрибут будет необходим. Поэтому было проведено автоматическое определение языка, исходя из разумного предположения о том, что общение в рамках одного запроса ведется на одном языке.

Для определения языка использовалась модель fastText lid.176.bin, выбранная по рекомендациям в https://modelpredict.com/language-identification-survey. Реализована функция detect_lang_ft(model, text), результаты применения которой сохранены в колонках df:

- language: код языка (ru, en, lv и др.);
- lang_prob: вероятность.

По итогам определения было обнаружено 45 языков.

45 rows 45 rows × 2 cols

language	count	avg_prob
ru	5623	0.927360
lv	4215	0.842232
en	2959	0.647669
sl	32	0.186156
lt	31	0.345032
pl	29	0.284414
sr	16	0.273625
pt	12	0.336167
de	12	0.520583
fr	12	0.459750
cs	9	0.221889
es	9	0.227778

Основными языками ожидаемо стали:

- ru: 5623 файлов (средняя вероятность 0.927);
- lv: 4215 (0.842);
- en: 2959 (0.648);

Были также обнаружены небольшие «хвосты» других языков (sl, lt, pl и др.). Изучение соответствующих файлов показало, что эти случаи вызваны:

- русскими текстами, написанными транслитерацией;
- оставшимися неубранными дисклеймерами на других языках.

Для упрощения дальнейших исследований было принято решение оставить только те записи, в которых были определены три целевых языка, и удалить остальные.

Итоговый размер df: 12797 записей. Таблица df в итоговом виде была сохранена в файл формата requests.parquet для дальнейшего использования.

2.9 Выводы по разделу

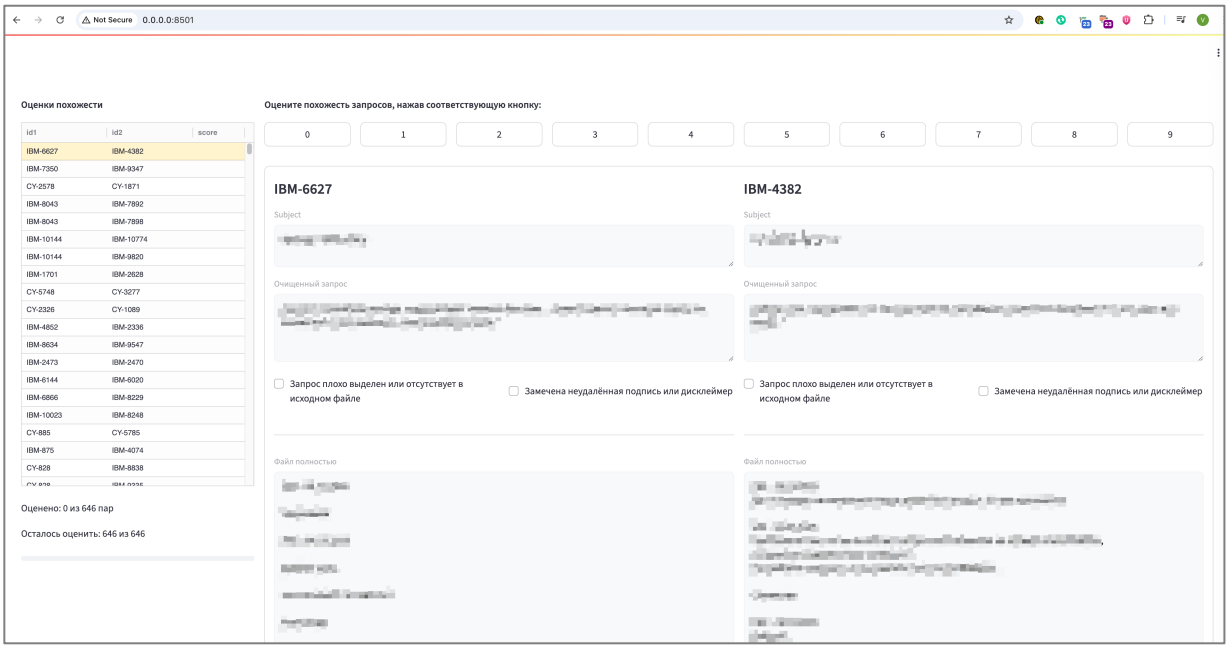
Была проведена работа по анализу и очистке исходного архива, направленная на достижение основной цели проекта (поиска похожих запросов).

К сожалению, полной автоматической воспроизводимости очистки достичь не удалось, так как корректное удаление адресов, подписей и дисклеймеров потребовало ручного труда. Также не в некоторых случаях не удалось правильно определить язык.

При переходе к промышленной эксплуатации заказчику будут даны рекомендации по очистке сообщений на начальном этапе их обработки оператором.

3. Формирование экспертной разметки

Для того, чтобы предоставить информацию для дообучения модели на предметной области заказчика, а также для оценки качества дообучения, с заказчиком была достигнута договоренность о том, что к работе будут привлечены его эксперты. Для этого было разработано вспомогательное приложение, которое в удобном для эксперта виде отображало на экране пары запросов, отобранные моделью, и позволяло эксперту давать оценку похожести по шкале от 0 (совершенно не похожи) до 9 (идеально похожи).



Исходный код приложения приведен в файле check_similarity.py.

Так как у заказчика не было практики работы с python, но он использует Docker, приложение вместе со всей необходимой инфраструктурой было упаковано в контейнер Docker и передано заказчику с инструкциями по его запуску.

Первоначально заказчику было предложено проставить оценки случайно отобранным парам запросов, но при таком способе отбора почти все пары получали низкие оценки, что поставило под сомнение возможность дообучения и вызвало неудовольствие заказчика. Он потребовал предоставить эксперту набор пар запросов, уже отобранных моделью как

похожие. Это привело к изменению плана проекта: в первую очередь решено было выбрать модель, провести на ее базе первоначальный отбор пар запросов, а затем дообучать эту же модель на результатах экспертной оценки.

При обсуждении критериев качества работы модели заказчик остановился на единственной метрике Precision, отражающей степень правдоподобности положительных оценок, выданных моделью:

Precision = TP / (TP + FP)

где TP - число пар, которые модель считает похожими и эксперт подтверждает, а FP - число пар, которые модель считает похожими, но эксперт не подтверждает.

Соответствующий код представлен в ноутбуке 2___prepare_expert_scoring.ipynb.

3.1 Выбор модели для предварительной генерации пар

Предварительный список моделей-кандидатов был сформирован на основе следующих критериев:

- поддержка русского, английского и латышского языков;
- ориентация на задачи sentence similarity;
- возможность работы на локальном компьютере (ОЗУ 32 ГБ) без GPU.

В качестве кандидатов рассматривались трансформерные модели, представленные на HuggingFace. По фильтрам (ru & lv & en), ограничению на размер (< 6B) и задаче Sentence Similarity были обнаружены 24 модели.

24 rows 24 rows × 5 cols

	Group	Model	Size	Updated	Downloads
18	sartifyllc	African-Cross-Lingua-Embeddings-Model	0.5B	31.01.2025	231
22	setu4993	LEALLA-base	0.1B	19.10.2023	252
11	setu4993	LEALLA-large	0.1B	19.10.2023	46
10	setu4993	LEALLA-small	0.1B	19.10.2023	35
21	setu4993	LaBSE	0.5B	19.10.2023	6.98k
15	sentence-transformers	LaBSE	0.5B	06.03.2025	899k
23	Blaxzter	LaBSE-sentence-embeddings	0.5B	04.05.2023	46
3	onelevelstudio	M-E5-BASE	0.3B	19.03.2025	9
14	onelevelstudio	M-MPNET-BASE	0.3B	19.03.2025	7
13	onelevelstudio	ML-E5-0.3B	0.3B	26.03.2025	9
2	onelevelstudio	MPNET-0.3B	0.3B	26.03.2025	9
8	aiana94	NaSE	0.5B	19.06.2024	7
7	cointegrated	SONAR_200_text_encoder	0.8B	08.11.2024	2.65k
4	sentence-transformers	distiluse-base-multilingual-c	0.1B	06.03.2025	1.4M
20	antoinelouis	dpr-xm	0.9B	25.03.2024	38
17	intfloat	multilingual-e5-base	0.3B	17.02.2025	1.49M
9	woody72	multilingual-e5-base	0.3B	05.11.2023	8
12	Hiveurban	multilingual-e5-base	0.3B	25.06.2025	11
6	intfloat	multilingual-e5-small	0.1B	17.02.2025	1.9M
19	Marqo	multilingual-e5-small	0.1B	05.09.2024	20
0	oxygeneDev	paraphrase-multilingual	0.1B	17.07.2025	1.16k
16	sentence-transformers	paraphrase-multilingual-MiniL…	0.1B	06.03.2025	11.5M
5	sentence-transformers	paraphrase-multilingual-mpnet	0.3B	06.03.2025	3.21M
1	ipetrukha	ukr-paraphrase-multilingual-mpnet-base	0.3B	30.03.2025	12

По критерию популярности/актуальности из них были выбраны семейства paraphrase-multilingual и multilingual-e5. Для проверки работоспособности этих моделей на компьютере исполнителя было проведено испытание, детальная информацию о методике и результатах которого приведена в ноутбуке 2_1_model_benchmaks.ipynb.

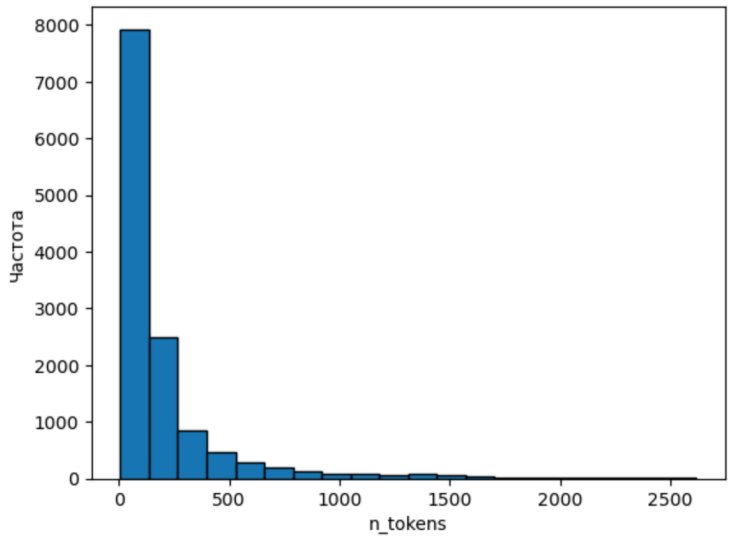
Все отобранные модели показали приемлемое время загрузки и вычислений без дообучения. Для начала экспериментов было принято решение использовать модель intfloat/multilingual-e5-small как минимально достаточную по качеству и ресурсам.

3.2 Подготовка текстов под требования E5

Особенность моделей E5: требование префиксов для разных типов входа. Для «запросов» (query) добавляется префикс query:. Это реализовано функцией compose_e5_request(subject, details, prefix="query"), которая добавляет в начало запроса префикс и содержимое колонки subject.

3.3 Проверка ограничения на длину входа

Выбранная модель обеспечивает длину входа в 512 токенов. Чтобы убедиться, что этого достаточно в данном случае, была проведена токенизация запросов и проведен простейший анализ результатов.




```
Показано значений: 12797 из 12797 (100.00%)
Ширина одного бина: 131
```

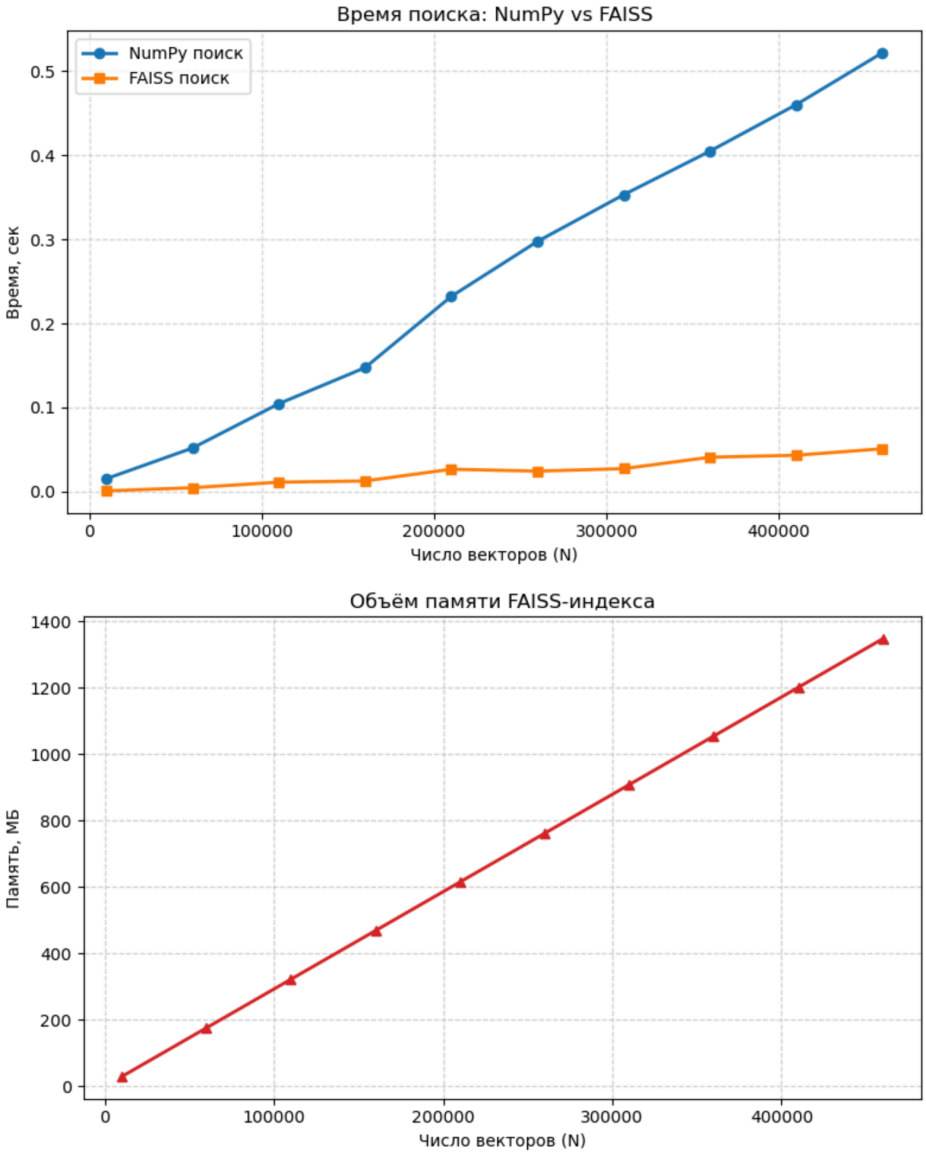
Анализ показал, что 91% запросов укладываются в лимит 512 токенов. Для оставшихся примерно 9% принята простая стратегия: модель будет обрезать хвост текста автоматически.

3.4 Вычисление эмбеддингов и оценка ресурсоёмкости

Для расчёта эмбеддингов была реализована функция encode_requests(model, texts, backup_file="embeddings.npy

Расчет эмбеддингов занял около 42 секунд. Объем массива составил 19 МБ.

Отдельно изучался вопрос использования индекса FAISS. Для текущего набора данных абсолютный выигрыш по времени оказался несущественным, даже при условии многократного роста в будущем. Поэтому было принято решение не использовать FAISS и проводить прямой поиск похожих пар по скалярному произведению.



Подробности бенчмаркинга FAISS приведены в ноутбуке 2_2_faiss_benchmark.ipynb

3.5 Базовый поиск похожих запросов и проверка мультиязычности

Похожие запросы определялись по максимуму косинусного сходства. При этом проверялся вопрос: действительно ли модель работает как мультиязычная, то есть будет ли находить похожие запросы на другом языке.

Для этого проводился следующий эксперимент:

- выбирались100 случайных запросов;
- для каждого из них модель выдавала наиболее похожий (top 1);
- фиксировались языки исходного запроса и найденного кандидата.

Вывод: несмотря на мультиязычность, на первых позициях модель чаще всего возвращает запросы на том же языке, что и исходный. Было решено учесть этот факт при формировании списка запросов для экспертной оценки.

3.6 Генерация пар для экспертной оценки

Учитывая наблюдение о предпочтении одного языка, была реализована следующая стратегия генерации пар запросов.

Для каждого случайно выбранного исходного запроса формируются до двух пар:

1) top-1 наиболее похожий запрос (независимо от языка);
2) если top-1 оказался на том же языке, добавляется ближайший из top-k на другом языке (если такой найден).

Таким образом эксперт получает не только «очевидные» пары внутри одного языка, но и дополнительные мультиязычные случаи, что повышает ценность данных для последующего обучения.

Для генерации пар запросов была реализована функция collect_similar_pairs, которая реализует описанную стратегию, а также исключает появление дублирующихся пар.

В итоге был сформирован файл pairs_for_expert_scoring.csv, содержащий колонки id1, id2. Этот файл вместе со вспомогательным приложением для выставления оценок и очищенным набором данных requests.parquet был передан эксперту. В ответ был получен файл pairs_with_expert_scoring.csv, который содержит те же id1, id2, а также выставленные экспертом оценки по шкале от 0 до 9.

3.8 Выводы по разделу

Выбрана базовая модель intfloat/multilingual-e5-small, удовлетворяющая требованиям многоязычности и ресурсным ограничениям.

Реализовано единое каноническое представление текста e5_request с префиксом query:, необходимым для корректной работы E5.

Проведена проверка длины входа: 91% обращений укладываются в 512 токенов; принята стратегия обрезки хвоста для оставшихся.

Рассчитаны и сохранены эмбеддинги.

Проведено испытание FAISS и принято решение не использовать его ввиду незначительной экономии времени.

Проведена проверка мультиязычности на 100 случайных запросах; выявлено предпочтение выдачи top-1 на том же языке.

Реализован алгоритм формирования пар для эксперта: top-1 плюс ближайший на другом языке (если найден), с исключением дубликатов.

Подготовлен набор данных для экспертной оценки.

Разработано приложение для проведения экспертной оценки с инструкциями по его установке и эксплуатации.

Приложение и набор данных переданы заказчику для проведения экспертной оценки.

4. Дообучение модели

Раздел поясняет методику и результаты дообучения модели на основе экспертных оценок. Соответствующий код приведен в ноутбуке 3___transfer_learning.ipynb.

4.1 Переход на дообучение в облаке

Попытка выполнить дообучение локально без графического процессора оказалась практически невозможной, так как каждый прогон должен был выполняться несколько часов. После консультаций с заказчиком было принято решение выполнить дообучение в Google Colab с арендой GPU. Для основного эксперимента было достаточно минимального GPU T4 (ОЗУ 15 GB). Для экспериментов с более крупными моделями потребовалась GPU A100 (ОЗУ 40 GB).

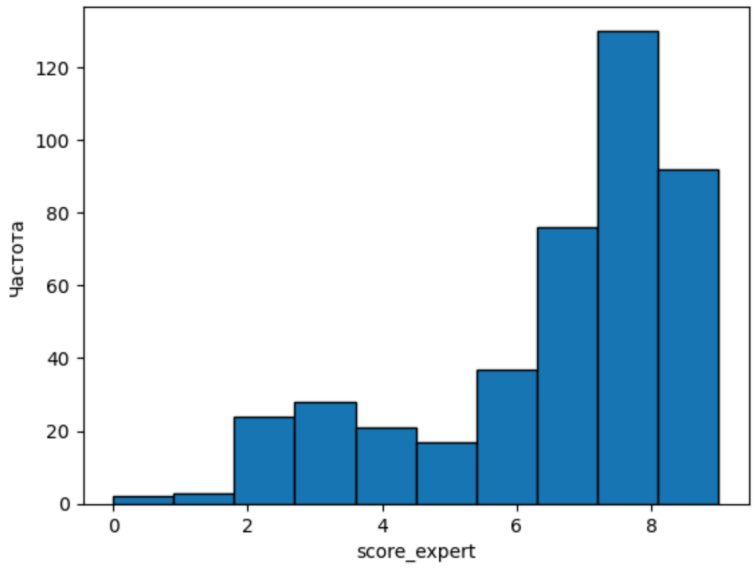
4.2 Базовый уровень качества до обучения

Для дообучения использовались результаты оценок похожести пар запросов, выданных экспертом.

Изначально эксперт оценивал близость пар по шкале от 0 до 9, но в итоге пришел к выводу, что в будущем для оценки похожести достаточно выставлять более простую оценку 0 (не похожи) или 1 (похожи).

Соответственно, уже проставленные оценки было решено бинаризировать, приняв за "похожие" все пары, которым была выставлена оценка выше 6 баллов. Соответственно упростился выбор ключевой метрики для оценки качества работы модели: была выбрана оценка precision = TP / (TP + FP), показывающая, насколько выбор модели близок к оценкам эксперта.

Эксперт совершил трудовой подвиг и смог оценить 430 пар запросов. Как можно видеть, значительная часть из них была оценена высоко.




```
Показано значений: 430 из 430 (100.00%)
Ширина одного бина: 1
```

Начальное значение precision по итогам экспертной оценки с учетом согласованного порога >6 составило 0.69, что уже было позитивно воспринято заказчиком. Целью дообучения было поставлено улучшить precision на базе экспертных оценок.

4.3 Подготовка данных для обучения

4.3.1 Источник разметки

В качестве обучающих данных использовался файл pairs_with_expert_scores.csv, содержащий колонки id1, id2 и score (оценка эксперта). Оценки эксперта были приведены к показателю 0 или 1 на основе согласованного порога >6.

4.3.2 Привязка пар к текстам обращений

Для извлечения текстов из файла requests.parquet был загружен ранее очищенный набор данных с колонками:

- id: идентификатор обращения;
- e5_request: текст обращения в виде, подготовленном для моделей семейства E5.

Из пар (id1, id2) строился датасет для обучения в формате HuggingFace Datasets:

- sentence1: текст обращения с id1;
- sentence2: текст обращения с id2;
- label: бинарная метка эксперта label_expert.

Для построения датасета была создана функция build_dataset(...), которая связывала идентификаторы с текстами через индексирование по id.

4.4 Формирование train/val разбиения

В ходе простого случайного разбиения датасета на обучающую и валидационную выборки выяснилось, что изначальное значение precision в них может сильно различаться, что в дальнейшем плохо скажется на обучении и анализе результатов. Это было очевидным следствием небольшого размера датасета: доля положительных оценок эксперта в одной из выборок случайно оказывалась существенно больше, чем в другой.

Проблема была устранена перебором random_state при вызове функции scores.sample до тех пор, пока пока precision на train и val не становились достаточно близкими.

4.5 Выбор базовой модели и постановка обучения

4.5.1 Модель семейства E5 (small)

В качестве исходной модели использовалась та же модель intfloat/multilingual-e5-small, что применялась для первоначального поиска похожих запросов.

4.5.2 Выбор функции потерь

Поскольку целевая разметка бинарная (0/1), была выбрана функция потерь ContrastiveLoss из библиотеки Sentence Transformers (https://github.com/huggingface/sentence-transformers/blob/main/sentence_transformers/losses/ContrastiveLoss.py)

4.6 Гиперпараметры обучения и контроль качества

Обучение выполнялось с использованием SentenceTransformerTrainer и набора аргументов SentenceTransformerTrainingArguments со следующими ключевыми параметрами, подобранными после некоторого количества неудачных экспериментов:

- num_train_epochs = 10
- per_device_train_batch_size = 32
- per_device_eval_batch_size = 32
- learning_rate = 2e-5
- warmup_ratio = 0.1
- eval_strategy = "epoch" и logging_strategy = "epoch"
- save_strategy = "epoch"
- load_best_model_at_end = True
- metric_for_best_model = "eval_loss", greater_is_better = False

Таким образом, после каждой эпохи выполнялась валидация, сохранялся чекпойнт, а по окончании обучения автоматически восстанавливалась версия модели с наименьшим eval_loss.

4.7 Методика вычисления precision после дообучения

Для оценки качества дообученной модели была написана функция tuned_precision(model, ds), которая:

1. Строит нормализованные эмбеддинги sentence1 и sentence2.
2. Вычисляет cosine similarity как скалярное произведение нормализованных векторов.
3. Преобразует similarity в бинарный прогноз по согласованному порогу.
4. Считает precision относительно истинных бинаризированных оценок эксперта.

4.8 Сравнение нескольких дообученных моделей

Последовательно выполнялись эксперименты с тремя вариантами дообучения:

1. Базовая модель intfloat/multilingual-e5-small (без fine-tuning) - контрольное значение.
2. Дообученный вариант small-модели intfloat/multilingual-e5-small-finetuned.
3. Дообученный вариант более крупной модели intfloat/multilingual-e5-base-finetuned.
4. Дообученная модель из другого семейства sentence-transformers/paraphrase-multilingual-mpnet-base-v2-finetuned.

Переход к multilingual-e5-base потребовал смены GPU на A100 (40 GB) из-за увеличенного потребления памяти.

Суммарное потребление ресурсов Google Colab составило около 50 единиц, что примерно соответствует 5 долларам США.

Итоговые значения precision на валидационной выборке составили:

- intfloat/multilingual-e5-small: 0.693
- intfloat/multilingual-e5-small-finetuned: 0.787
- multilingual-e5-base-finetuned: 0.766
- paraphrase-multilingual-mpnet-base-v2-finetuned: 0.773

4.9 Выводы по разделу

Дообучение изначально выбранной small-модели дало наибольший прирост precision (с 0.693 до 0.787). При этом переход к более крупной модели (E5-base) и альтернативному семейству (MPNet) не дал дополнительного выигрыша, несмотря на более высокие вычислительные затраты.

Несмотря на небольшой размер обучающей выборки (менее 3% исходного архива), дообучение показало заметное улучшение качества работы модели. Модель intfloat/multilingual-e5-small будет рекомендована для промышленного внедрения и дальнейшего дообучения.

При внедрении системы в постоянную эксплуатацию заказчику будет предложено давать оценку подбора похожих для каждого вновь поступающего запроса, и таким образом постоянно расширять набор данных для последующих циклов дообучения модели.

5. Заключение

В рамках данного проекта была разработана прикладная система семантического поиска аналогичных обращений в службу технической поддержки с использованием методов глубокого обучения. Работа выполнена на реальных, слабо структурированных и многоязычных данных, что приближает полученные результаты к условиям практической эксплуатации.

5.1 Основные результаты работы

В ходе выполнения проекта были последовательно решены следующие задачи:

1. Проведён анализ исходного архива обращений в техническую поддержку и реализована процедура подготовки данных, позволяющая преобразовать неструктурированные текстовые файлы в датасет, пригодный для применения трансформерных моделей. Из-за слабой структурированности данных не удалось достичь полной автоматической воспроизводимости результатов.
2. Выполнен отбор и проведен бенчмарк нескольких многоязычных моделей семантических эмбеддингов на экспертно размеченных данных, по результатам которого выбрана базовая модель intfloat/multilingual-e5-small как оптимальная по соотношению качества и вычислительных затрат.
3. Сформирована методика экспертной оценки семантической близости обращений, включающая:
    - генерацию пар запросов с использованием базовой модели;
    - контроль мультиязычных случаев;
    - подготовку данных для ручной разметки;
    - создание вспомогательного приложения для проведения экспертной оценки.

4. Экспериментально исследована целесообразность применения библиотеки FAISS для ускорения поиска.
5. Реализовано дообучение выбранной модели с использованием экспертной разметки в бинарной постановке задачи. Показано, что дообучение позволяет существенно повысить precision семантического поиска (с 0.693 до 0.787) даже на небольшом обучающем датасете (менее 3% от всего архива).

5.2 Практическая значимость

Разработанная система может быть использована как прототип систем, внедряемых в реальных службах технической поддержки для:

- ускорения обработки новых обращений за счёт поиска релевантных исторических инцидентов;
- повышения качества ответов операторов поддержки;
- снижения зависимости от индивидуального опыта конкретных специалистов;
- последующего накопления и использования экспертных оценок как обучающего сигнала.

Особое значение имеет тот факт, что система ориентирована на многоязычную среду и не требует предварительного перевода текстов, что важно для международных компаний и распределённых команд поддержки.

5.3 Ограничения реализованного решения

Несмотря на достигнутые результаты, разработанное решение имеет ряд ограничений:

- не удалось добиться полной автоматической воспроизводимости очистки данных. При развитии системы эта проблема может быть устранена путем выделения содержательной части оператором в ходе обработки запросов;
- дообучение проводилось только на данных, размеченных экспертом. Получение этих данных потребовало существенных усилий. При развитии системы имеет смысл изучить и дополнительно применить другие методы, например дообучение на размеченном датасете из аналогичной предметной области (The TechQA Dataset, https://arxiv.org/abs/1911.02984);
- для больших архивов обращений (сотни тысяч и более) может потребоваться пересмотр решения об отказе от векторной индексации.

5.4 Направления дальнейшего развития

В качестве направлений дальнейшего развития проекта можно выделить:

- интеграцию разработанной системы в действующую у заказчика платформу Service Desk, с реализацией возможности дать оценку результатам каждого поискового запроса. Результатом будет непрерывное расширение объёма экспертно-размеченных данных;
- разработку методов и инструментов для периодического дообучения модели по мере накопления новых обращений;
- разработку методов и инструментов для разделения полезного контента сообщений и служебной информации (адресов, подписей, дисклеймеров) на этапе добавления сообщений в архив;

- исследование гибридных подходов, сочетающих семантический поиск и структурированные атрибуты обращений (категории, ключевые слова, даты обращения и т.д.);
- структурирование исходных данных с выделением найденных и подтвержденных решений из потока переписки, и переход от поиска похожих запросов к поиску подходящих решений;
- дообучение модели на одном из открыто доступных размеченных датасетов, относящихся к предметной области заказчика.

Список использованных источников

1. Attention is All You Need.
   https://arxiv.org/abs/1706.03762
2. Comparison of language identification models.
   https://modelpredict.com/language-identification-survey
3. HuggingFace. *Transformers Library Documentation*
   https://huggingface.co/docs/transformers/
4. HuggingFace. *intfloat/multilingual-e5-small*.
   https://huggingface.co/intfloat/multilingual-e5-small
5. HuggingFace. *intfloat/multilingual-e5-base*.
   https://huggingface.co/intfloat/multilingual-e5-base
6. HuggingFace. *sentence-transformers/paraphrase-multilingual-mpnet-base-v2*.
   https://huggingface.co/sentence-transformers/paraphrase-multilingual-mpnet-base-v2
7. HuggingFace. *sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2*. –
   https://huggingface.co/sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2
8. ContrastiveLoss function
   https://github.com/huggingface/sentence-transformers/blob/main/sentence_transformers/losses/ContrastiveLoss.py
9. *Sentence Transformers Documentation*.
   https://www.sbert.net/
10. FAISS Documentation.
    https://github.com/facebookresearch/faiss
11. Google. *Google Colab Documentation*.
    https://colab.research.google.com/
12. NumPy Developers. *NumPy Documentation*.
    https://numpy.org/doc/
13. RFC 5322: Internet Message Format.
    https://datatracker.ietf.org/doc/html/rfc5322
14. RFC 5321: Simple Mail Transfer Protocol.
    https://datatracker.ietf.org/doc/html/rfc5321
15. Идентификация языков с помощью fastText.
    https://fasttext.cc/docs/en/language-identification.html
16. The TechQA Dataset
    https://arxiv.org/abs/1911.02984