**“RAG для Центрального Банка России”**

**Руководство по использованию и разработке**

**Версия 1.0.0**

**Дата выпуска: 14.03.2024**

**Автор: Чесников Леонид Сергеевич**

**Содержание:**

[**1. Описание проекта. 3**](#_v7xexun5pxhj)

[**2. Локальная установка 4**](#_v8wzcr4zmjpm)

[**3. Обработка документов 5**](#_p11r0y611ana)

[**4. Обработка текста и работа с моделями 6**](#_m823hfibubqc)

[**5. О базе данных ClickHouse 7**](#_3xuydggpvxtx)

[**6. Алгоритм парсинга файлов с сайта cbr.ru 9**](#_2so2i137ayd1)

[**7. API 10**](#_67kcdv4tca9u)

[**8. Ссылка на GitHub 11**](#_jazyk2ihqice)

[**9. Лицензия 11**](#_8exjmehjfy)

[**10. Контакты 12**](#_gwhm3ry21pd2)

# 

# **Описание проекта.**

**RAG модель для Центрального Банка России** — это инновационное решение, разработанное командой **POST ИИрония**. Модель представляет собой систему, способную **обрабатывать большие объемы текстовых данных**, **извлекать информацию** и **предоставлять ответы** на запросы пользователей, что значительно **повысит эффективность** работы с информацией в Центральном Банке России.

* **Цели проекта:**
  + **Создание системы для обработки** разнообразной **документации** Центрального Банка России с целью быстрого и точного предоставления информации.
  + **Разработка чатбота**, **способного отвечать на вопросы** пользователей и **оказывать поддержку** по различным темам, связанным с деятельностью Центрального Банка России.
  + **Реализация модели для семантического поиска** в документах Центрального Банка России для повышения эффективности работы с информацией**.**
* **Технологии и инструменты:**
  + **Python** для разработки программного обеспечения.
  + Библиотеки **transformers, PyTorch, FastAPI** для создания и обучения моделей и API.
  + **ClickHouse** для хранения и управления данными.
  + **Docker** для контейнеризации приложения.
  + **Git** для управления версиями и совместной работы над проектом.
  + **Другие инструменты:** Включают в себя стандартные библиотеки Python и инструменты для разработки веб-приложений.
  + **Модели:**

**sbert\_large\_nlu\_ru:** используетсядля преобразования текста в вектор.

**TinyLlama-1.1B-Chat-v1.0:** используетсядля интерактивного ведения диалога с пользователями

* **Функциональность проекта:**
  + **Обработка и анализ** различных **документов** Центрального Банка России с использованием модели **RAG** для выявления ключевой информации.
  + **Разработка чатбота** для общения с пользователями, **предоставляющего информацию** **и ответы** на вопросы на естественном языке.
  + **Создание системы семантического поиска** для быстрого доступа к нужным документам и информации Центрального Банка России.

# **Локальная установка**

**Требования**

**Python 3.11.8+**

**Jupyter Notebook или JupyterLab**

**Шаги установки**

* Клонирование репозитория

*git clone* [*https://github.com/POST-AIrony/PurpleHack2024.git*](https://github.com/POST-AIrony/PurpleHack2024.git)

* *Установка зависимостей*

*pip install -r requirements.txt*

**Запуск Jupyter Notebook**

* Запустите **Jupyter Notebook** или **JupyterLab**:

*jupyter notebook*

* Откройте файлы **inference.ipynb** и **append\_to\_clickhouse.ipynb** из репозитория в Jupyter Notebook.

**Запуск чатбота и поиска документов**

* В файле **inference.ipynb** выполните все ячейки кода для запуска чатбота и поиска документов по запросу.

**Добавление документа в базу данных**

* В файле **append\_to\_clickhouse.ipynb** выполните ячейку кода для добавления нового документа в базу данных ClickHouse.

**Заключение**

Локальная установка чатбота довольно проста и позволяет легко запустить и использовать систему для обработки запросов и управления документами.

# **Обработка документов**

**Введение**

В данном разделе представлено описание процесса обработки текстовых документов с использованием различных методов и алгоритмов. Рассматриваются основные концепции и классы, применяемые для обработки документов, включая разделение текста на части и создание новых документов на основе этих частей.

**Классы и методы**

Одним из ключевых классов, используемых для разделения текста на части, является класс **RecursiveChunker**. Он предоставляет механизмы для разбиения текста на части с учетом различных разделителей и ограничений на размер части. Класс **Document** представляет собой отдельный документ, который может быть разбит на части с помощью **RecursiveChunker**.

**Методы RecursiveChunker**

**\_merge\_splits**(self, splits: Iterable[str], separator: str) -> List[str]: объединяет части текста с заданным разделителем, учитывая ограничения на размер части.

\_join\_docs(self, docs: List[str], separator: str) -> Optional[str]: объединяет список частей текста в один текст с заданным разделителем.

**create\_documents**(self, texts: List[str], metadatas: Optional[List[dict]] = None) -> List[Document]: создает список документов на основе текстов.

**split\_documents**(self, documents: Iterable[Document]) -> List[Any]: разбивает документы на части и создает новые документы на основе частей текста.

**transform\_documents**(self, documents: Sequence[Document]) -> Sequence[Document]: трансформирует документы, разбивая их на части и создавая новые документы на основе частей текста.

**\_split\_text\_with\_regex**(text: str, separator: str, keep\_separator: bool) -> List[str]: разделяет текст на части с использованием регулярного выражения.

**\_split\_text**(self, text: str, separators: List[str]) -> List[str]: рекурсивно разбивает текст на части с учетом списка разделителей.

**Класс Document**

Класс Document представляет собой отдельный документ, который может быть разбит на части с помощью класса **RecursiveChunker**. Он содержит текст документа и метаданные, описывающие документ.

**Класс SentenceChunker**

Класс **SentenceChunker** представляет собой специализированный алгоритм для разбиения текста на предложения. Он используется для разделения документов на отдельные предложения, что упрощает анализ текста на уровне предложений.

**Основные преимущества**

Гибкость в настройке разделителей и ограничений на размер части позволяет эффективно обрабатывать разнообразные текстовые данные.

Возможность добавления метаданных к каждой части текста позволяет сохранять дополнительную информацию о документах.

Методы класса **RecursiveChunker** обеспечивают высокую производительность и надежность при обработке больших объемов текста.

Класс **SentenceChunker** облегчает разделение текста на предложения, что полезно при анализе текста на уровне предложений.

**Заключение**

Обработка документов с использованием классов **RecursiveChunker**, **Document** и **SentenceChunker** представляет собой эффективный и гибкий подход к обработке текстовых данных. Благодаря широким возможностям настройки и высокой производительности эти классы могут быть успешно применены в различных областях, требующих обработки текстовой информации.

# **Обработка текста и работа с моделями**

**Создание векторов (эмбеддингов)**

Для преобразования текста в векторное представление используется модель **transformer**. Процесс включает в себя следующие шаги:

* **Токенизация текста**: Исходный текст разбивается на токены с использованием токенизатора, предварительно загруженного из библиотеки **transformers**. Токенизация выполняется с учетом заданных параметров, таких как длина последовательности (max\_length), и может включать дополнительные преобразования, такие как добавление специальных токенов и обрезание текста при необходимости.
* **Кодирование текста**: Закодированный текст передается модели **transformer** для преобразования токенов в **вектора**. Модель возвращает выход, содержащий эмбеддинги токенов и другие данные.
* **Усреднение токенов**: Полученные эмбеддинги токенов **усредняются** с использованием **маски внимания**. Маска внимания указывает на **значимость токенов** и используется для **исключения паддингов** при усреднении.
* **Возвращение результата**: Усредненный эмбеддинг возвращается в качестве векторного представления текста.

**Работа моделей**

Для загрузки и использования моделей используется библиотека **transformers**. Процесс работы моделей включает в себя следующие шаги:

* **Загрузка модели и токенизатора**: Модель и токенизатор загружаются из предобученных параметров, указанных пользователем.
* **Преобразование текста в векторное представление**: Текст передается в функцию **txt2embeddings**, которая использует токенизатор и модель для преобразования текста в векторное представление.
* **Поиск результатов в базе данных**: Для поиска похожих векторов в базе данных используется функция **search\_results**, которая выполняет запрос к базе данных с учетом вектора и возвращает список результатов с наименованием, URL, датой, номером, текстом и расстоянием.
* **Генерация ответа от чатбота**: Для генерации ответа от чатбота на основе предоставленного чата и, возможно, документа, используется функция **generate\_answer**. Она создает **объект разговора**, добавляет сообщения из чата и, если предоставлен документ, добавляет его в контекст чата. Затем она генерирует ответ от чатбота с учетом параметров генерации текста.
* **Загрузка чатбота:** Для загрузки чатбота используется функция **load\_chatbot**, которая загружает чатбот с помощью **pipeline** из библиотеки **transformers**.
* **Загрузка моделей:** Для загрузки токенизатора и модели используется функция **load\_models**, которая загружает токенизатор и модель для указанной предобученной модели.

**Заключение**

Описанные выше классы и алгоритмы обеспечивают **эффективное** выполнение обработки документов и генерации ответов от чатбота. Их использование позволяет **быстро и качественно** обрабатывать текстовые данные, делая процесс работы с текстом более **удобным и эффективным**.

# **О базе данных ClickHouse**

1. **Основные характеристики**

**ClickHouse** - это колоночная система управления базами данных (СУБД), предназначенная для работы с большими объемами данных. Внутри **ClickHouse** хранятся данные различных типов, такие как текстовая информация, числовые значения, даты, заголовки документов, типы файлов, URL адреса, типы частей документов и векторные представления (Embedding).

1. **Структура данных**

**Text** и **Embedding**: основные данные, хранящиеся в **ClickHouse**. **Text** содержит текстовую информацию, а **Embedding** - векторные представления данных.

**Number**: регистрационные номера документов.

**Date**: даты, используемые для хранения информации о времени.

**Title**: заголовки документов, содержащие информацию о содержании документа.

**FileType**: типы файлов, хранящие информацию о формате файла.

**Url**: URL адреса, используемые для ссылок на источники данных.

**ChunkType**: Определитель того, при помощи какого алгоритма разделялся текст.

1. **Объем данных**

Общее количество строк в базе данных **ClickHouse** составляет **336000**. Это позволяет хранить и обрабатывать большие объёмы информации, что делает **ClickHouse** эффективным инструментом для анализа данных.

1. **Работа в Docker контейнере**

**ClickHouse** может быть развернут и использован внутри **Docker** контейнера. Это обеспечивает удобство в управлении базой данных и обеспечивает возможность быстрого развертывания **ClickHouse** в различных средах.

1. **Преимущества ClickHouse**

* **Высокая производительность** при обработке больших объемов данных.
* **Эффективное использование ресурсов** системы благодаря колоночной структуре данных.
* Возможность обработки запросов на большом количестве узлов, обеспечивая **масштабируемость**.
* **Поддержка различных типов данных**, позволяющая работать с разнообразной информацией.

**Заключение**

**ClickHouse** - это мощная колоночная система управления базами данных, обладающая высокой **производительностью** и **масштабируемостью**. Она предоставляет удобные средства для хранения и обработки различных типов данных, что делает ее **идеальным** выбором для создания **RAG модели**.

# **Алгоритм парсинга файлов с сайта cbr.ru**

1. **Загрузка и обработка PDF файлов**

Для загрузки и обработки PDF файлов используется библиотека **pdfplumber**.Функция **extract\_text\_from\_pdf** принимает путь к PDF файлу и извлекает текст из каждой страницы, включая текст изображений и таблиц.

Для извлечения текста из таблиц используется функция **extract\_table**, которая находит таблицы на странице и извлекает их в формате списка списков строк.

1. **Очистка текста**

Функция **clean\_text** очищает текст от нежелательных символов, лишних пробелов и форматирует его для улучшения читаемости. Например, удаляются повторяющиеся знаки препинания и пробелы перед или после дефисов.

Текст извлекается из элементов страницы PDF с помощью функции **text\_extraction**, которая возвращает текст элемента страницы.

1. **Обработка изображений**

Для извлечения текста из изображений используется библиотека **pytesseract**.Функция **crop\_convert\_and\_extract\_text** обрезает изображение до указанного элемента, конвертирует его в изображение PNG и извлекает текст с помощью **pytesseract.**

1. **Проверка текста на целостность**

Функция **is\_broken\_text** проверяет текст на "сломанность", то есть на недостаточную длину или слишком много пробелов. Это позволяет исключить тексты, которые могут быть некорректно извлечены.

1. **Общий процесс парсинга**

В цикле происходит обработка каждого элемента данных. Если элемент содержит текст, то он очищается и проверяется на целостность. Если текст успешно обработан, он добавляется в список новых данных.

В случае возникновения ошибки при обработке, информация об ошибке добавляется в элемент данных, который затем добавляется в список данных с ошибками.

1. **Запись результатов**

Успешно обработанные данные разделяются на 2 категории “хороший текст” и “плохой текст”, и сохраняются в различные json файлы для дальнейшего использования.

**Преимущества решения**

* **Автоматизация** процесса извлечения информации из файлов с сайта cbr.ru.
* **Высокая точность** извлечения текста из различных типов файлов.
* Возможность **обработки** больших объемов данных **за короткое время**.

**Заключение**

Алгоритм парсинга файлов с сайта **cbr.ru** позволяет эффективно извлекать и обрабатывать информацию, содержащуюся в документах, распространяемых Центральным Банком России. Это обеспечивает доступ к актуальным данным для анализа финансовой информации и статистики, что является важным инструментом для принятия решений в области экономики и финансов.

# **API**

**Описание**

API представляет собой интерфейс для общения с чатботом и управления базой данных **ClickHouse**. Позволяет отправлять сообщения чатботу, получать ответы, добавлять новые записи в базу данных.

**Эндпоинты**

* **POST /chat-bot**

**Описание**: Новое сообщение для чатбота.

**Параметры**:

**chat: ChatInfo** - информация о чате, включая сообщения.

**k: int** - количество результатов поиска документов.

**Возвращаемое значение:** Ответ чатбота на сообщение и список документов, найденных по запросу.

* **POST /append-row**

**Описание**: Создание записи в базе данных.

**Параметры**:

**name: str** - название записи.

url: str - URL записи.

date: str - дата записи.

num: str - номер записи.

text: str - текст записи.

**Действие:** Обрабатывает и добавляет документ в базу данных.

**Плюсы решения**

* **Простота использования**: API предоставляет простой интерфейс для общения с чатботом и управления базой данных.
* **Интеграция с ClickHouse**: Использование ClickHouse для хранения данных обеспечивает высокую производительность и масштабируемость.
* **Быстрый поиск**: Возможность быстрого поиска документов по текстовому запросу.

**Заключение**

**API** представляет собой **удобный** способ взаимодействия с чатботом и управления базой данных **ClickHouse**. Его использование позволяет **эффективно** обрабатывать запросы и получать **быстрые ответы** от чатбота.

# **Ссылка на** [**GitHub**](https://github.com/POST-AIrony/PurpleHack2024)

Проект размещен на GitHub, где вы можете ознакомиться с исходным кодом, просмотреть последние обновления и внести свой вклад в развитие проекта.

# **Лицензия**

**MIT License**

Авторские права (c) 2024 POST AIrony

Настоящим предоставляется бесплатное разрешение, безвозмездно, каждому лицу, получившему копию этого программного обеспечения и связанных с ним файлов документации («Программное обеспечение»), чтобы обеспечить лицам, которым предоставляется программное обеспечение, разрешение на беспрепятственное владение, использование, копирование, изменение, объединение, публикацию, распространение, сублицензирование и/или продажу копий программного обеспечения, а также лицам, которым предоставляется программное обеспечение, условия на следующих условиях:

Вышеуказанное уведомление об авторском праве и это разрешение должны быть включены во все копии или значимые части программного обеспечения.

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДОСТАВЛЯЕТСЯ «КАК ЕСТЬ», БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, ВЫРАЖЕННЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ, НО, НЕ ОГРАНИЧИВАЯСЬ, ПОДРАЗУМЕВАЕМЫЕ ГАРАНТИИ ТОВАРНОГО СОСТОЯНИЯ, ПРИГОДНОСТИ ДЛЯ ОПРЕДЕЛЕННОЙ ЦЕЛИ ИЛИ ОТСУТСТВИЯ НАРУШЕНИЙ.

НИ В КОЕМ СЛУЧАЕ АВТОРЫ ИЛИ АВТОРСКИЕ ПРАВА НА ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ НЕ НЕСУТ ОТВЕТСТВЕННОСТИ ПО ДОГОВОРНЫМ ОБЯЗАТЕЛЬСТВАМ, В ОТНОШЕНИИ ДЕЙСТВИЯ, ДЕФЕКТОВ ИЛИ ИНЫХ ОБЯЗАТЕЛЬСТВ, БУДЬ ТО В ДОПУСТИМОСТИ, ИЛИ ИНЫМ ОБРАЗОМ, ВОЗНИКШИМ ИЗ, ИЛИ В СВЯЗИ С ПРОГРАММНЫМ ОБЕСПЕЧЕНИЕМ ИЛИ ИСПОЛЬЗОВАНИЕМ ИЛИ ИНЫМИ ДЕЙСТВИЯМИ В ПРОГРАММНОМ ОБЕСПЕЧЕНИИ.

# **Контакты**

[Чесников Леонид](https://t.me/FatherKomm) (Ai-Engineer & Project-manager)

[Овчинников Алексей](https://t.me/allelleo) (Backend)  
[Тронев Алексей](https://t.me/slund3r) (Design)

[Сорокина Александра](https://t.me/saha_sorokina) (Frontend)

[Борис Карабут](https://t.me/KorobkaBoris) (Frontend)