**Слайд 1**

Добрый день! Мы представляем проект по теме «Мультимодальная RAG-система по отчету Сбера».

**Слайд 2 – Актуальность темы и ее проблематика**

RAG или Retrieval Augmented Generation, как известно – это техника, позволяющая создавать на базе генеративных моделей вопросно-ответные системы по собственным источникам данных. На слайде видим упрощенную схему RAG – запрос пользователя векторизуется (переводится в эмбеддинг), далее по косинусному сходству выполняется поиск наиболее релевантных чанков в базе данных, и тексты этих чанков подставляются в качестве контекста при запросе большой языковой модели, которая и формирует окончательный ответ. Как мы видим из свежего отчета, RAG является одной из самых популярных техник использования больших языковых моделей, поскольку возможность обсуждения с LLM вопросов по своим источникам данных является очень важным кейсом.

**Слайд 3 – State-of-the-art техники RAG**

Поскольку «ванильный» RAG создавать неинтересно, мы решили попробовать собрать state-of-the-art RAG на базе наиболее зарекомендовавших себя в последнее время подходов. Про обычные алгоритмы машинного обучения мы знаем, что соревнования на реальные деньги на площадке Kaggle позволили установить наиболее успешные из них – например, мы сейчас понимаем, что градиентный бустинг в большинстве задач машинного обучения эффективнее метода опорных векторов, если брать из коробки. Аналогично мы решили изучить, были ли недавно какие-либо соревнования по RAG, и обнаружили, что этой весной было известное соревнование Enterprise RAG Challenge, в котором победил Илья Райс. Причем в данном соревновании также требовалось создать вопросно-ответную систему по годовым отчетам компаний. Илья написал пост на Хабре про свое решение, поэтому мы решили его внимательно изучить и адаптировать для нашей задачи. Здесь изображена схема, которую привел Илья, и мы дальше подробно остановимся на каждом этапе, но сейчас вкратце упомянем основные этапы, чтобы обрисовать общую картину. Если брать широкими мазками, то мы парсим PDF, предобрабатываем, делим на чанки, векторизуем, и далее при запросе пользователя, когда мы находим наиболее релевантные чанки, мы возвращаем не только тексты, соответствующие этим чанкам, но вообще всю страницу, затем выполняем дополнительную фильтрацию этих страниц в ходе LLM-реранкинга и подаем наиболее релевантные страницы в контекст запроса для большой языковой модели, которая и формирует финальный ответ. Но давайте подробнее остановимся на каждом этапе, и на наших особенностях адаптации и реализации этого решения.

**Слайд 4 – Parsing**

На входе у нас был отчет Сбера в виде PDF файла (402 страницы). Мы взяли библиотеку pdfplumber и с помощью ее встроенных методов выделили текст, таблицы и изображения. Таблицы мы дополнительно перевели в формат Markdown, т.е. с разделяющими символами. Победитель Enterprise Challenge использовал тяжеловесную библиотеку Docling, арендовав сервер на Runpod – мы сделали так же, но достаточно долго разбирались с конфликтом зависимостей, и поскольку время было ограничено, решили вернуться к библиотеке pdfplumber, которая в парсинге текста и таблиц показала хорошее качество. Также мы запускали библиотеку unstructured, которую в свое время использовали в домашнем задании, но она работала очень медленно, то тоже ее отклонили. Таким образом мы получили текстовые данные и картинки.

**Слайд 5 – Ingestion**

Далее на следующим этапе нам требовалось векторизовать извлеченные данные. Для этого текстовые данные мы разбили с помощью рекурсивного Markdown-сплиттера на чанки по 300 символов с перекрытием 80 символов. Принцип работы рекурсивного сплиттера прост – мы принимаем некие символы или отступы в качестве разделяющих заголовков разных уровней и сначала делим весь текст по заголовкам первого уровня. Если получающиеся в результате разбиения чанки оказываются менее 300 символов, то разбиение останавливается – если нет, то продолжается по заголовкам второго уровня. И так далее. Почему мы взяли размер чанка 300 символов и оверлап 80 – это эмпирические значения – победитель соревнования выбрал чанки по 300 символов и оверлап 50, мы же взяли некоторый запас по оверлапу. В случае таблиц мы не делили на дополнительные чанки – как мы заметили, у нас таблицы были в основном небольшие и простой структуры, и мы приняли одну таблицу за один чанк. Далее мы с помощью обращения к ChatGPT-4o создали описания изображений, и описание одного изображения опять же приняли за один чанк. И далее все полученные текстовые чанки векторизовали путем обращения к text-embedding-3-large. Получили базу из около пяти с половиной тысяч векторов размерностью 3072 каждый, а также сформировали файл metadata.json, в котором сохранили информацию о том, какой чанк относится к какому типу (текст, таблица или изображение) и на какой странице находится, а также сохранили описание изображения. В дальнейшем эта информация понадобится нам для Parent-page enrichment, о чем сейчас подробнее расскажем. Стоит отметить, что вышеупомянутый этап предобработки – т.е. парсинга и создания базы данных – мы провели отдельно в Google Colab, а получившуюся базу данных положили рядом в пространстве на HuggingFace, в котором и развернули решение.

**Слайд 6 – Retrieval**

Итак, переходим на этап Retrieval. Запрос пользователя (например, «какие основные финансовые результаты Сбера за 2023 год?») мы переводим в векторное представление, используя всё ту же модель text-embedding-3-large, и, как и в классическом RAG, находим по косинусному сходству топ релевантных чанков (в нашем случае выбрали топ 20). Но далее, в отличие от классического RAG мы выбираем не сами чанки, а весь текст страницы, на которой расположен отобранный чанк, а также описание изображений (т.е. применяем так называемый parent-page enrichment). И вот эти тексты отдельных страниц по очереди отдаем на дополнительную оценку релевантности запросу по шкале от 1 до 10 быстрой и дешевой модели GPT-4o-mini, которая в свою очередь отбирает топ 5 страниц. И вот эти топ 5 страниц мы уже и используем в качестве контекста для LLM. Победитель Enterprise RAG Challenge использовал ту же последовательность действий, но брал несколько иное количество топ чанков и топ страниц (но у него на вход было и больше данных).

**Слайд 7 – Augmentation + Generation**

И далее мы формируем запрос, задавая роль эксперта, прося использовать цепочку мыслей, подавая исходный вопрос пользователя и найденный ранее релевантный контекст, а также просим GPT-4o ответить строго в формате json. Согласно статье победителя, просьба использовать Chain-of-Thoughts вела к лучшему качеству, и поэтому мы последовали этой рекомендации. Формат ответа мы верифицировали с помощью pydantic и оставили fallback функцию на случай, если LLM ответит не в json – вот эта запасная функция полностью аналогична, только мы уже не просим ответить в формате json и не выполняем верификацию с помощью pydantic.

**Слайд 8 – Demo**

На данном слайде приведен скриншот развернутой нами системы. Но мы можем посмотреть работу системы вживую (переходим в пространство на HuggingFace).

**Слайд 9 – Выводы**

В заключение отметим основные выводы по работе. RAG – одно из наиболее актуальных применений больших языковых моделей на сегодняшний день. В работе была реализована мультимодальная RAG-система, архитектура которой была вдохновлена победным решением недавнего крупного соревнования по RAG. В третьем пункте еще раз перечислены основные используемые библиотеки и технологии. С какими основными сложностями мы столкнулись – это не совсем верное выделение и распознавание картинок – все-таки, если бы было больше времени, лучше было бы использовать Docling, а также уточнить промпт для GPT-4o для распознавания картинок и реализовать повторный вызов, если GPT-4o отказывается описать изображение, поскольку на нем есть люди (возможно, это их какой-то фильтр, который то срабатывает, то нет). На будущее интересно было бы замерить качество собранного пайплайна (например, с помощью библиотеки RAGAS) и сравнить собранный пайплайн с более простыми.

**Слайд 10 – Список используемых источников/программных средств**

Как уже говорили, использовали Google Colab для парсинга и создания файла базы данных, а само решение развернули на HuggingFace. Также приведена ссылка на статью победителя соревнования Enterprise RAG Challenge. На этом всё, спасибо за внимание.