Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра автоматизированных систем управления (АСУ)

К ЗАЩИТЕ ДОПУСТИТЬ

Зав. кафедрой АСУ

канд. техн. наук, доцент

В.В. Романенко

(подпись)

« » 202*5* г.

(дата)

**СИСТЕМА ИНТЕЛЛЕКТУАЛЬНОГО ПОИСКА В КОРПОРАТИВНЫХ БАЗАХ ЗНАНИЙ**

Бакалаврская работа

По направлению подготовки 09.03.01

«Информатика и вычислительная техника»

|  |  |
| --- | --- |
|  | Выполнил: студент гр. 431-3  Е.П.Бекиш  (подпись) (И.О. Фамилия)  «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_ 202*5* г.  (дата) |
|  | Руководитель:  доцент АСУ ТУСУР, к.т.н.  (должность, ученая степень, звание)  А.Я. Суханов  (подпись) (И.О. Фамилия)  «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_ 202*5* г.  (дата) |

Томск 2025

Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

**ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)**

Кафедра автоматизированных систем управления (АСУ)

УТВЕРЖДАЮ

Зав. кафедрой АСУ

канд. техн. наук, доцент

В.В. Романенко

(подпись)

« » 202*5* г.

(дата)

ЗАДАНИЕ

на бакалаврскую работу

студенту гр 431-3 факультета систем управления

Бекиш Егору Павловичу

(Ф.И.О. студента)

1. Тема бакалаврской работы (БР): Система интеллектуального поиска в

корпоративных базах знаний

(утверждена приказом по вузу от « » мая 202*5*г. № 2783 ст).

2. Срок сдачи студентом законченной БР: « 28 » июня 202*5*г.

3. Исходные данные к работе:

3.1. ОС ТУСУР 01-2021.

3.2.

4. Содержание расчетно-пояснительной записки / перечень подлежащих разработке вопросов:

4.1. Анализ требований.

4.2. Определение спецификаций.

4.3. Проектирование.

4.4. Кодирование.

4.5. Тестирование.

5. Перечень графического материала (с точным указанием обязательных листов презентации):

5.1. Постановка цели и задачи.

5.2. Обзор предметной области.

5.3. Функциональные требования.

5.4. Обзор аналогов.

5.5. Стек технологий.

5.6. Проектирование.

5.7. Реализация.

5.8. Демонстрация.

5.9. Тестирование.

6. Дата выдачи задания: « 10 » мая 202*5*г.

Руководитель бакалаврской работы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| доцент АСУ ТУСУР, к.т.н.  (должность, ученая степень, звание) | | (подпись) | | А.Я. Суханов  (И.О. Фамилия) | |
| Задание принял к исполнению: « 10 » мая 202*5*г. | | | | |  |
| Студент гр. 431-3 | (подпись) | | Бекиш Е.П.  (Ф.И.О.) | | |

**Реферат**

Бакалаврская работа содержит N страниц, N рисунков, N таблиц, N источников, N приложений.

ЯЗЫКОВЫЕ МОДЕЛИ, МАШИННОЕ ОБУЧЕНИЕ, АНАЛИЗ ДАННЫХ, ЭМБЕДДИНГ, …

Объект разработки: представляет собой использование языковых моделей.

Предмет разработки: методы и технологии внедрения языковых моделей в корпоративных базах знаний, представленной в виде списка документов формата txt, pdf, docx, doc, pptx, ppt и реляционной база данных.

Цель работы: разработать систему, которая обеспечивает по заданному запросу сотрудника выдавать быстрый и корректный ответ, используя внутреннюю базу знаний, представленную в виде списка документов формата txt, pdf, docx, doc, pptx, ppt и реляционной база данных, для повышения эффективности работы с данными.

Полученный результат работы: внедрение системы в корпорации различного рода деятельности для повышения эффективности работоспособности сотрудников.

Область применения разработки: система предназначена для корпораций с объемным электронным документооборотом.

Бакалаврская работа выполнена в текстовом редакторе – Microsoft Word и представлена в электронной версии в электронной образовательной среде ТУСУРа.

**Abstract**

The bachelor's thesis contains N pages, N figures, N tables, N sources, N appendices.

LANGUAGE MODELS, MACHINE LEARNING, DATA ANALYSIS, EMBEDDING, …

Object of development: represents the use of language models.

Subject of development: methods and technologies for implementing language models in corporate knowledge bases, presented as a list of documents in the txt, pdf, docx, doc, pptx, ppt and relational database formats.

The purpose of the work: to develop a system that ensures that an employee's request is answered quickly and correctly, using an internal knowledge base presented in the form of a list of documents in the txt, pdf, docx, doc, pptx, ppt format and a relational database to improve data efficiency.

The result of the work: the implementation of the system in corporations of various types of activities to improve the efficiency of employees.

Scope of the development: the system is designed for corporations with extensive electronic document management.

The bachelor's thesis was completed in a Microsoft Word text editor and presented in an electronic version in the electronic educational environment of TUSUR.

**Оглавление**

[Введение 8](#_Toc199865894)

[1 АНАЛИЗ ТРЕБОВАНИЙ 10](#_Toc199865895)

[1.1 Проблематика 10](#_Toc199865896)

[1.2 Развитие больших языковых моделей 10](#_Toc199865897)

[1.2.1 Появление обработки естественного языка 10](#_Toc199865898)

[1.2.2 Развитие векторного представления слов 11](#_Toc199865899)

[1.2.3 Выборка необходимых документов и данных 24](#_Toc199865900)

[1.3 Архитектуры нейронных сетей 27](#_Toc199865901)

[1.3.1 Сверточная нейронная сеть 27](#_Toc199865902)

[1.3.2 Рекуррентная нейронная сеть 28](#_Toc199865903)

[1.3.3 Графовая нейронная сеть 28](#_Toc199865904)

[1.3.4 Трансформеры 29](#_Toc199865905)

[1.4 Формулирование требований 30](#_Toc199865906)

[1.5 Обзор аналогов 31](#_Toc199865907)

[2 ФУНКЦИОНАЛЬНЫЕ ВОЗМОЖНОСТИ И СПЕЦИФИКАЦИИ 34](#_Toc199865908)

[2.1 Выбор модели 34](#_Toc199865909)

[2.2 Выбор инструментов реализации 35](#_Toc199865910)

[2.3 Определение входных и выходных данных 37](#_Toc199865911)

[2.3.1 Входные данные 37](#_Toc199865912)

[2.3.2 Выходные данные 37](#_Toc199865913)

[3 ПРОЕКТИРОВАНИЕ 38](#_Toc199865914)

[3.1 Диаграмма прецендентов 38](#_Toc199865915)

[3.2 Диаграмма последовательности 40](#_Toc199865916)

[3.3 Макет интерфейса 41](#_Toc199865917)

[3.4 Диаграмма классов 42](#_Toc199865918)

[4 КОДИРОВАНИЕ 43](#_Toc199865919)

[4.1 Конфигурация инструментов разработки 43](#_Toc199865920)

[4.2 Программная реализация 46](#_Toc199865921)

[4.3 Руководство администратора 47](#_Toc199865922)

[4.4 Руководство сотрудника 48](#_Toc199865923)

[5 ТЕСТИРОВАНИЕ 50](#_Toc199865924)

[Заключение 51](#_Toc199865925)

[Список используемых источников 52](#_Toc199865926)

[Приложение А (справочное) Диаграмма прецедентов 53](#_Toc199865927)

[Приложение Б (справочное) Диаграмма последовательности 54](#_Toc199865928)

[Приложение В (справочное) Макеты интерфейса 56](#_Toc199865929)

[Приложение Г (справочное) Диаграмма классов 58](#_Toc199865930)

[Приложение Д (справочное) Конфигурация системы 59](#_Toc199865931)

[Приложение Е (справочное) Реализация системы 61](#_Toc199865932)

[Приложение Ж (справочное) Тестирование системы 63](#_Toc199865933)

[Приложение З (справочное) Список научных трудов по теме ВКР 64](#_Toc199865934)

# **Введение**

Языковые модели [] являются важной и активно развивающейся областью, которая за последние несколько лет привлекает большое внимание промышленников, научных и академических кругов.

В современном мире объемы информации с каждым днем растут все больше и больше, поэтому анализировать информацию становится труднее, как и выбрать более корректную. В организациях существуют собственные база знаний, которые имеют тоже не маленький объем. Поэтому, чтобы повысить собственное качество и работоспособность им бы хотелось избегать анализа лишнего материала, т.к. на нее тратится большое время.

Объект разработки представляет собой языковые модели. Предметом разработки является использование методов и технологий языковых моделей в корпоративных базах знаний.

В связи с этим, в качестве основной задачи выступает создание системы, которая позволяет сотруднику искать ту информацию, которая необходима и релевантна. Осуществляться будет за счет подключения и внедрения разработанной системы, которая будет необходимые данные в своей базе знаний.

Целью работы является сокращение времени доступа сотрудников к корпоративной информации за счет обработки запросов к внутренним документам предприятия на естественном языке и их обработка средствами ИИ.

Для достижения поставленной цели необходимо выполнить следующие задачи:

1. Анализ необходимых требований;
2. Реализация функциональных возможностей и выработка спецификаций;
3. Проектирование;
4. Реализация системы;
5. Тестирование.

Результатом выполненной работы является система, которая находит нужную информацию на поставленный запрос. Система представляет собой чат, в который пользователь может задавать вопрос, а сотрудник компании подключать внутреннею базу знаний и искать информацию в ней.

# **АНАЛИЗ ТРЕБОВАНИЙ**

## **Проблематика**

За последние несколько десятилетий объемы информации очень увеличились, что повлекло за собой очень детальный анализ этого входного потока, что очень замедляет как учебный, так и рабочий процесс. В этих условиях традиционные методы работы с информацией и базами знаний становятся неэффективными и требуют значительных ресурсов для поддержания актуальности и доступности информации. Языковые модели (LLM), такие как ChatGPT, предлагают инновационные решения для автоматизации обработки и анализа данных, улучшения поиска и доступа к знаниям для пользователей и сотрудников компаний, а также поддержки принятия решений. Однако, несмотря на очевидные преимущества, многие пользователи и компании еще не полностью осознали потенциал LLM и сталкиваются с вызовами при их внедрении. Важно понять, почему интеграция LLM в собственную жизнь или корпоративную базу является ключевым фактором успеха в условиях быстро меняющегося цифрового ландшафта и как преодолеть барьеры на пути к успешной реализации таких проектов.

## **Развитие больших языковых моделей**

### **Появление обработки естественного языка**

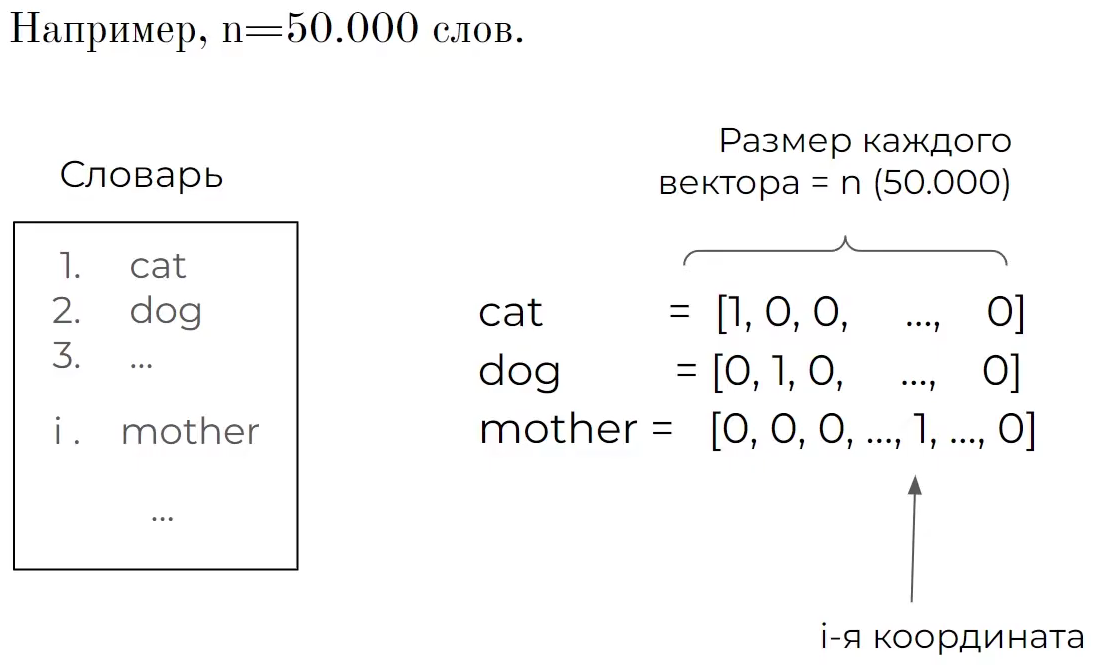
В 1950-м году Алан Тьюринг опубликовал статью "Computing Machinery and Intelligence" [1], в которой рассматривал вопрос: «Могут ли машины мыслить?», где, для начала, нужно определить термины в данном вопросе - «мыслить» и «машина». В дальнейшем выдвинул идею, что человек может взаимодействовать с машиной на естественном языке. Спустя 16 лет была опубликована первая программа по генерации текста ELIZA [2], которая имитировала психотерапевта, используя шаблоны и правила.

В 70 – 80-х годах развивался метод машинного обучения на базе правил [3], где данные правила задавались вручную, но системы на тот момент не поддерживали масштабируемость.

### **1.2.2 Развитие векторного представления слов**

Векторное представление слов – вектор в пространстве с фиксированной размерностью. На вход подаются коллекция документов, на выходе получаем векторное представление из коллекции документов.

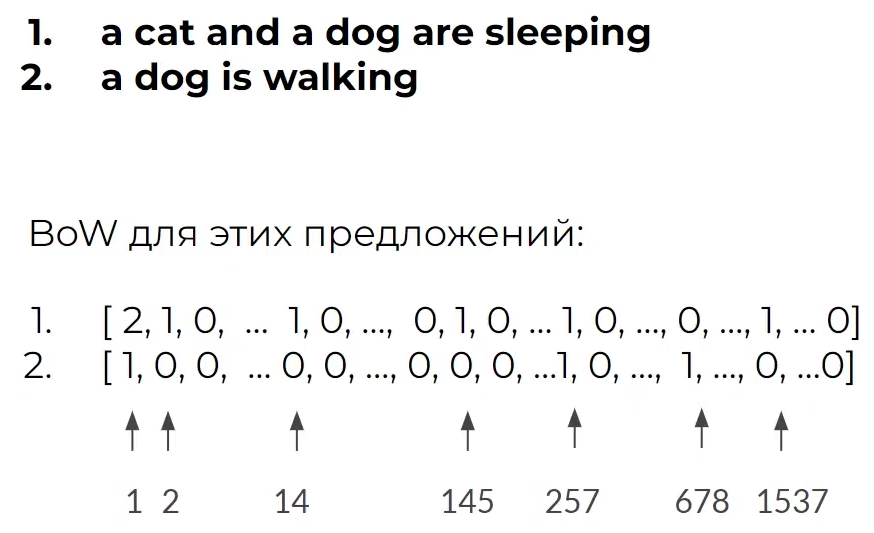
One-hot encoding – метод представления в векторном виде. Создается словарь фиксированного размера *N*, где для каждого слова из словаря соответствует вектор размера *N*, каждый вектор состоит из 1 единицы на -м месте и (*N – 1*) нулей, где – номер слова в словаре. Таким образом, запись данного метода представлена на рисунке 1.1.

Рисунок 1.1 – One-hot encoding

В приведённом описание и примере можно сделать вывод, что данный метод обладает множеством недостатков:

* 1. Размерность векторов может быть довольно большой;
  2. Слова, не попавшие в словарь, не будут обработаны;
  3. Векторы слов не отражают смысл слов;
  4. Семантическая значимость слов между собой отсутствует.

Bag to Words (BoG) – Мешок слов, представление предложений, основанный на методе one-hot encoding. В данном случае вектором предложений является сумма векторов слов. Или же можно сказать, что мешок слов для предложения – это вектор, у которого длина равна длине словаря и на -м месте стоит такое число сколько раз -е слово из словаря встречается в данном предложении. Запись данного метода можно увидеть на рисунке 1.2.

Рисунок 1.2 – Bag to Words

Следовательно, данный метод наследуется от предыдущего, будет иметь такие же недостатки, но только уже для предложений:

1. Предложения, не попавшие в словарь, не будут обработаны;
2. Векторы предложений не отражают смысл слов;
3. Семантическая значимость предложений между собой отсутствует.

И для того, чтобы перейти к следующему методу, стоит обратить внимание на еще один недостаток – различные слова в любом тексте имеют разную важность/вес для передачи контекста и смысла этого текста. Например, любой текст на английском языке имеет множество артиклей, предлогов, которые в свою очередь не несут никакого смыслового контекста, т.к. мы все же сможем понять без них, о чем идет речь. На рисунке 1.3 красным выделено, какие слова в каждом предложение имеют высокую важность для понимания, зеленным те, которые не имеют высокой важности.

Рисунок 1.3 – Важность слов в предложениях

Тогда перейдем к методу, который учитывает данную важность и будет упомянут RAG. TF-IDF – вычисляет важность каждого слова в документе относительно количества его употреблений в данном документе и во всей коллекции текстов. Этот метод позволяет выделить ключевые слова и понять, какие слова имеют больший вес для определенного документа в контексте всей коллекции. Или же более простыми словами – мера важности слова для документа среди документов .

Алгебраическая форма записи выгляди следующим образом:

(1.1)

где – входное слово;

– текущий документ;

– коллекция документов;

– относительная частота встречаемого слова в документе;

– обратная частота встречаемого слова в наборе документов.

Относительная частота рассчитывается следующим образом:

(1.2)

где – входное слово;

– документ, в котором ищется слово ;

– количество слова в документе ;

– количество слов в документе .

Обратная частота рассчитывается следующим образом:

(1.3)

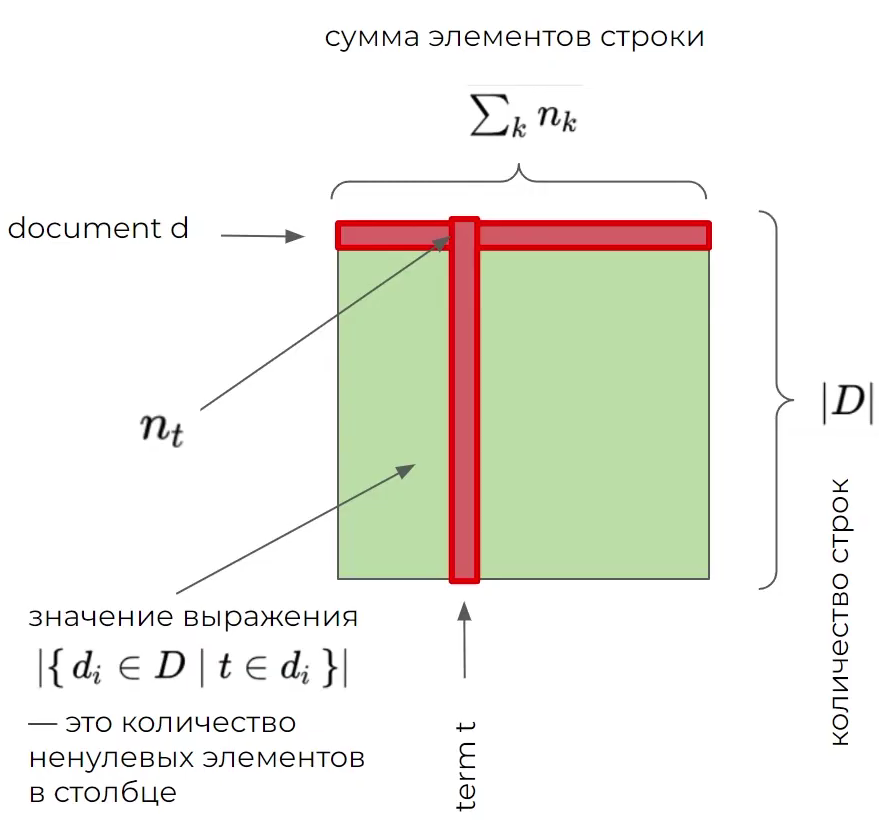
где – входное слово;

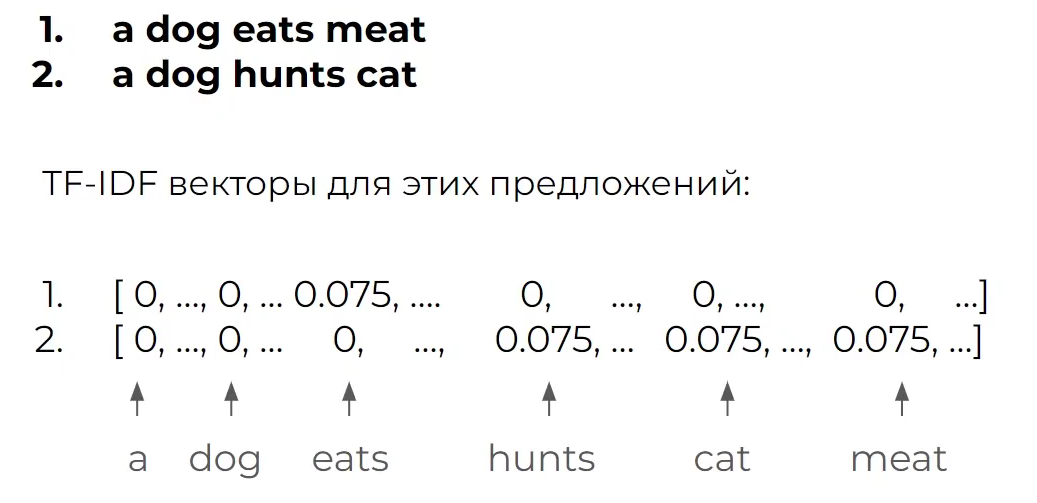
– коллекция документов, их количество;

– количество документов, где слово встречается.

Следовательно, значение обратной частоты показывает, чем оно больше, т.е. количество слов встречается реже во всех документах, тем оно является более важным по смыслу в текущем документе, в противном случае, как было сказано про артикли и предлоги до этого, их количество велико, что говорит нам о том, что их важность очень мала и не несет никакой смысловой нагрузки.

Теперь, когда можно найти важность каждого слова среди документов, строятся векторы документов из данных значений, который имеет размер словаря *N* и на -й позиции вектора будет стоять значение -го слова данного документа. Для наглядности пример можно увидеть на рисунках 1.4 – 1.5.

Рисунок 1.4 – Матрица TF-IDF

Рисунок 1.5 – Векторное представление TF-IDF

Тем самым, данный метод добавляет важные характеристики:

1. Векторы документов приобретают смысл;
2. Ранжирование и выделение ключевых слов в документах.

Среди недостатков можно выделить следующие:

1. Размерность словаря фиксирована;
2. Размерность векторов могут быть довольно большой;
3. При изменении документов требуется перерасчет значений.

На основе предыдущего метода была разработана модификация TF – IDF – Best Maching 25 (BM25) – полнотекстовый поиск заключается в нахождении *наиболее релевантных* запросу документов из множества вариантов.

Алгебраическая форма записи выгляди следующим образом:

|  |  |
| --- | --- |
|  | (1.4) |

где: – документ;

– запрос;

– обратная частота документа;

– частота слова в документе.

Часто слова в документе выглядит следующем образом:

|  |  |
| --- | --- |
|  | (1.5) |

где – запрос;

– документ;

– частота запроса в документе;

– свободные коэффициенты (, , );

– длина документа;

– средняя длина документа в нашей коллекции.

Обратная частота документа выгляди следующим образом:

|  |  |
| --- | --- |
|  | (1.6) |

где – общее количество элементов в коллекции;

– количество документов, содержащих элемент запроса;

– это количество документов, не содержащих элемент запроса.

Сам алгоритм BM25, как и было выше сказано основан на принципе вероятностного ранжирования – если извлечённые документы упорядочены по уменьшению вероятности релевантности данных, то эффективность системы оптимальна для этих данных.

К сожалению, вычислить истинную вероятность релевантности документа запросу почти невозможно.

Поэтому, дошли до того, что можно упростить уравнение за счет того, что важен порядок документов, точнее вес документа, а не точная вероятность. Можно убрать те члены уравнения, которые были необходимы для вычислений вероятности там, где их важность ничего не изменит с учетом существующего порядка.

Уравнение вычисляет вес на основании частотности элементов:

|  |  |
| --- | --- |
|  | (1.7) |

где – это вес рассматриваемого документа;

– это вероятность того, что элемент запроса встретится в документе с заданной частотой , если документ релевантен

Получаем, что уравнение сводится к тому, что можно увидеть частотность элементов запроса в документе, если документ релевантен или не релевантен, и вероятности того, что элемент совсем не встретится, если документ релевантен или не релевантен.

Следовательно, данный модифицированный метод добавляет нам то, что теперь можно выбирать ключевую информацию из релевантных документов, что в разы повышает точность ответа.

С основными методами нахождения ранжированных слов, предложений и документов разобрались. Теперь необходимо как-то сформировать верное сформулированное и понятное по смыслу предложение. Одна из моделей – N-граммы – это последовательность из N подряд идущих символов, звуков, слов, слогов и т.п. в тексте. Данная модель может быть:

1. Униграммой (1-граммы) – отдельные символы, слова и т.п.;
2. Биграммой (2-граммы) – пара слов;
3. Триграммы (3-граммы) – тройка слов.

Статическая модель, предсказывающая или предугадывающая следующее слово, основанное на вероятностной модели. Рассчитывается по цепному правилу:

|  |  |
| --- | --- |
|  | (1.8) |

где – слово в тексте;

– общий случай N-граммы.

Как можем заметить, что вероятность слова зависит от предыдущего , а не от всего предыдущего текста, что было до этого, тогда общий случай выглядит следующим образом:

|  |  |
| --- | --- |
|  | (1.9) |

Возникает проблема нулевых вероятностей, говорит о том, что если n-грамма не была в обучающихся данных, то вероятность равно нулю и модель не работает. Существует аддитивное сглаживание Лапсала, которое исправляет данную проблему. Алгебраическая запись выглядит следующим образом:

|  |  |
| --- | --- |
|  | (1.10) |

где – размер словаря.

Как и было сказано, мы можем предугадывать следующее слова, учитывая только предыдущее слово, но нам необходимо учитывать большую часть контекста, как это показано на рисунке 1.5. Вместо пропусков мы можем поставить слова по смыслу, приведенные в таблице, которые отражают, где они могут использоваться. Например, возьмем из таблицы слово «лошадь» и посмотрим, как учитывается контекст из большего количества слов, а не только предыдущего. В первом предложение для нас ориентиром будет слово «ездит», которое точном нам говорит о том, что «лошадь» обладает таким свойством. Для второго и третьего предложения ситуацию совсем другая, слова: «колесо», «проколото», «рама» не относятся к свойствам «лошади».

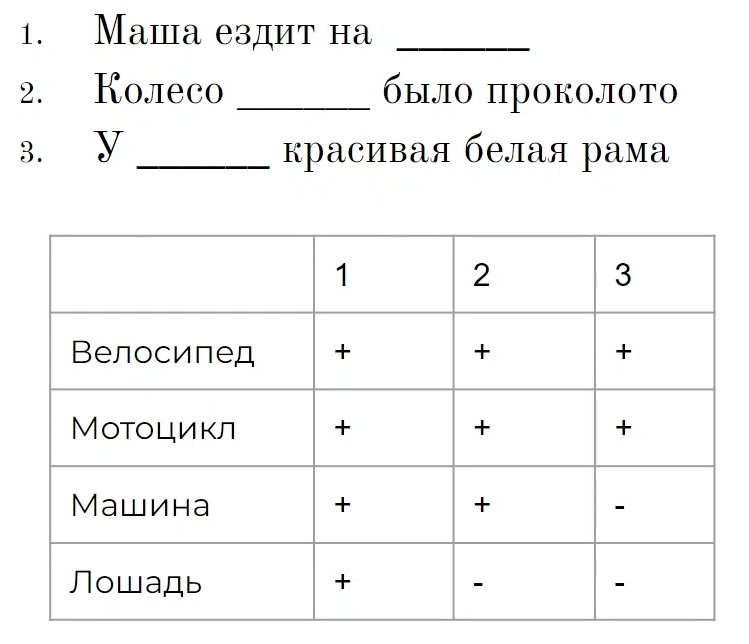


Рисунок 1.5 – Контекстный смысл слов

Следовательно, появляются следующие идеи:

1. По контексту можно понять какое слово может стоять на месте пропуска, а какое нет, т.е. слово должно обладать определенным соответствующим свойством;
2. Смысл слова можно определить через контекст, в котором оно может использоваться, т.е. берется слово, смысл которого неизвестен, но известен контекст, тогда можно определить какими свойствами оно обладает.

Для этого построим матрицу контекстных векторов слов, как показано на рисунке 1.6. На пересечении строк и столбцов находится количество встреч -го слова в контексте с -м.

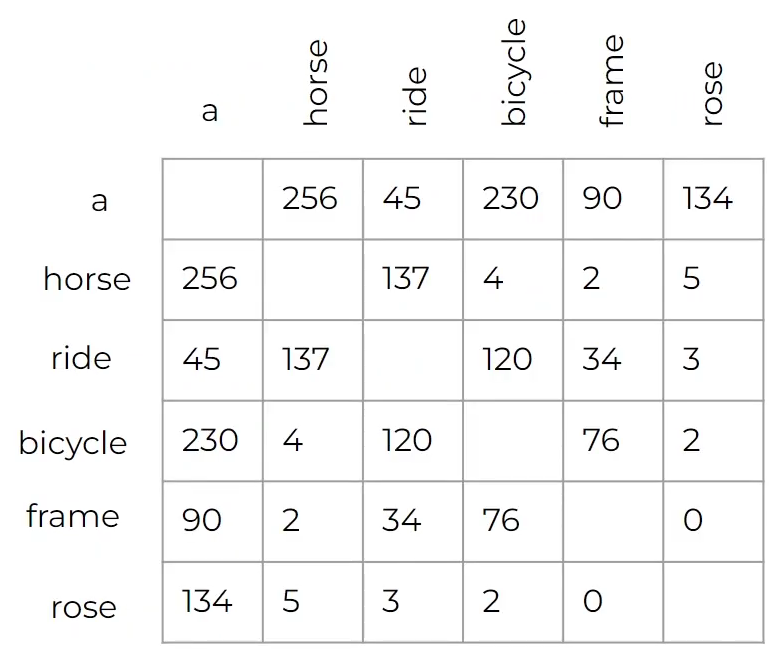


Рисунок 1.6 – Матрица контекстных векторов слов

Получается следующим образом. Из нашего набора последовательно берется пять слов, из них выбирается центральное и два слова до и после него – контекст центрального слова. Дальше сдвигаемся на одно слово и повторяем данный алгоритм. Это можно увидеть на рисунках 1.7 – 1.8.

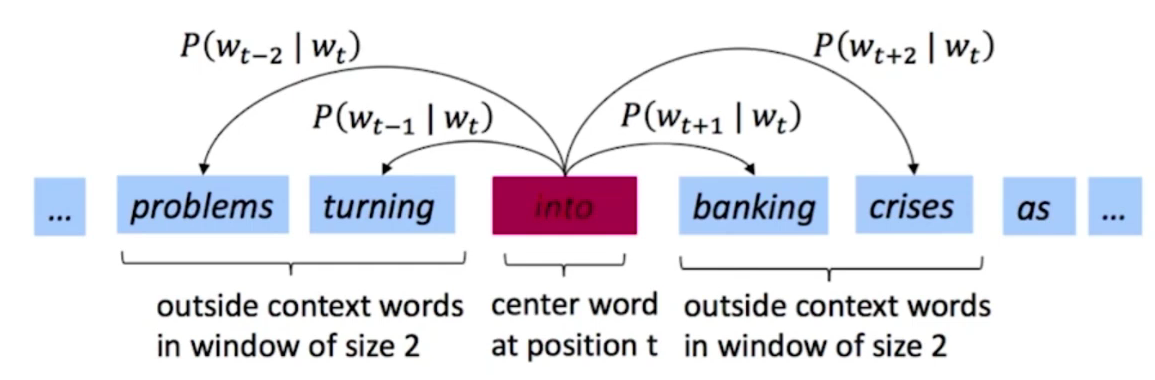
**

Рисунок 1.7 – Схематическое изображение алгоритма при центральном слове into

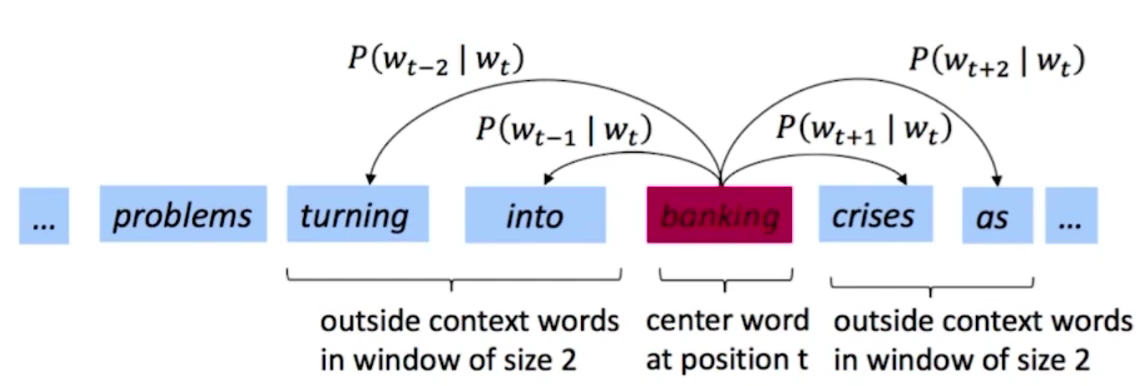
**

Рисунок 1.8 – Схематическое изображение алгоритма при центральном слове banking

Основная проблема, которая остается до сих пор – размерность. Для этого можно воспользоваться SVD разложением, показано на рисунке 1.9.

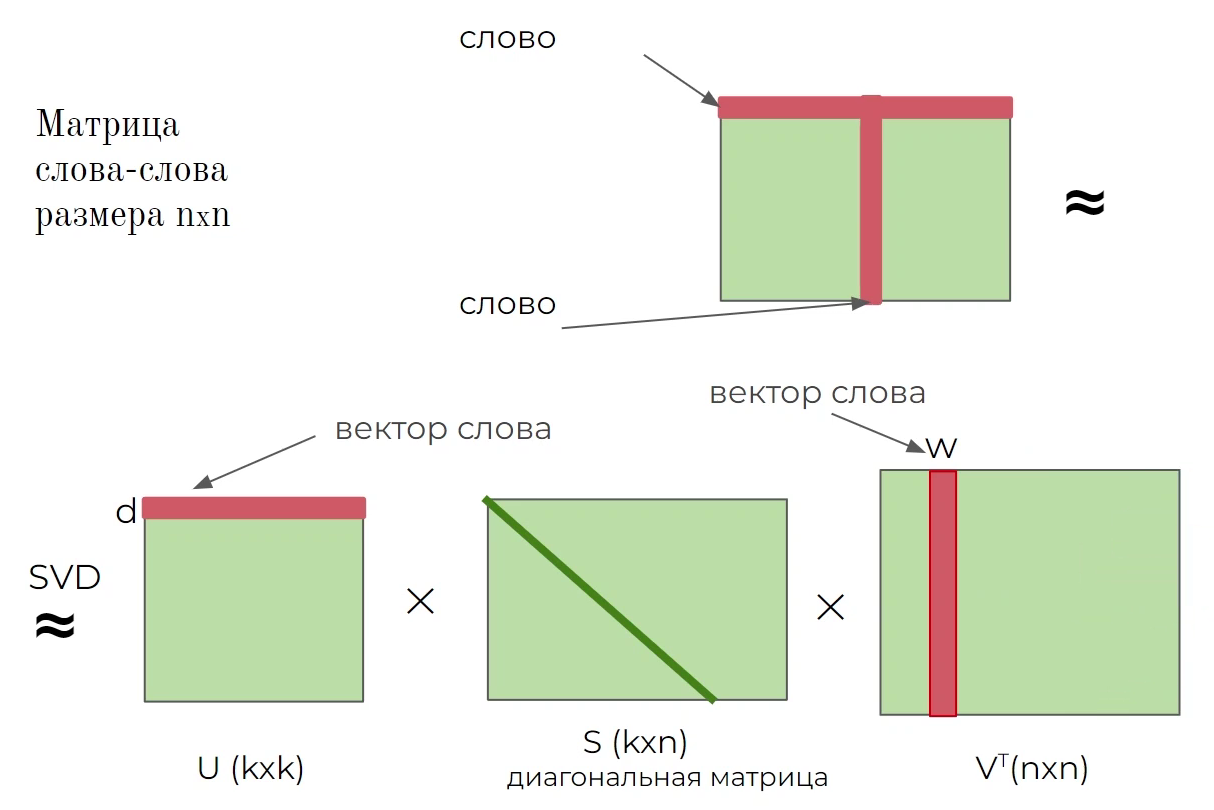


Рисунок 1.9 – Использование SVD разложения

Тем самым, к данному разложению можно применить латентный семантический анализ. Позволяет уменьшить размерность, что дает естественный прирост в скорости обработки слов, не потеряв их смысл. Данный метод представлен на рисунке 1.10.

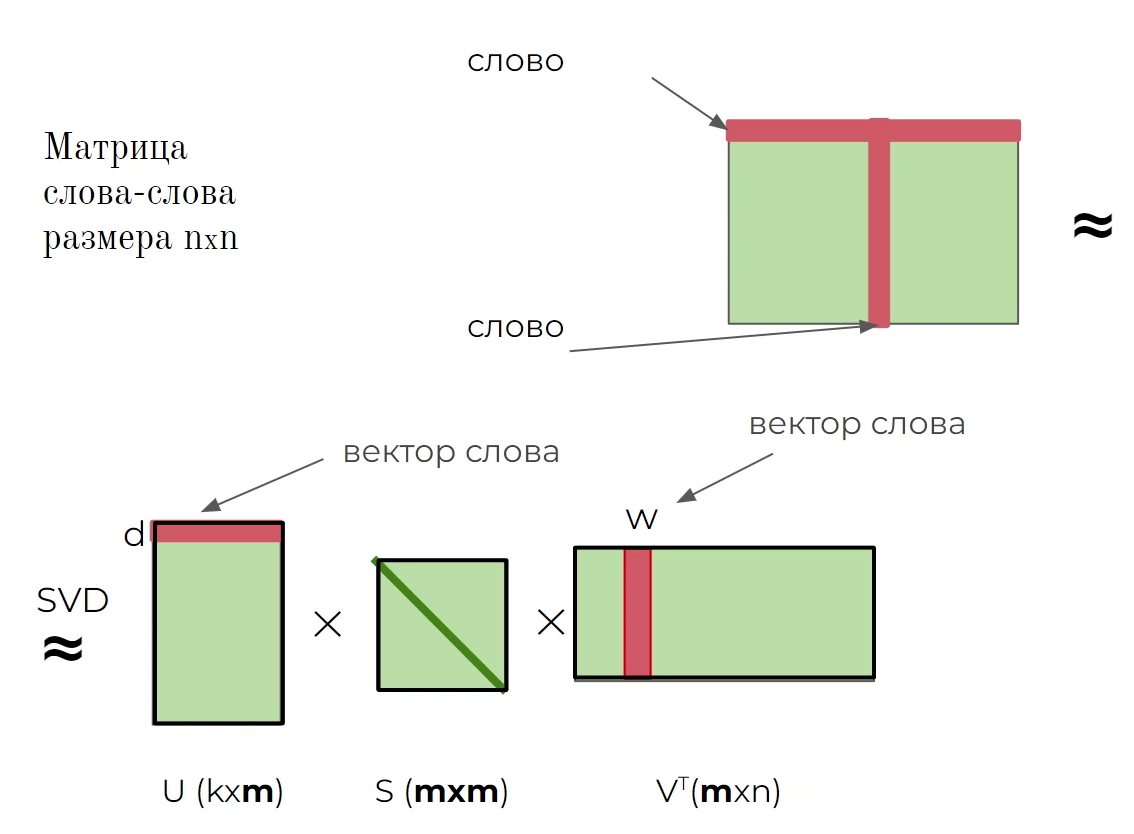


Рисунок 1.10 – Латентный семантический анализ

На основе выше приведенных методов и идей в 2013 году с появлением глубокого обучения появилась модель Word2Vec. Она имеет другое название более распространённое – эмбеддинг. Используется нейросеть, которая обучается на данных словах и позволяет сравнивать их смысл, контекст с помощью некоторой метрики, в основном применяется косинусное расстояние. Алгебраический вид, следующий:

|  |  |
| --- | --- |
|  | (1.11) |

где и – векторы.

Тогда, такие обученные слова в векторном виде будут называться эмбеддингом слов.

Так же необходимо учитывать контекст, поэтому воспользуемся алгоритмом выше, представленный на рисунках 1.7 – 1.8, когда у нас присутствует центральное слово и контекстом является два слова до и после него. На этих данных нейронная сеть будет учиться.

Вид данной нейросети будет показан на рисунке 1.11.

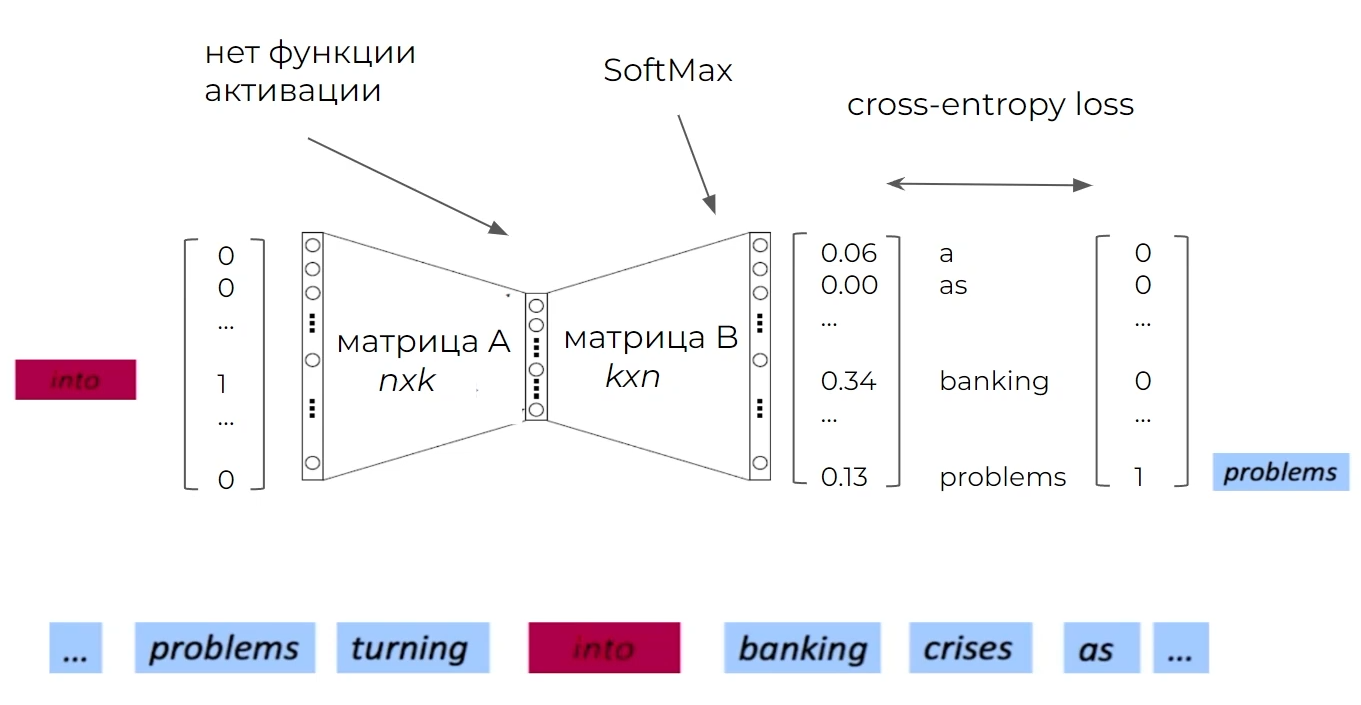
**

Рисунок 1.11 – Нейросеть модели Word2Vec

Стоит отметить одну из важных вещей, которая отображена на рисунке 1.11. Это SoftMax – многопеременная логическая функция, которая обобщает логическую регрессию для многомерных случаев. В нашей модели на выходе мы получаем вектор, который отображает для наших слов, они и будут является многопеременными, распределение, где большие вероятности у тех слов, которые могут стоять в контексте с центральным словом, а с маленькой вероятностью у тех, которые не могут встречать рядом.

И в конце, после обучения, мы получаем вектор для всех слов в словаре, где каждый вектор отражает смысл слов. Их можно сравнить между собой с помощью косинусного расстояния.

### **Выборка необходимых документов и данных**

На данный момент, большинство общедоступных LLM при создании промпта ищут информацию в открытых интернет-источниках, но при поставленной цели и задачи необходимо искать среди тех данных, которые были предоставлены модели. Такой метод называется Retrieval Augmented Generation (RAG) – это метод работы с большими языковыми моделями, когда пользователь пишет свой промпт, а система программно к этому вопросу «подмешивает» дополнительную информацию из каких‑то внешних источников и подаете все целиком на вход языковой модели. Другими словами, добавление в контекст запроса к языковой модели дополнительную информацию, на основе которой языковая модель может дать пользователю более полный и точный ответ.

В текущем случае выборка будет происходить полностью из тех документов, которые находятся в базе данных корпорации.

На рисунке 1.12 отображена концепция RAG.

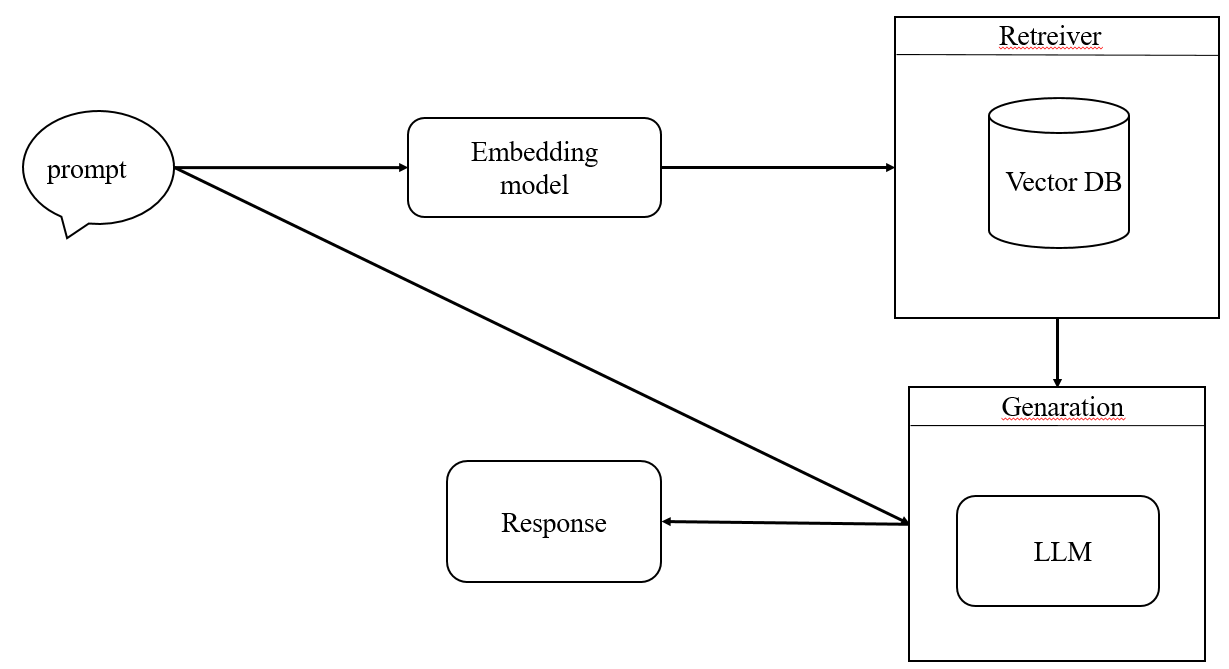


Рисунок 1.12 – концепция RAG

Простыми слова, сотрудник вводит запрос, система его обрабатывает, ищет в корпоративной базе знаний соответствующий документ, извлекает необходимую информацию, генерирующейся через языковую модель, и в конце возвращает ответ на поставленный запрос.

При внедрении данного метода могут возникнуть следующие проблемы:

* 1. Нечеткий поиск – просто взять запрос и найти по точному соответствию все куски из базы знаний не получится;
  2. Размер данных документа из базы знаний – неизвестен размер «кусков» текста, который обрабатывает LLM;
  3. На заданный запрос может найти несколько релевантных документов.

Размер текста в контексте больших языковых моделей называется чанками – небольшие куски текста, которые между собой объединяются и получается осмысленный ответ. Тогда выделить следующие идею:

1. Чем меньше чанк по размеру, тем точнее будет буквальный поиск, чем больше размер чанка тем больше поиск приближается к смысловому.
2. Разные запросы пользователя могут содержать разное количество чанков, которое необходимо добавлять в контекст. Необходимо опытным путем подобрать тот самый коэффициент, ниже которого чанк смысла не имеет и будет лишь замусоривать ваш контекст.
3. Чанки должны перекрывать друг друга, чтобы был шанс подать на вход последовательность чанков, которые следуют друг за другом вместе, а не просто вырванные из контекста куски.
4. Начало и конец чанка должны быть осмысленными, в идеале должны совпадать с началом и концом предложения, а лучше абзаца, чтобы вся мысль была в чанке целиком.

К определению более релевантного документа можно воспользоваться методами из раздела 1.2.2:

* 1. TF-IDF (вычисляет важность каждого слова в документе относительно количества его употреблений в данном документе и во всей коллекции текстов);
  2. BM25 (функция ранжирования, используемая поисковыми системами для упорядочивания документов по их релевантности данному поисковому запросу).

Если же документы, лежащие в базе знаний, не упорядочены между собой, то можно воспользоваться графом знаний. Граф знаний – это направленный реляционный граф, где вершины обозначают сущности, а рёбра отражают связи между ними. Такие графы строятся на основе триплетов вида (*субъект, отношение, объект*) или (*s, r, o*), что позволяет систематизировать информацию как о конкретных объектах, так и об абстрактных концепциях. Таким образом, граф знаний служит структурированной моделью для представления разнообразных фактов.

Таким образом документы можно структурировать между собой, обладая общей тематикой, что уменьшит время поиска и повысит точность. Пример такого графа представлен на рисунке 1.13.

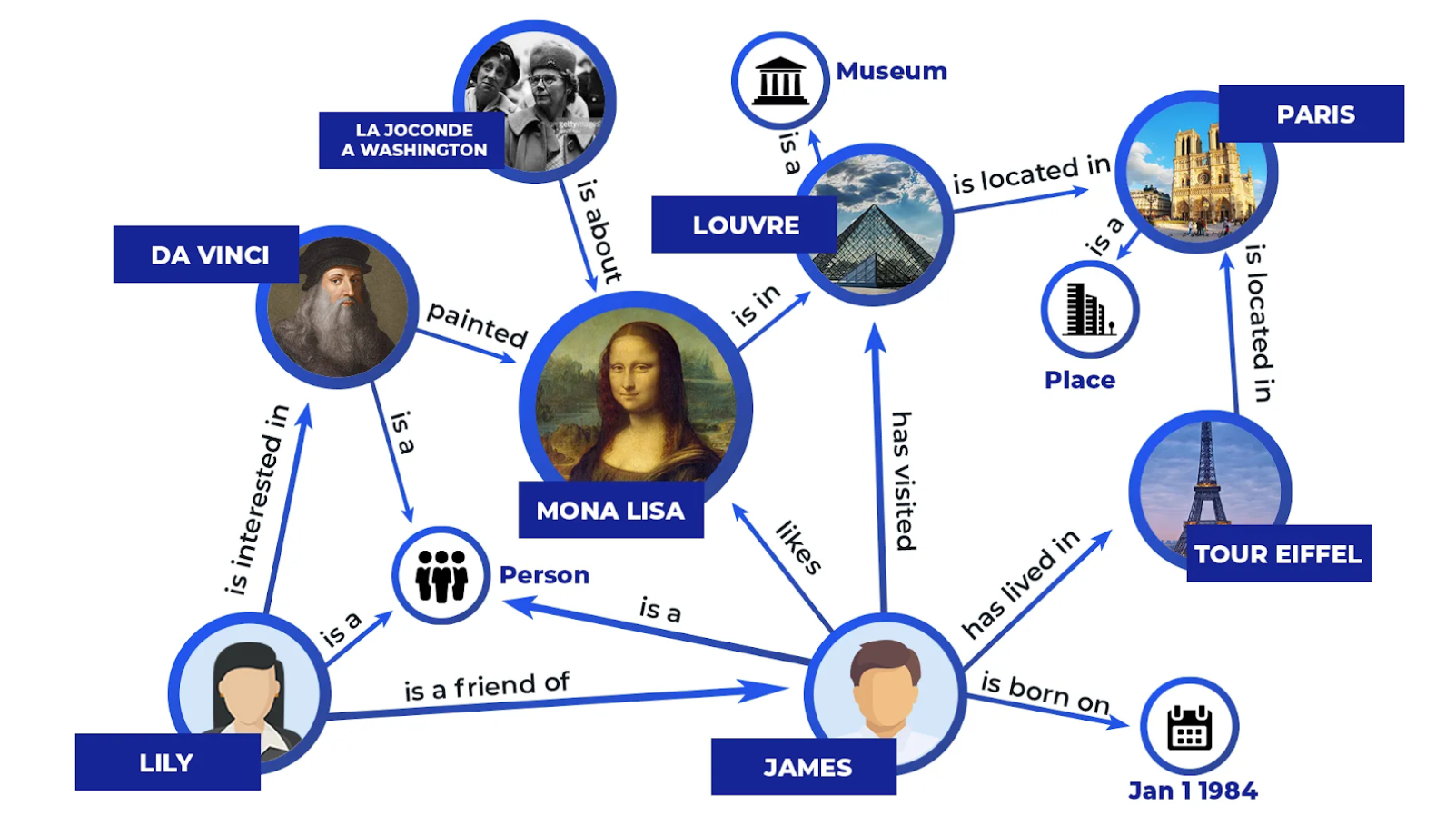


Рисунок 1.13 – Граф знаний

## **1.3 Архитектуры нейронных сетей**

### **1.3.1 Сверточная нейронная сеть**

Свёрточная нейронная сеть (CNN) [] обрабатывает данные, которые имеют многомерную структуру, например, изображения, представленные в виде нескольких слоёв. Существуют следующие типы:

1. Свёрточные слои – применяют фильтры, которые "выделяют" локальные особенности.
2. Пулинговые слои – уменьшают размерность данных, сохраняя ключевую информацию.
3. Полносвязные слои – используются в конце для классификации или регрессии, анализируя все признаки вместе.

Если на вход подаётся цветное изображение, то после первого свёрточного слоя может получиться несколько десятков карт признаков, каждая из которых активируется на разных деталях изображения.

Таким образом, CNN постепенно преобразует исходные данные в иерархию признаков.

### **1.3.2 Рекуррентная нейронная сеть**

Рекуррентные нейронные сети (RNN) [] – специализированные модели для работы с последовательностями. RNN отличаются от обычных нейросетей наличием циклических связей, которые позволяют учитывать предыдущие состояния при обработке новых данных. Это делает их идеальными для задач, где важен порядок и контекст, таких как:

* Анализ текста (перевод, генерация)
* Обработка временных рядов (прогнозирование)
* Распознавание речи

Обучение RNN:

1. Принцип работы:

* На каждом шаге RNN получает новый элемент последовательности и учитывает скрытое состояние от предыдущего шага.
* Позволяет сети помнить контекст.

1. Обратное распространение ошибки:

* Градиент вычисляется не только на основе текущего шага, но и с учетом всех предыдущих.
* Например, чтобы обучить сеть правильно предсказывать 4-е слово в предложении, нужно учесть ошибки от первых трёх слов.
* Из-за этого градиенты могут быть маленькими или большими, что усложняет обучение.

### **1.3.3 Графовая нейронная сеть**

Графовые нейронные сети (GNN) [] – работа с данными в их естественной структуре. GNN – это тип нейронных сетей, предназначенный для прямой работы с графами без необходимости преобразовывать их в таблицы или последовательности. Это позволяет сохранить связи и структуру графа, что особенно важно для задач, где отношения между объектами играют ключевую роль.

Обучение RNN:

* Механизм распространения информации. GNN состоит из взаимосвязанных модулей\*\*, которые соответствуют узлам графа. Эти модули обмениваются данными в соответствии со связями графа.
* Итеративное обновление состояний. В процессе обучения каждый модуль постепенно обновляет своё состояние, учитывая информацию от соседних узлов. Этот процесс повторяется до тех пор, пока система не стабилизируется. Чтобы гарантировать сходимость, механизм распространения искусственно ограничивают.
* Формирование выходных данных. После стабилизации финальное состояние каждого модуля используется для получения результата.

### **1.3.4 Трансформеры**

Трансформер [] представляет собой нейросетевую модель, состоящую из двух основных блоков: кодировщика и декодировщика.

Вот как это работает:

1. Подготовка входных данных:

* Исходная последовательность (например, текст) преобразуется в числовые векторные представления.
* К этим векторам добавляется информация о позициях элементов.
* Важно: порядок обработки элементов не имеет значения - все элементы обрабатываются одновременно.

1. Работа кодировщика:

* Полученные векторы поступают в многослойный кодирующий блок.
* Каждый слой кодировщика анализирует взаимосвязи между всеми элементами последовательности.
* В результате создаются обогащенные векторы, содержащие контекстную информацию о каждом элементе.

1. Работа декодировщика:

* Принимает два вида данных: частично сформированный результат и выход кодировщика.
* Также состоит из нескольких одинаковых слоев.
* Постепенно преобразует входные данные в конечную последовательность.

1. Финальный этап:

* Декодированные векторы преобразуются в вероятностное распределение.
* На основе этих вероятностей выбираются конечные элементы результата.

## **1.4 Формулирование требований**

Обращая внимание на предметную область, были сформулированы определенные функциональные требования к разрабатываемой системе:

* + 1. Пользователь должен иметь возможность авторизоваться для использования системы;
    2. Система должна сформировать токен, с помощью которого пользователь может пользоваться внутренней API;
    3. Пользователь должен иметь возможность формировать запрос;
    4. Элементы интерфейса системы должны включать в себя поле ввода, чтобы пользователь мог сформировать запрос;
    5. Система должна предоставлять возможность пользователю загружать необходимые файлы, если у него стоит задача найти необходимую информацию в них, если же это сотрудник, то внедрить/подключить к внутренней корпоративной базе знаний;

Система должна формировать корректный и понятный ответ пользователю.

## **1.5 Обзор аналогов**

Существует множество моделей, которые на данный момент могут удовлетворять условия актуальности и постановки задачи, но более мощные по своей точности, производительности и функционалу подходят следующие:

* + 1. IBM Watson [] – это мощная платформа искусственного интеллекта, которая включает в себя множество инструментов для обработки естественного языка, машинного обучения и аналитики данных. Она широко используется в корпоративных средах для автоматизации бизнес-процессов, улучшения клиентского обслуживания и оптимизации принятия решений. Watson может интегрироваться с корпоративной базой данных для создания чат-ботов, анализа текстов, прогнозирования и других задач, связанных с обработкой больших объемов данных.

1. Azure Cognitive Services от Microsoft [] предлагает набор API-интерфейсов для интеграции функций искусственного интеллекта в приложения и системы. Эти сервисы включают обработку текста, речи, изображений и видео. С помощью Azure Cognitive Services можно создавать интеллектуальные приложения, которые взаимодействуют с корпоративными данными, анализируют тексты, распознают речь и многое другое.
2. Google Cloud AI [] предоставляет инструменты и сервисы для разработки и развертывания моделей машинного обучения, включая Natural Language Processing (NLP). Платформа поддерживает создание и обучение собственных моделей, а также использование готовых решений. Google Cloud AI может быть использована для анализа текстов из корпоративных баз данных, автоматического ответа на запросы клиентов, классификации документов и других NLP-задач.
3. Amazon Lex [] — это сервис для создания голосовых и текстовых интерфейсов на основе технологии Alexa. Он позволяет разрабатывать чат-боты и другие интерактивные интерфейсы, которые могут взаимодействовать с пользователями через текстовые сообщения или голосовые команды. Lex может быть интегрирован с корпоративной системой для автоматизации взаимодействия с клиентами, сотрудников или партнеров, обеспечивая быстрый доступ к необходимой информации.
4. OpenAI GPT [] — это семейство языковых моделей, разработанных компанией OpenAI. Они используются для генерации текста, перевода, обобщения и других сложных задач обработки естественного языка. Хотя GPT изначально был создан для открытых систем, он может быть адаптирован и использован в корпоративных приложениях для различных целей, таких как автоматическое составление отчетов, ответы на вопросы пользователей и поддержка принятия решений на основе данных.
5. Hugging Face Chat [] — это интерактивная платформа для общения с искусственным интеллектом. Она предназначена для тестирования и демонстрации возможностей различных языковых моделей, доступных на платформе Hugging Face Hub. Пользователи могут взаимодействовать с моделями прямо в браузере, задавая вопросы, проводя эксперименты и оценивая производительность моделей в реальных условиях.
6. DeepSeek [] — это крупная языковая модель (LLM). Хотя модель пока находится на стадии раннего развития и менее известна широкой публике, она вызывает значительный интерес среди специалистов в области искусственного интеллекта благодаря своему инновационному подходу к обработке информации и широкому спектру потенциальных применений.

Следует отметить, что несмотря на наличие аналогов их использование невозможно в связи с санкциями и ограничением доступа российским пользователям. Но проанализировав отечественные решения, отвечающих требованиям, была выбрана модель GigaChat от Сбербанка []. Для взаимодействия необходим доступ к API. Сбербанк предоставляет защищенное соединение для получения, чтобы данные нельзя было перехватить. С полным перечнем документов можно ознакомится в разделе GigaChat "Юридические документы GigaChat" [].

Стоит отметить про локальное использование модели. Если же компании не подходит вариант с подключением в интернет или же у нее отсутствует ключ ПКЗИ для взаимодействия с GigaChat API, то необходимо на свои сервера скачать необходимую модель для дальнейшего использования.

# **2 ФУНКЦИОНАЛЬНЫЕ ВОЗМОЖНОСТИ И СПЕЦИФИКАЦИИ**

## **2.1 Выбор модели**

За внутреннею модель разрабатываемой системы была выбрана, выше упомянута, GigaChat. Данная модель включает в себя следующие необходимые компоненты:

1. Авторизация, для получения токена, которая предоставляет API для взаимодействия с моделью;
2. Добавление собственных файлов или базы знаний;
3. Обработка большого размера текста.

На данном этапе формируется список требований к функционалу и характеристикам система, а также проводится анализ уже существующих решений. В ходе разработки были выявлены следующие требования:

1. Система должна включать в себя возможность выбрать иную языковую модель, если в ходе доработок такие буду добавлены для иных задач;
2. Система должна включать в себя возможность искать ответ не только внутри корпоративной базы знаний, но и среди открытых источников. Сотрудник может получать ответ, не обращаясь к корпоративной базе данных;
3. Система должна включать в себя возможность искать ответ в указанном файле. Сотрудник ранее работал с определенным файлом и знает, что в этом файле находится необходимая ему информация.

После того, как были определены цели был проведен анализ рисков, которые могут возникнуть при внедрении и использование системы. Выделенные следующие риски:

1. Удобство использования. Создание интуитивного, понятного интерфейса;
2. Совместимость. Обеспечить работоспособность на любую операционную систему.

Этап разработки обеспечивает реализацию функциональности на предыдущих этапах. Разработка велась поэтапно, в соответствие с предоставляемыми компонентами модели компонентами и требованиями системы, где каждый был реализован и протестирован отдельными функциями и модулями.

Тестирование производилось на каждом этапе для корректной, стабильной и полноценной работы системы.

## **2.2 Выбор инструментов реализации**

В ходе анализа, основываясь на вышеопределенной области, был выбран набор используемых инструментов, который обеспечивает эффективную разработку. Также, одним из факторов выбора – популярность, стабильность, простота использования, функциональность. Рассмотрим данный набор инструментов разработки:

1. Jupyter Notebook [] – это интерактивное веб-приложение для создания и совместного использования документов, содержащих код, визуализации данных, математические формулы и пояснительный текст. Это мощный инструмент для анализа данных, машинного обучения, численного моделирования и других задач, связанных с программированием и наукой о данных.
2. Visual Studio Code [] – это бесплатный редактор исходного кода от компании Microsoft, который стал одним из самых популярных инструментов среди разработчиков благодаря своей гибкости, расширяемости и поддержке множества языков программирования. VS Code предоставляет мощные функции для редактирования кода, отладки, управления версиями и интеграции с различными системами контроля версий и платформами разработки.
3. Streamlit [] – это библиотека для Python, которая позволяет быстро создавать веб-приложения, поддерживает различные виджеты, такие как ползунки, выпадающие списки, кнопки и чекбоксы, что позволяет пользователям взаимодействовать с приложением в реальном времени. Она идеально подходит для создания прототипов, чатов, ботов, дашбордов и интерактивных отчетов, позволяя разработчикам сосредоточиться на логике приложения, а не на фронтенде.
4. UV был выбран в качестве альтернативного менеджера зависимостей и сборщика Python-пакетов. Этот инструмент сочетает в себе высокую скорость работы, современный подход к управлению зависимостями и совместимость с экосистемой Python. UV обеспечивает быстрое разрешение зависимостей, параллельную установку пакетов и поддержку виртуальных окружений, что делает его отличным выбором для больших проектов. Кроме того, UV совместим с существующими стандартами, такими как pyproject.toml и requirements.txt, что упрощает миграцию с других инструментов. Он также поддерживает создание и публикацию пакетов, помогая разработчикам поддерживать порядок в проекте и ускорять рабочие процессы. Благодаря своей производительности и удобству [].

Выбранный список инструментов описывает функциональность, удобство, гибкость, скорость и удобство пользования. Обеспечивают полную поддержку для всех этапов разработки, начиная от настройки конфигурации до анализа и редактирования полученного ответа

## **2.3 Определение входных и выходных данных**

### **2.3.1 Входные данные**

Для полного функционирования системы, обеспечивающая ответ на запрос, были определены следующие данные, которые поступают в нее:

1. Авторизационные данные, предоставляющие токен для API модели;
2. Сформированный корректный запрос;
3. Загруженные документы или подключенная база знаний;
4. Выбор определенного документа для поиска информации при необходимости;
5. Выбор языковой модели при необходимости.

### **2.3.2 Выходные данные**

Выходные данные включают в себя ответ на сформированный запрос, который представляет собой отредактированный, отформатированный текст.

# **3 ПРОЕКТИРОВАНИЕ**

В данной работе использован системный подход к проектированию системы. Раздел включает в себя описание диаграммы прецедентов, последовательности, классов. Для структурирования и визуализации компонент системы была выбрана нотация UML (Unified Modeling Language).

Для определения и описания ключевых функций системы была построена диаграмма прецедентов. Чтобы показать, как объекты системы взаимодействуют друг с другом при выполнении основных процессов, были разработаны диаграммы последовательности.

Помимо этого, были разработаны пакеты, которые отражают структуру системы. Диаграмма классов была спроектирована для отображения данной структуры.

## **3.1 Диаграмма прецендентов**

Диаграмма прецедентов, представленная на рисунке А.1 в приложении А, описывает функциональные возможности системы, а также взаимодействие пользователя с системой. Рассмотрим каждый элемент диаграммы и его роль:

1. Создание запроса позволяет отправить запрос в систему для обработки и выдачи релевантного ответа. До этого этапа можно дополнить следующим функционалом:
2. Выбор языковой модели, предоставляет сотруднику возможность выбрать модель из списка предоставленных для обработки запроса в необходимом формате.
3. Область поиска, предоставляет сотруднику возможность выбрать вариант поиска в системе.
4. Поиск из базы знаний, предоставляет сотруднику релевантный ответ из подключенной базы знаний.
5. Поиск из открытых источников, предоставляет сотруднику релевантный ответ из источников общего доступа, интернет.
6. Добавление документа к запросу, позволяет сотруднику искать релевантный ответ из приложенного документа.
7. Загрузка документов, подключение базы знаний:
8. Просмотр базы знаний и документов, позволяет сотруднику получить список всех имеющихся документов.
9. Удаление документов, позволяет сотруднику удалять загруженные собственные документы. У сотрудника будет возможность удалить выбранные документы, так и все сразу.

В качестве основного актора выступает сотрудник, который взаимодействует с системой. Сотрудник может ввести запрос, выбрать необходимую языковую модель, по умолчанию ответ ищется в базе знаний, но можно выбрать генерацию ответа из приложенных документов или открытых источников, при необходимости можно загрузить необходимые документы, просмотреть и удалить их.

Далее были описаны основные взаимодействия, предусмотренные в данной системе:

* Сотрудник вводит запрос, после чего он может выбрать дополнительные параметры, такие как: выбрать языковую модель, источник получения.
* Сотрудник выбирает языковую модель, по умолчанию установлена GigaChat.
* Сотрудник выбирает источник поиска ответа, по умолчанию поиск происходит в подключенной базе знаний, так же существуют варианты поиска в открытых источниках и выбор документов из базы знаний и загруженных лично сотрудником.
* Сотрудник может загружать имеющиеся документы при необходимости.
* Сотрудник может просмотреть документы, находящиеся в базе знаний и загруженных самим сотрудником.
* Сотрудник может удалить собственные загруженные документы выборочно либо все.
* По завершению выбора необходимых параметров сотрудник отправляет запрос, и система возвращает релевантный ответ.

В данном подразделе была представлена и проанализирована диаграмма прецедентов системы. Диаграмма отразила основные сценарии использования, такие как создание запроса, выбор языковой модели, выбор источника поиска, добавление документов, удаление добавленных документов.

## **3.2 Диаграмма последовательности**

Диаграммы последовательности, представленные на рисунках Б.1 – Б.2 приложения Б, были спроектированы компоненты системы во временном контексте. В данном подразделе будут рассмотрены диаграммы последовательности, включающая такие объекты, как сотрудник, пользовательский интерфейс (UI), GigaChat API.

Взаимодействие начинается с попадания на главный экран системы, где происходит получение токена и документы из базы знаний и те, которые были до этого загружены сотрудником.

Далее вводится запроса. На UI отображены параметры такие как: выбор языковой модели, выбор источника поиска. При необходимости сотрудник может изменить данные параметры.

После определения параметров сотрудник отправляет запрос, UI обрабатывает и передает в GigaChat API. API формирует релевантный ответ и возвращает обратно в UI, где происходит отображение данного ответа.

Для того, чтобы взаимодействовать с документами необходимо их добавить. После можно проверить их успешную загрузку в базу. В дальнейшем, если файлы устарели, необходимо их обновить или очистить базу полностью.

Таким образом, диаграммы последовательностей демонстрируют, как сотрудник, взаимодействуя с интерфейсом, управляет процессом подготовки данных, настройки параметров, и как различные компоненты системы координируют свои действия для достижения цели.

## **3.3 Макет интерфейса**

В данном подразделе был рассмотрен макет пользовательского интерфейса, представленный на рисунках B.1 – В.4 приложения В. Одна из главных задач – организация элементов интерфейса, которая должна быть интуитивно понятна сотрудникам для взаимодействия с системой.

Интерфейс включается в себя левую боковую панель, которую можно сворачивать при необходимости, и основную центральную рабочую область. Боковая панель предназначена для перехода между страницами разделов.

В ходе разработки было принято решение разделить все страницы на три основных раздела: "Описание", "СИПвКБЗ", "Документы/база знаний". Раздел "Описание" включается в себя страницу "Главная" – страница, на которую попадает сотрудник при запуске системы, где отображено описание созданной системы. Раздел "СИПвКБЗ", который расшифровывается как "Система интеллектуального поиска в корпоративных базах знаний" включается в себя следующие страницы: "Чат", "Закрыть чат". На странице "Чат" сотрудник задает свой вопрос, и система выдает ответ. Страница "Закрыть чат" служит для окончание работы с системой. В разделе "Документы/база знаний" так же было создано две страницы: "Загрузка документов", "Просмотр/удаление документов". На странице "Загрузка документов" отображен виджет для загрузки файлов, который принимает выбранные документы сотрудника и загружает их в хранилище. И на странице "Просмотр/удаление документов" отображен список текущих документов, которые были взяты из базы знаний и личного хранилища сотрудника. Те документы, которые находятся в корпоративной базе знаний не могут быть никак изменены, а те документы, которые были загружены в личное хранилище сотрудником, могут удаляться им же при необходимости.

Цветовая схема интерфейса выбирается от параметров системы компьютера, т.е. если у сотрудника светлая тема, то и схема то же будет светлая, в противном случае – темная.

Дополнительные функциональные элементы включаются в себя подсказки. Они могут быть следующего характера:

* При наведении курсором на виджет всплывает краткая информация использования;
* На самом виджете отображена информация пользования.

Данные справки помогают сотруднику быстро ориентироваться в настройках системы. Как и было сказано выше, при переходе на страницу "Главная" будут описаны возможности корректного взаимодействия с системой.

## **3.4 Диаграмма классов**

# **4 КОДИРОВАНИЕ**

## **4.1 Конфигурация инструментов разработки**

При разработке системы была выбрана операционная система Windows. Установка необходимых инструментов, сред и пакетов в основном происходили через ссылки на официальные источники, где находятся установочные устройства. Но для более детальной настройки был установлен пакетный менеджер chocolatey, была выполнена следующая команда:

@"%SystemRoot%\System32\WindowsPowerShell\v1.0\powershell.exe" -NoProfile -InputFormat None -ExecutionPolicy Bypass -Command "iex ((New-Object System.Net.WebClient).DownloadString('https://chocolatey.org/install.ps1'))" && SET "PATH=%PATH%;%ALLUSERSPROFILE%\chocolatey\bin"

Далее, для установки Visual Studio Code с помощью chocolatey была выполнена следующая команда:

choco install vscode

Теперь, необходимо установить Python, версия при установке была последняя 3.12, была выполнена следующая команда:

choco install -y python3

Для корректной работы с Python в VS Code необходимо установить расширения зависимостей для полноценного поддержания кода. Для этого были выполнены следующие команды:

code --install-extension ms-python.python

code --install-extension ms-python.vscode-pylance

После того, как была произведена успешная установка рабочей среды, необходимо создать проект с управлением в нем зависимостей. В качестве фреймворка был выбран UV. Для его установки была выполнена следующая команда:

pip install uv

Переходим к созданию проекта. Название было выбрано стандартное venv, для создания рабочей области проекта используется следующая команда: uv init venv. По мере расширения проекта в файл pyproject.toml добавляются библиотеки, устанавливаемые следующей командой:

uv add package\_name

С актуальной версией файла pyproject.toml можно ознакомиться в листинге Д.1 приложения Д.

Разберем библиотеки, которые использовались в ходе разработки системы.

Для отображения всех необходимых элементов интерфейса UI была выбрана библиотека streamlit, ее установка была выполнена следующей командой:

uv add streamlit

Для обмена данными с сервисов GigaChat API использовалась библиотека requests, ее установка была выполнена следующей командой:

uv add requests

Для корректного формирования данных из базы знаний и документов была использована библиотека pandas, ее установка была выполнена следующей командой:

uv add pandas

Для того, чтобы получать необходимые данные из базы знаний, которая представляет собой реляционную базу данных, была использована библиотека psycopg2, ее установка была выполнена следующей командой:

uv add psycopg2

Для того, чтобы обрабатывать данные из документа формата docx была использована библиотека python-docx, ее установка была выполнена следующей командой:

uv add python-docx

Для того, чтобы обрабатывать данные из документа формата pptx была использована библиотека python-pptx, ее установка была выполнена следующей командой:

uv add python-pptx

Для того, чтобы обрабатывать данные из документа формата pdf была использована библиотека pypdf, ее установка была выполнена следующей командой:

uv add pypdf

Для того, чтобы обрабатывать и суммировать большой объем текста были использованы комплекс библиотек: langchain, langchain-community, langchain-core, langchain-text-splitters, их установка была выполнена следующей командой:

uv add langchain langchain-community langchain-core langchain-text-splitters

Для того, чтобы загрузить обработанный и суммированный текст в модель были использованы две библиотеки: langchain-gigachat, langchain-ollama. GigaChat предназначена для удаленного взаимодействия, Ollama для локального обращения к модели, их установка была выполнена следующей командой:

uv add langchain-gigachat langchain-ollama

Сразу после установки streamlit необходимо настроить secrets.toml расположенный в папке .streamlit – папка окружения streamlit. В ней необходимо оставить данные удаленного взаимодействия с GigaChat API и базой знаний, которая представляет собой реляционную базу данных. Итоговая конфигурация представлена в листинге Д.2 приложения Д.

Данные этапы позволят создать полноценную работоспособную срезу для разработки описываемой системы.

## **4.2 Программная реализация**

В созданном проекте было реализовано четыре пакета: app, config, chat\_settings, docs\_settings.

В каждом пакете для дополнительного функционала для корректной работоспособности в различных частях системы, был создан файл utils.py

Пакет config предназначен для размещения библиотек, которые были описаны в разделе 4.1 и дополнительных функций конфигурации проекта, такие как: get\_gigachat\_token, get\_files\_name, get\_name\_files\_to\_id, get\_models, sql\_cursor, get\_knowledge. Все они служат для инициализации и получения данных при попадании на главную страницу системы.

Пакет chat\_settings содержит класс Chat, который служит для обмена сообщениями между сотрудником, задающий запрос, и внутренней моделью, которая обрабатывает запрос и выдает ответ. Так же пакет имеет визуальную функциональную составляющую в файле chat.py.

Пакет docs\_settings содержит классы, которые нужны для работы с базой знаний и документами. Классы LoadKnowLedge, UploadFiles, ViewFiles представляют собой функциональность, направленную на работу с документами, которые сотрудник может добавлять или удалять, а также просматривать их и то, что находится в базе знаний. Саму базу знаний сотрудник изменять не может. База знаний получается путем выгрузки разрешенных данных, предварительно обговоренной иными лицами корпорации связанных с информационной безопасностью.

Пакет app является центральным пакетом, где собираются вышеописанные пакеты для логической координации между собой и взаимодействия. Класс Main представляет собой отображения страниц системы и их структуру. Класс App является центральным классов, который наследует функции из пакета config для создания сессии, чтобы данные, при попадании на какую-либо страницу, не требовали повторного извлечения.

## **4.3 Руководство администратора**

Для того, чтобы пользоваться системой необходимо ее развернуть, в данном случае подразумевается, что у корпораций существует лицо, которое занимается данной задачей и после разворачивания интегрирует систему в корпоративный источник.

Для корректной работы системы, если модель расположенная удалена, требуются следующие системные характеристики:

* Операционная система: Windows 10 и выше, дистрибутивы ядра Linux, MacOS.
* Центральный процессов: Intel i5 и выше, AMD Ryzen 3 и выше.
* Оперативная память: минимум 8 ГБ ОЗУ и выше.
* Графический процесс: NVIDIA 1050 и выше, если центральный процессов имеет графический процессор, то он в нем может не нуждаться.
* Память компьютера: минимум 30 ГБ для установки необходимых приложений.
* Подключение к интернету для отправки запросов на удаленную модель GigaChat API.

Если же модель будет разворачиваться на личных серверах корпорации, требует следующие системные характеристики:

* Операционная система: Windows 10 и выше, дистрибутивы ядра Linux, MacOS.
* Центральный процессов: Intel Xeon и выше.
* Оперативная память: 64 ГБ ОЗУ и выше.
* Графический процесс: NVIDIA RTX 3090 и выше.
* Память компьютера: 500 ГБ и выше, если необходимо использовать большее количество моделей.

Администратор может модернизировать код программы для расширения потребностей системы и данное программное обеспечение является открытым. Для этого потребуется любая доступная среда разработки. Процесс клонирования и установки репозитория выглядит следующим образом:

1. Открыть терминал и клонировать репозиторий с GitHub: git clone https://github.com/egorbeckish/Intelligent-Search-System.
2. Перейти в папку проекта: cd Intelligent-Search-System.
3. Установка менеджера зависимостей UV: pip install uv.
4. В папке проекта необходимо установить зависимости, указанные в файле pyproject.toml: uv pip install -r path/pyproject.toml.
5. Если же модель, расположенная удаленно, то необходимо перейти на платформу GigaChat, сгенерировать необходимые API ключи для взаимодействия.
6. Сгенерированные API ключи вставить в файл secrets.toml.

После полноценной инициализации, администратор может запустить систему для проверки работоспособности: uv run streamlit run app.py

При корректном запуске, администратор может приступить к модернизации программного кода. Необходимая инструкция по установке и информации системы размещена в репозитории на GitHub в файле README.md.

## **4.4 Руководство сотрудника**

В данном разделе будет описаны основные функции системы для корректной работы сотрудника.

1. Панель перенаправления:
   1. Главная – страница описания о "Системе интеллектуального поиска в корпоративных базах знаний". На ней происходит инициализация необходимых данных, как показано на рисунке Е.1 приложения Е.
   2. Чат – страница с чатом системы.
   3. Закрыть чат – выход из системы.
   4. Загрузка документов – страница добавление документов в хранилище.
   5. Просмотр/удаление документов – страница просмотра загруженных документов и находящихся в базе знаний. Панель представлена на рисунке Е.2 приложения Е со всеми вышеперечисленными страницами.
2. Работа с чатом:
   1. Выбрать языковую модель из предоставленного списка. По умолчанию выбрана GigaChat.
   2. Выбрать область поиска ответа. По умолчанию стоит выборка из базы знаний. Так же предоставлены варианты: выбрать определенный документ, поиск из открытых источников. Краткая информация об вариантах разворачивается при наведении на них, как показано на рисунке Е.3 приложения Е.
   3. Очистить чат. При переполненном чате можно его очистить, если та информация, которая была отображена не используется.
3. Работа с загрузкой документов:
   1. Выбрать документы, хранящиеся в папке сотрудника. Для верной загрузки на виджете отображена подсказка с доступными форматами и максимальным размером документа, как показано на рисунке Е.4 приложения Е. После, при нажатии на виджет появляется интерфейс операционной системы, где показывается папка с файлами, представлено на рисунке Е.5 приложения Е.
4. Просмотр/удаление документов:
   1. Отображается список документов сотрудника и выгруженные из базы знаний.
   2. Удаление выбранных файлов из хранилища сотрудника.
   3. Удаление всех документов их хранилища сотрудника.

Следуя данной инструкции, сотрудник сможет эффективно взаимодействовать с системой.

# **5 ТЕСТИРОВАНИЕ**

# **Заключение**

# **Список используемых источников**

# **Приложение А** (справочное) **Диаграмма прецедентов**

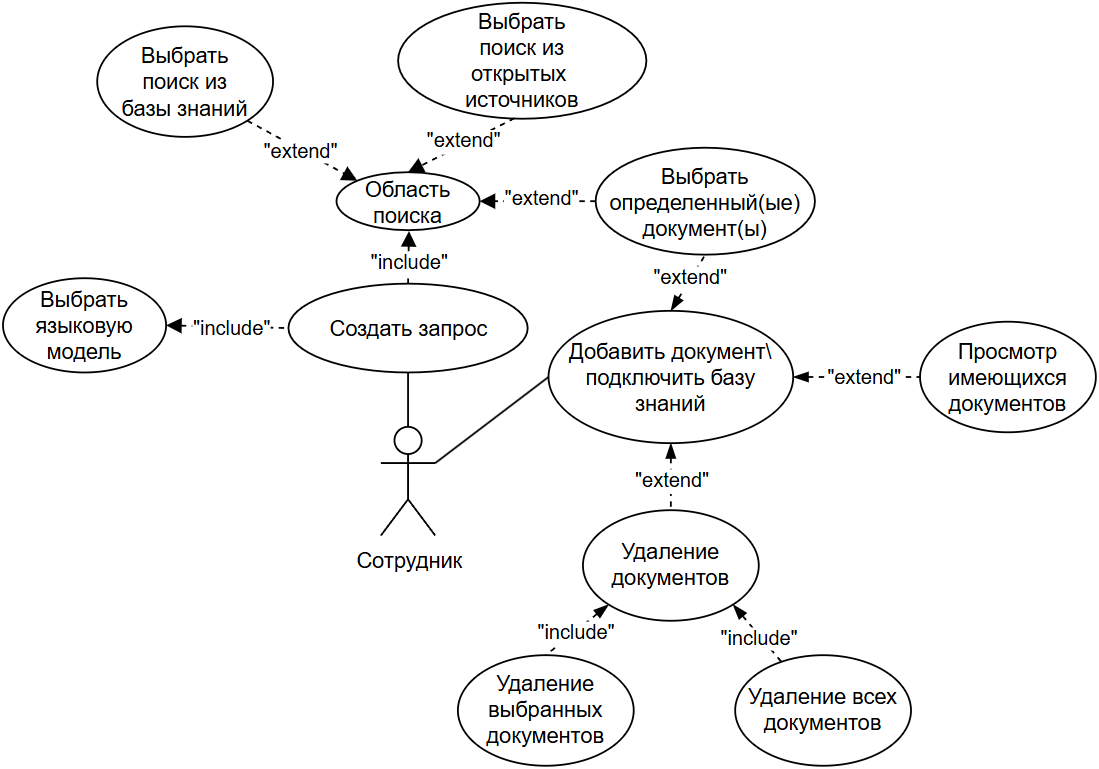


Рисунок А.1 – Диаграмма прецедентов

# **Приложение Б** (справочное) **Диаграмма последовательности**

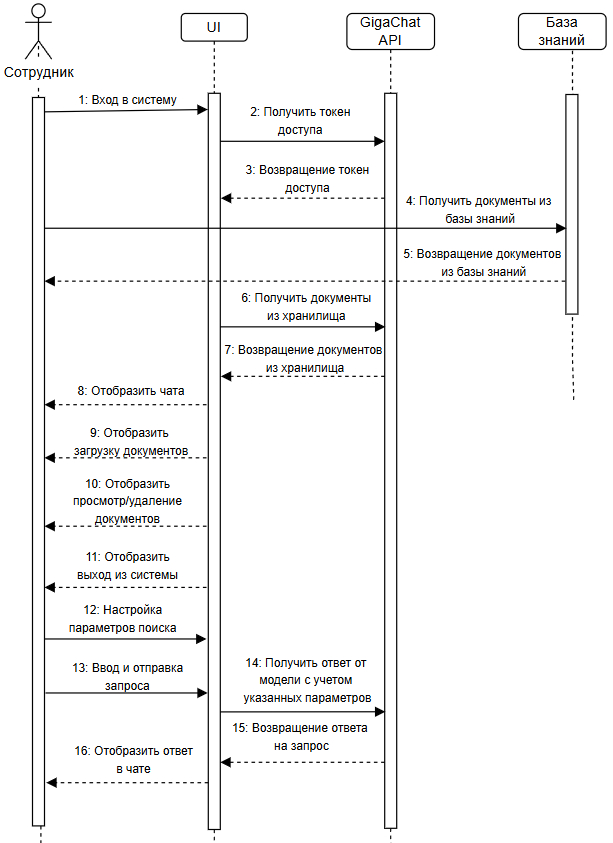


Рисунок Б.1 – Диаграмма последовательности этапа получения ответа на запрос

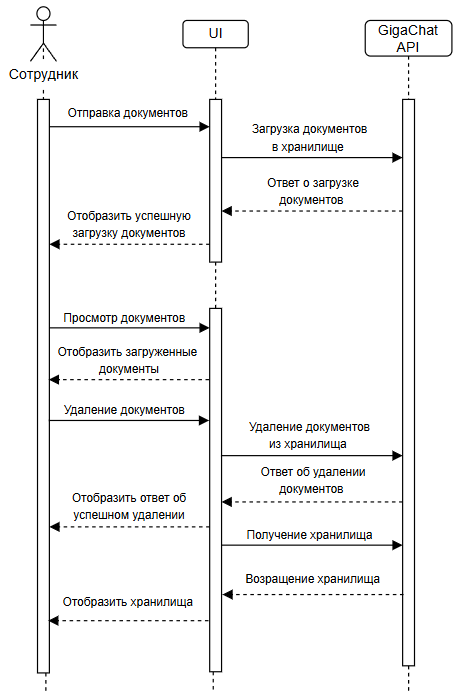


Рисунок Б.2 – Диаграмма последовательности этапа загрузки, просмотра, удаление документов из хранилища

# **Приложение В** (справочное) **Макеты интерфейса**

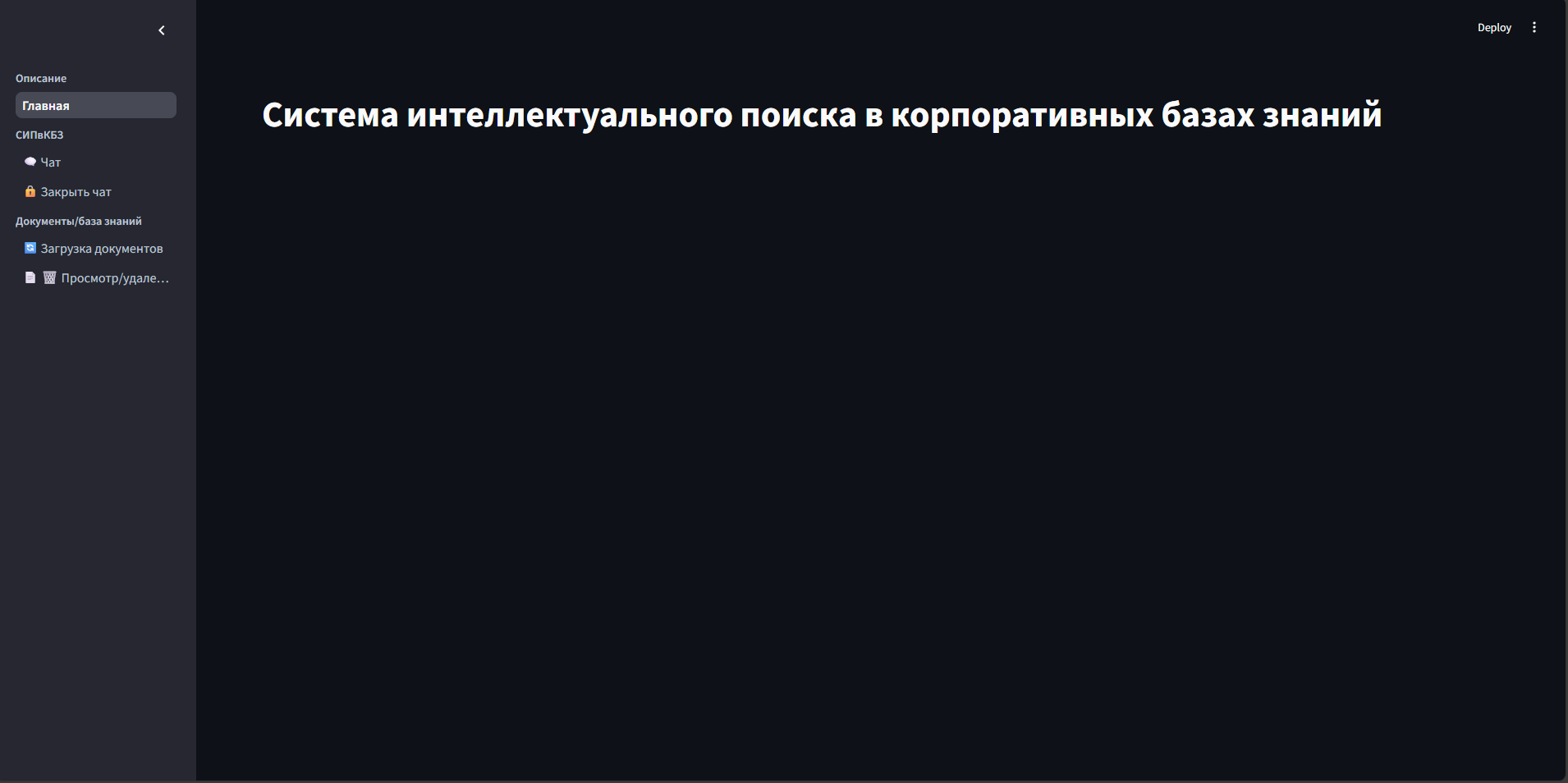


Рисунок В.1 – Макет главной страницы системы с описанием взаимодействия

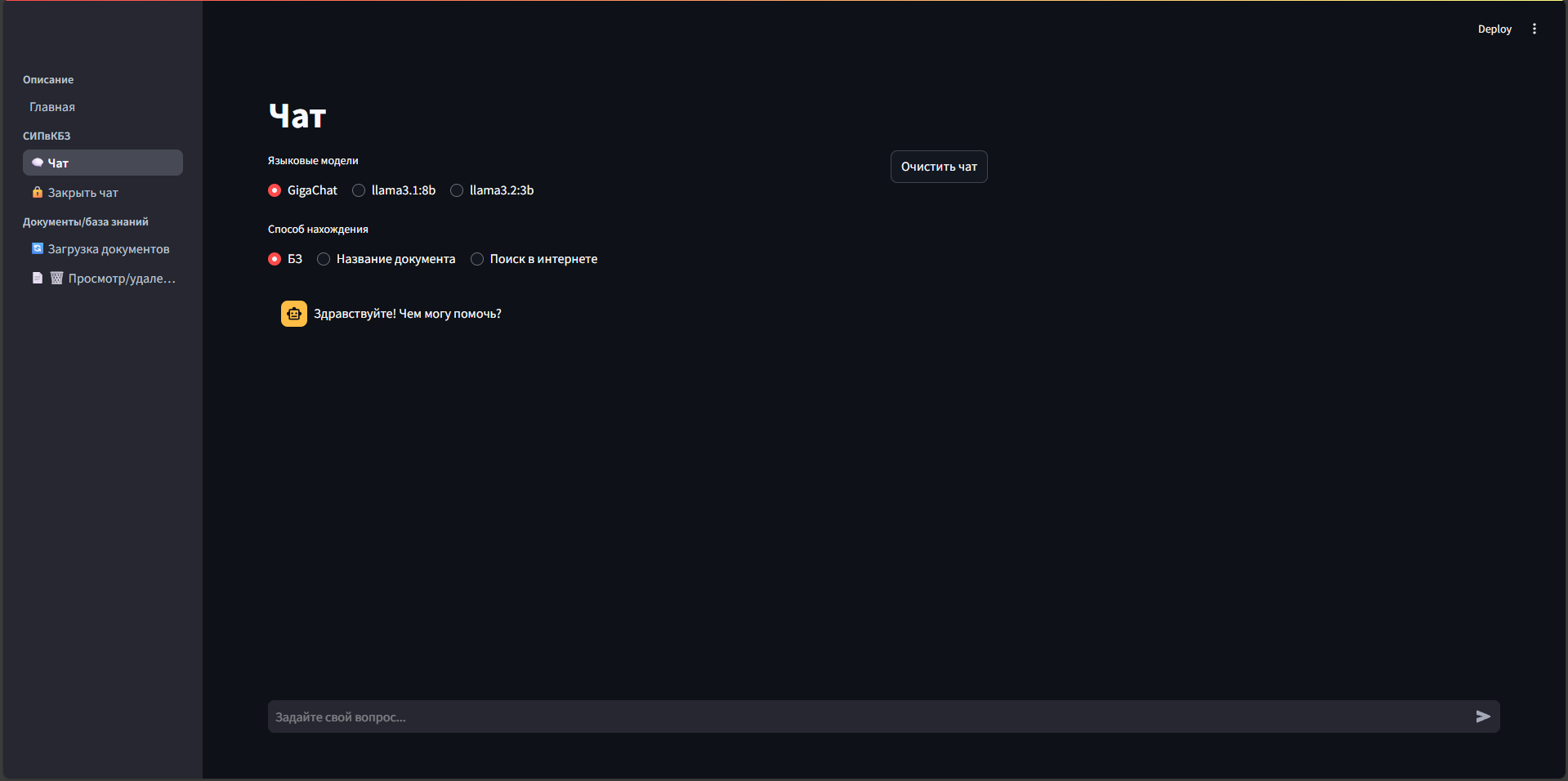
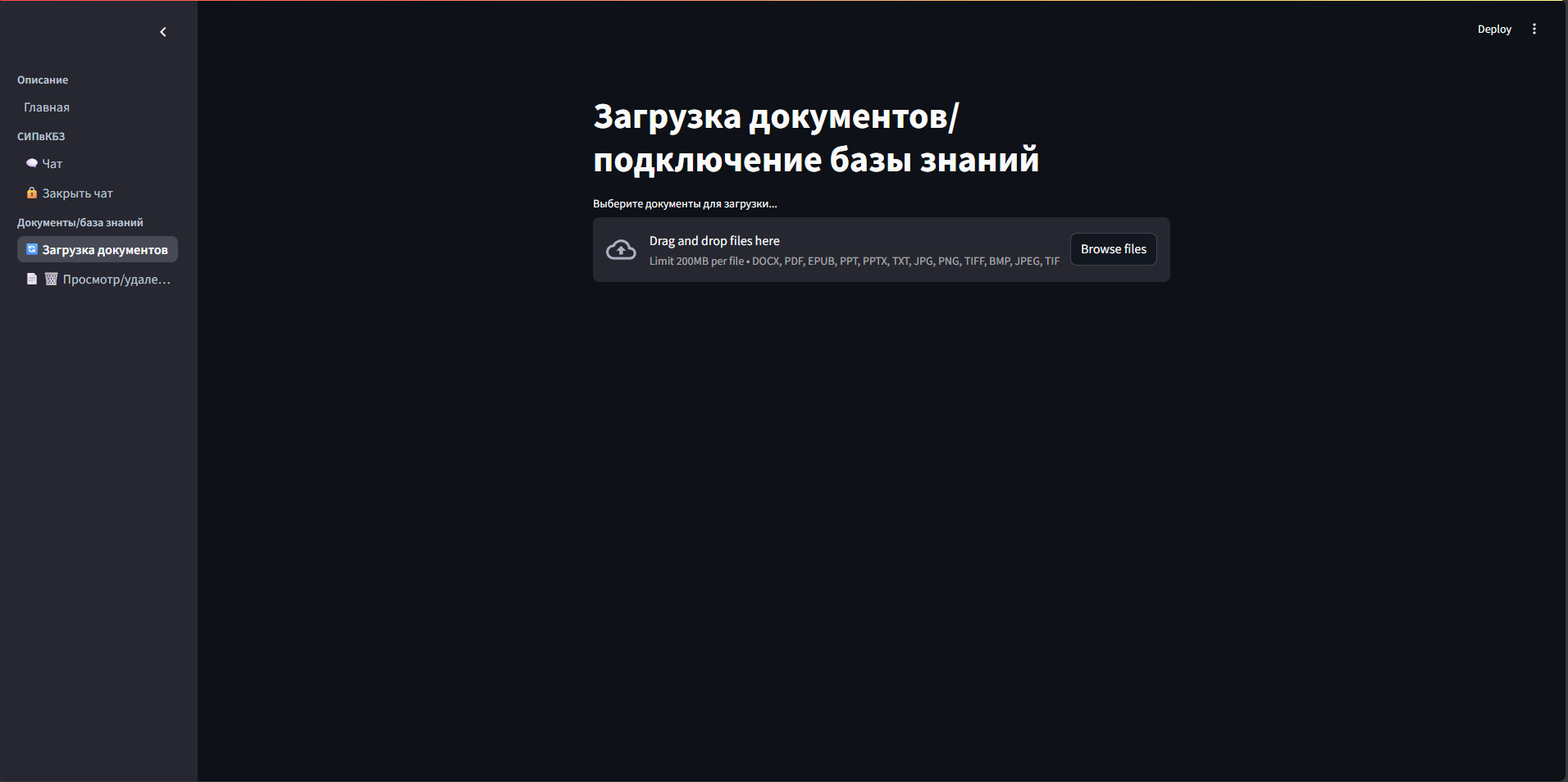


Рисунок В.2 – Макет чата системы

 Рисунок В.3 –Макет загрузки документов с предоставление справочной информации

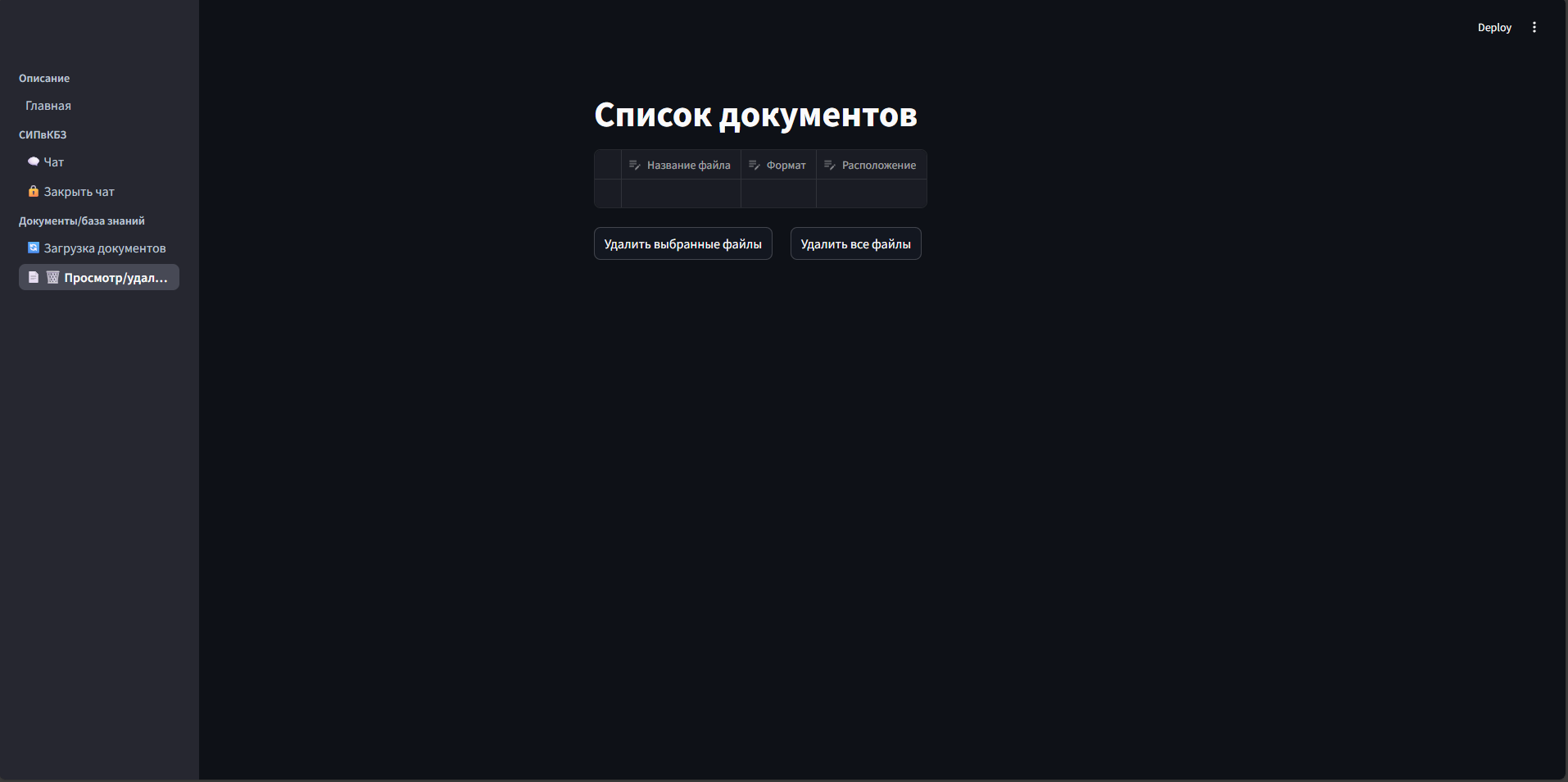


Рисунок В.4 –Макет просмотра/удаления документов из хранилища

# **Приложение Г** (справочное) **Диаграмма классов**

# **Приложение Д** (справочное) **Конфигурация системы**

Листинг Д.1 – Конфигурация проекта venv в файле pyproject.toml

[project]

name = "venv"

version = "0.1.0"

description = ""

readme = "README.md"

requires-python = ">=3.12"

dependencies = [

"langchain==0.3.25",

"langchain-community==0.3.24",

"langchain-gigachat==0.3.10",

"langchain-ollama==0.3.3",

"langchain-text-splitters==0.3.8",

"pandas==2.2.3",

"psutil==7.0.0",

"psycopg2==2.9.10",

"pypdf==5.5.0",

"python-docx==1.1.2",

"python-pptx==1.0.2",

"requests==2.32.3",

"streamlit==1.45.1",

"uv==0.7.8",

]

Листинг Д.2 – Конфигурация secrets.toml

CLIENT\_ID="ключ\_доступа"

SECRET=" ключ\_доступа "

AUTH\_KEY=" ключ\_доступа "

[система\_бд]

host="адрес

port="номер\_порта"

dbname="имя\_базы\_данных"

user="пользователь"

password="пароль"

[система\_бдlink]

link="система\_бд://пользователь:пароль@адрес/имя\_базы\_данных"

# **Приложение Е** (справочное) **Реализация системы**

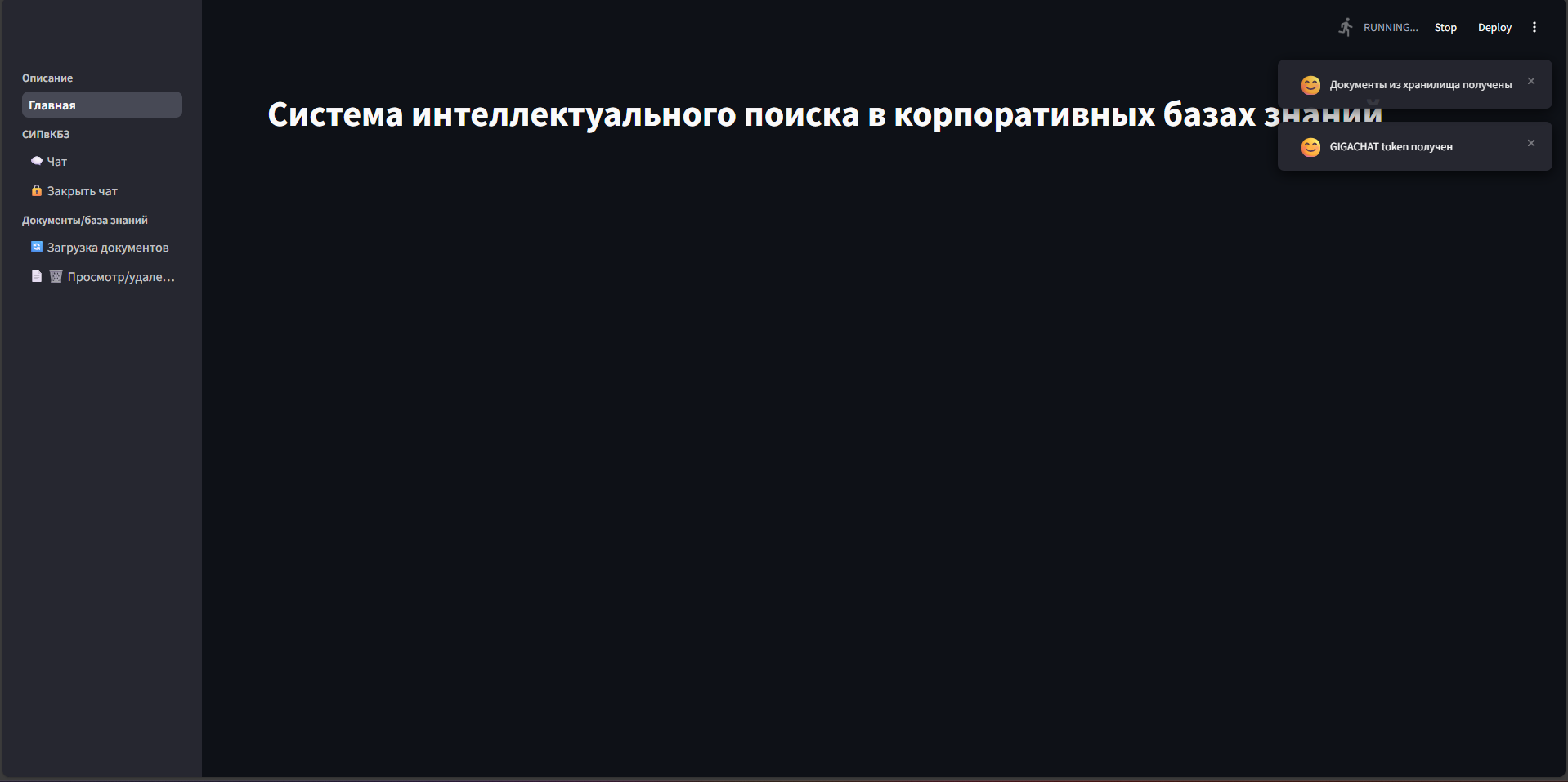


Рисунок Е.1 – Инициализация необходимых данных для работы системы

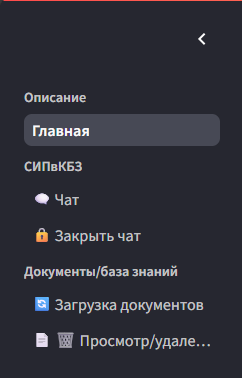


Рисунок Е.2 – Панель перемещения по страницам системы

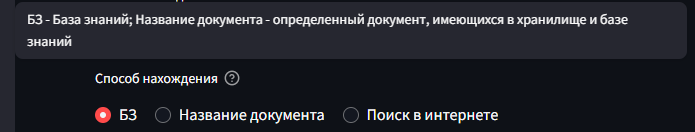


Рисунок Е.3 – Всплывающая подсказка при наведении на параметр

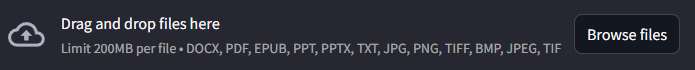


Рисунок Е.4 – Подсказка на самом виджете

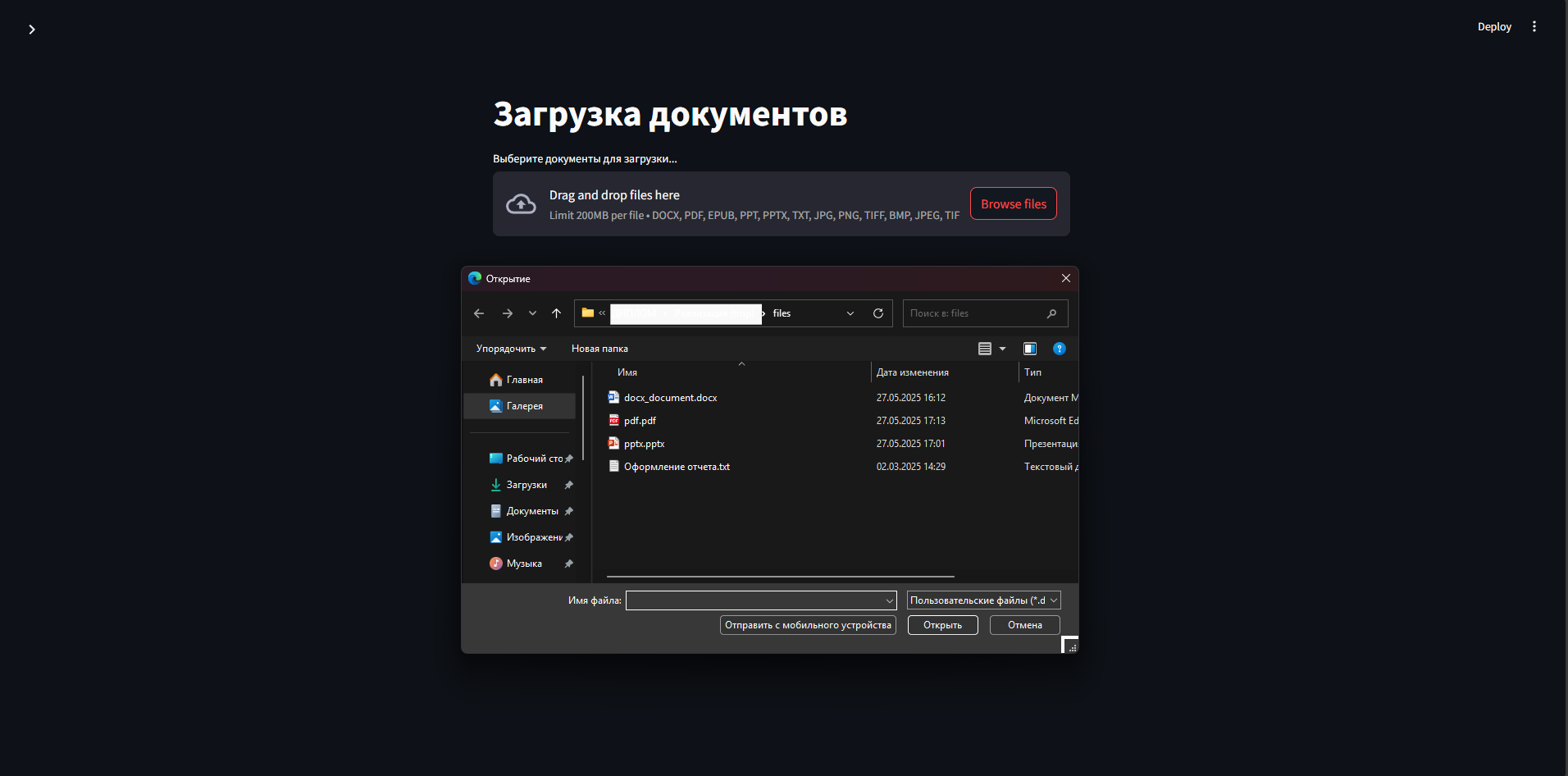


Рисунок Е.5 – Выбор документов при помощи вызова интерфейса из операционной системы

# **Приложение Ж** (справочное) **Тестирование системы**

# **Приложение З** (справочное) **Список научных трудов по теме ВКР**