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**

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

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

2. Срок сдачи студентом законченной БР: « » 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: « » 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: « » мая 202*5*г. | | | | |  |  |
| Студент гр. 431-3 | (подпись) | | Бекиш Е.П.  (Ф.И.О.) | | | |

**Реферат**

Бакалаварская 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: является использование методов и технологий языковых моделей в корпоративных базах знаний.

Цель работы: разработать систему, которая обеспечивает по заданному запросу сотрудника выдавать быстрый и корректный ответ, используя внутреннею базу знаний, для дальнейших действий.

Полученный результат работы: система, которая обеспечивает по заданному запросу пользователя выдавать быстрый и корректный ответ, используя внутреннею базу знаний, для дальнейших действий. (повтор цели работы)

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: is the use of methods and technologies of language models in corporate knowledge bases.

The purpose of the work: is to develop a system that provides a quick and correct response for further actions based on a user request.

The result of the work: a system that provides a quick and correct response to a given user request for further action. (повтор цели работы)

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 7](#_Toc196399476)

[1 АНАЛИЗ ТРЕБОВАНИЙ 9](#_Toc196399477)

[1.1 Проблематика 9](#_Toc196399478)

[1.2 Развитие больших языковых моделей 9](#_Toc196399479)

[1.2.1 Появление обработки естественного языка 9](#_Toc196399480)

[1.2.2 Развитие векторного представления слов 10](#_Toc196399481)

[1.2.3 Выборка необходимых документов и данных 23](#_Toc196399482)

[1.3 Формулирование требований 26](#_Toc196399483)

[1.4 Обзор аналогов 27](#_Toc196399484)

[2 ФУНКЦИОНАЛЬНЫЕ ВОЗМОЖНОСТИ И СПЕЦИФИКАЦИИ 29](#_Toc196399485)

[2.1 Выбор модели 29](#_Toc196399486)

[2.2 Выбор инструментов реализации 30](#_Toc196399487)

[2.3 Определение входных и выходных данных 32](#_Toc196399488)

[2.3.1 Входные данные 32](#_Toc196399489)

[2.3.2 Выходные данные 32](#_Toc196399490)

[3 ПРОЕКТИРОВАНИЕ 33](#_Toc196399491)

# **Введение**

Языковые модели [] являются важной и активно развивающейся областью, которая за последние несколько лет привлекает большое внимание промышленности, 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 очень большего количество времени.

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 в свои основные системы, чтобы находить необходимые данные в своей базе знаний.

Целью работы является разработка системы, обеспечивающая по заданному запросу сотрудника ответ, который является релевантным.

Для достижения поставленной цели необходимо выполнить следующие задачи:

1. Анализ необходимых требований;
2. Функциональные возможности и спецификации;
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. 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**

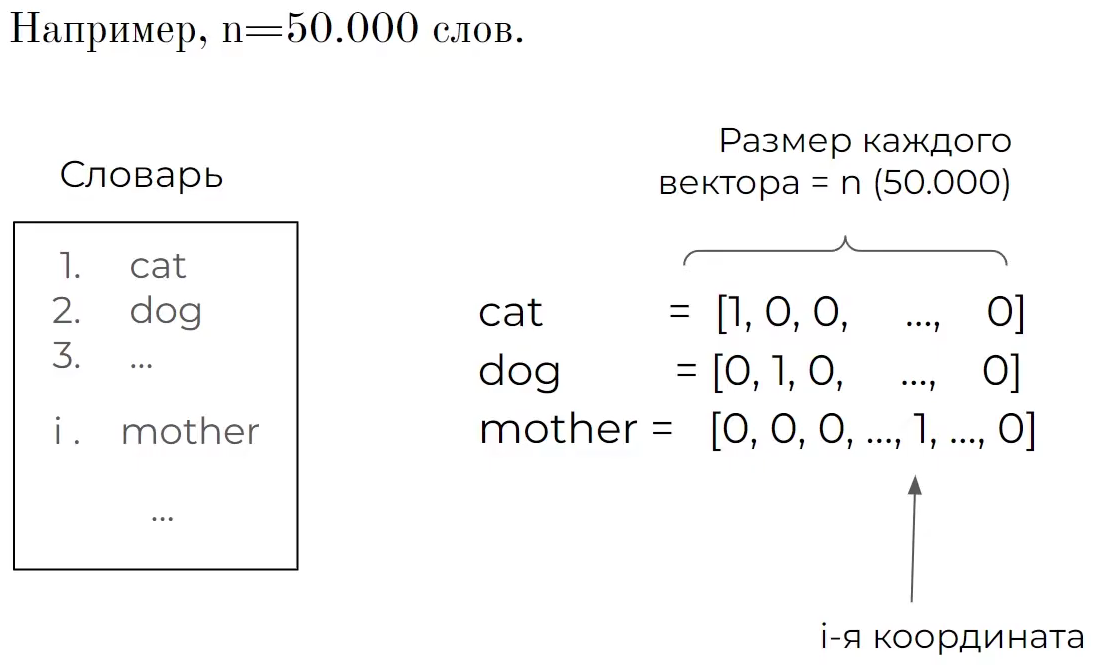
В 1950-м году Алан Тьюринг опубликовал статью "Computing Machinery and Intelligence" [1], в которой рассматривал вопрос: «Могут ли машины мыслить?», где, для начала, нужно определить термины в данном вопросе - «мыслить» и «машина». В дальнейшем выдвинул идею, что человек может взаимодействовать с машиной 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.

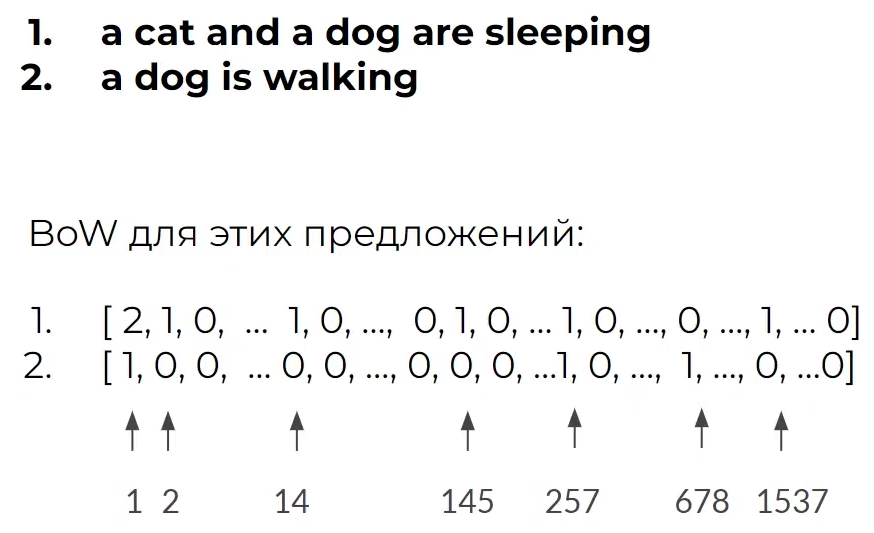
One-hot encoding – метод представления в векторном виде. Создается словарь фиксированного размера *N*, где для каждого слова из словаря соответствует вектор размера *N*, каждый вектор состоит из 1 единицы на -м месте и (*N – 1*) нулей, где – номер слова в словаре. Таким образом, запись данного метода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

Рисунок 1.1 – One-hot encoding

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

* 1. Размерность векторов могут 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)

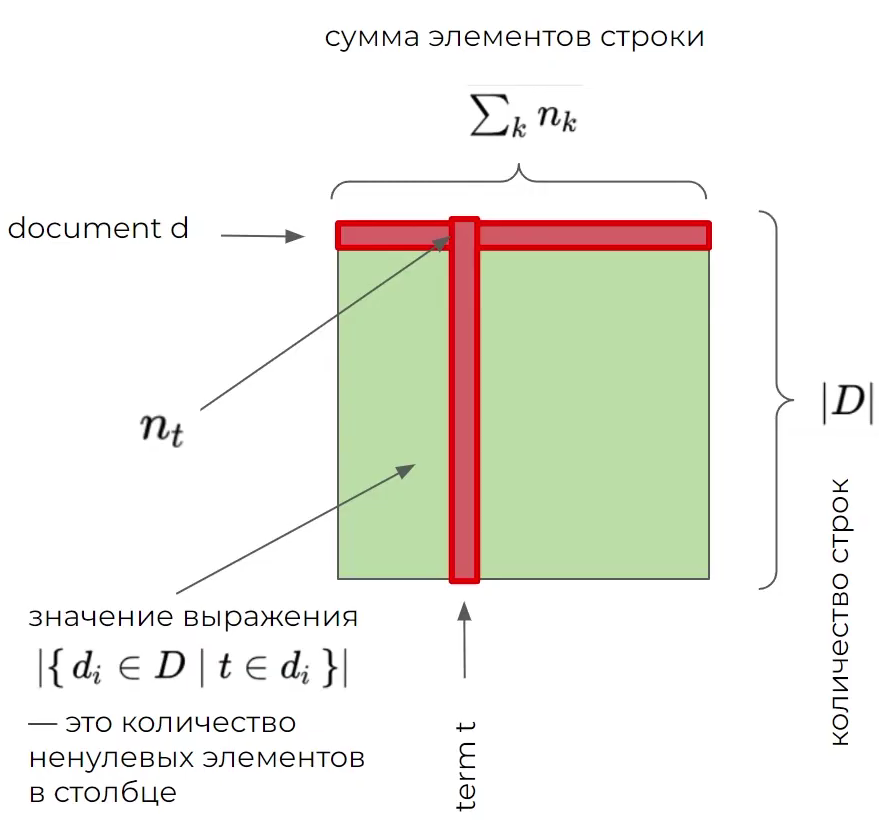
где – входное слово;

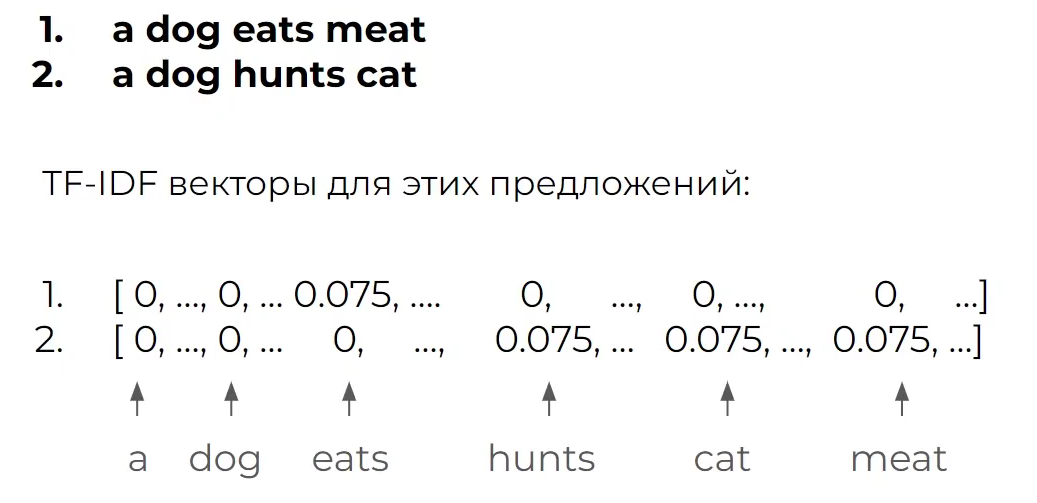
– коллекция документов, их количество;

– 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) – пары слов;
3. Триграммы (3-граммы) – тройки 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».

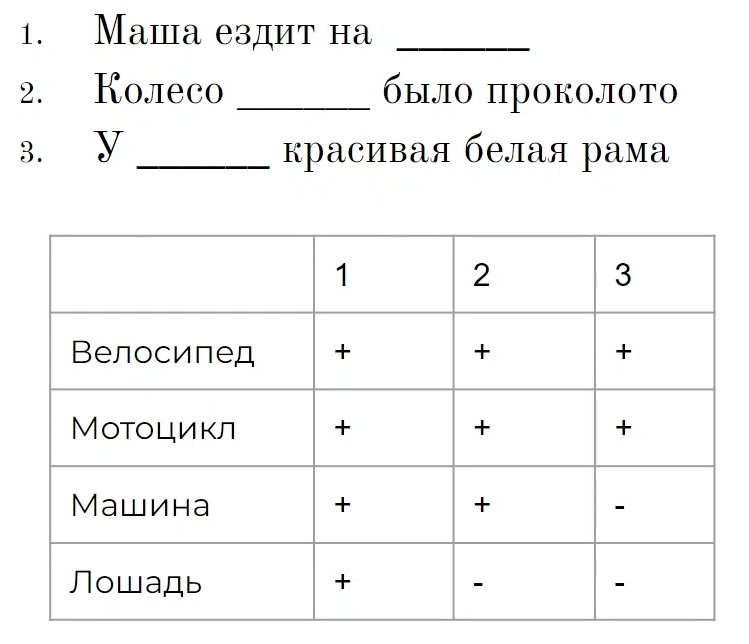


Рисунок 1.5 – Контекстный смысл слов

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.

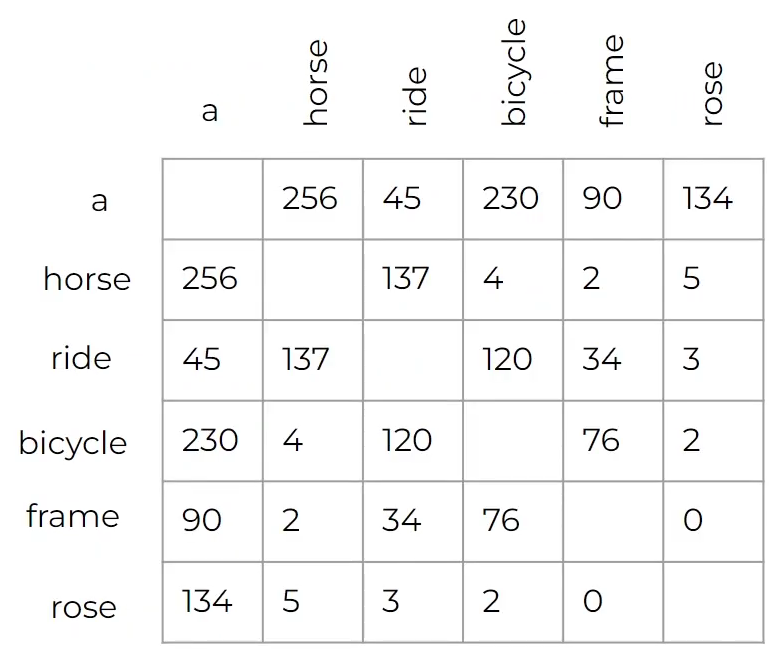


Рисунок 1.6 – Матрица контекстных векторов слов

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.

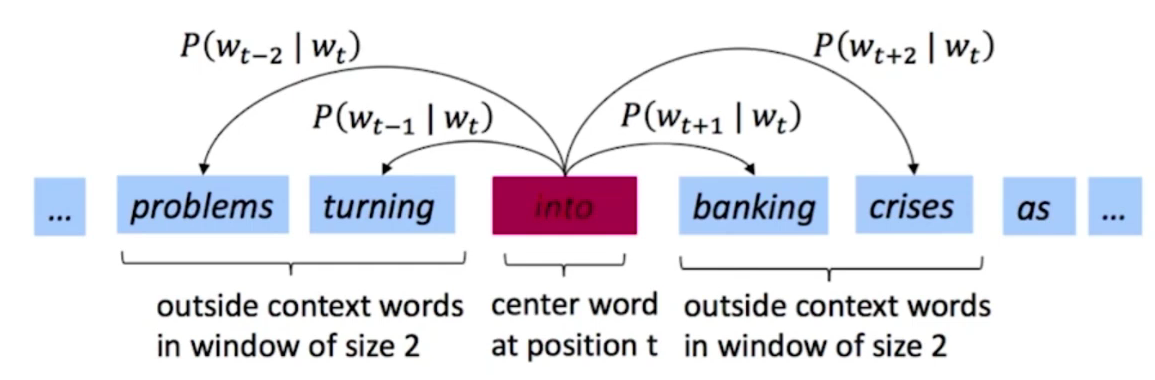
**

Рисунок 1.7 – Схематическое изображение алгоритма при центральном слове into

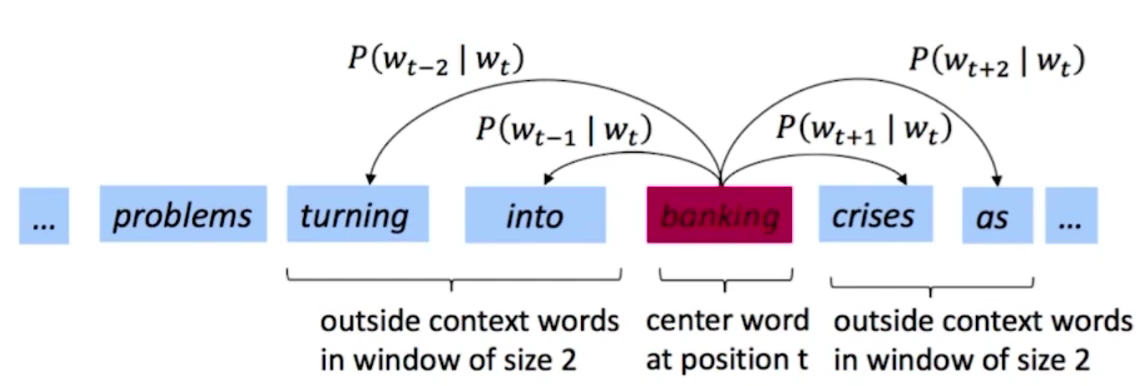
**

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.

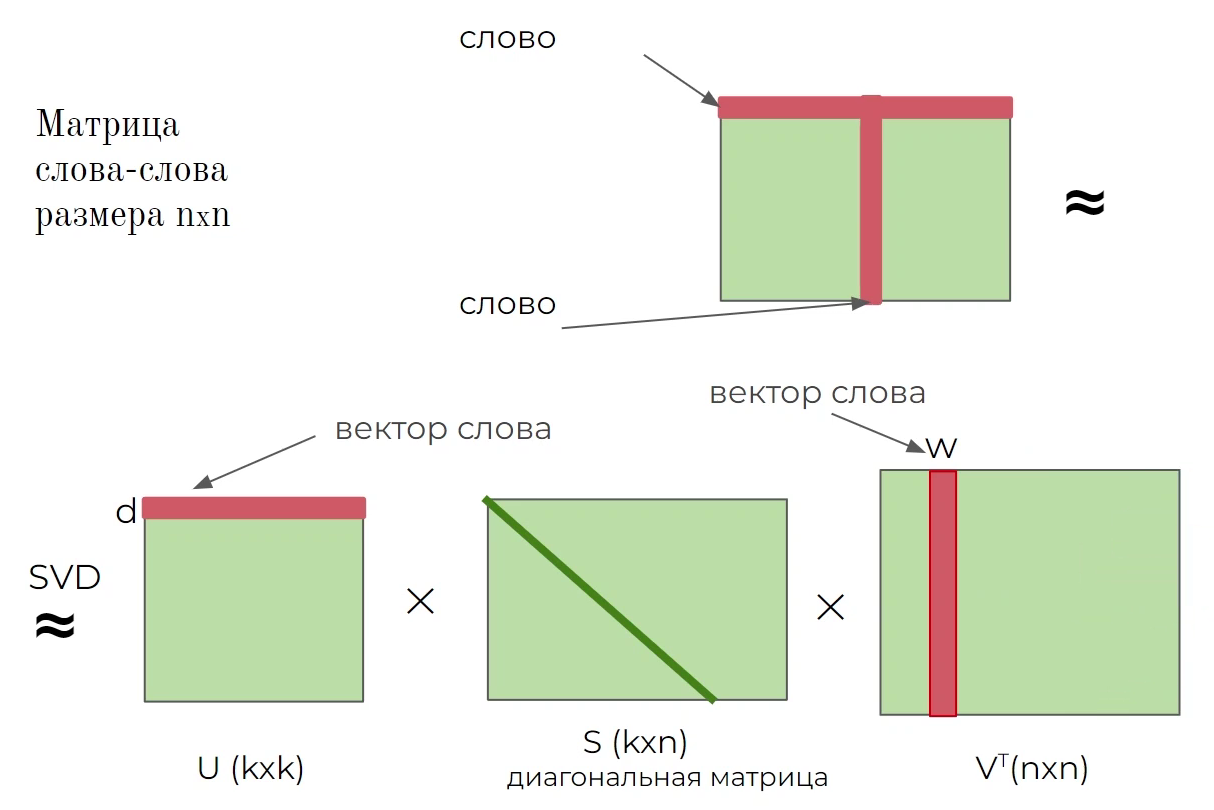


Рисунок 1.9 – Использование SVD разложения

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.

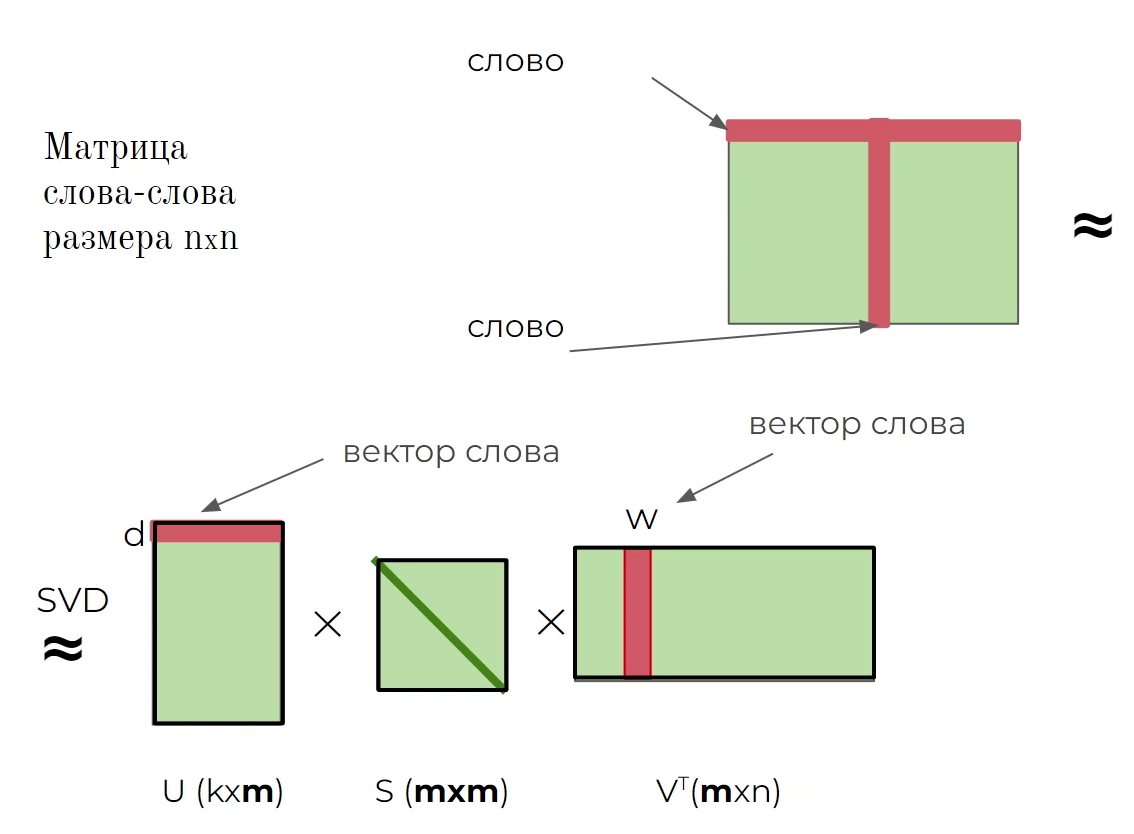


Рисунок 1.10 – Латентный семантический анализ

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.

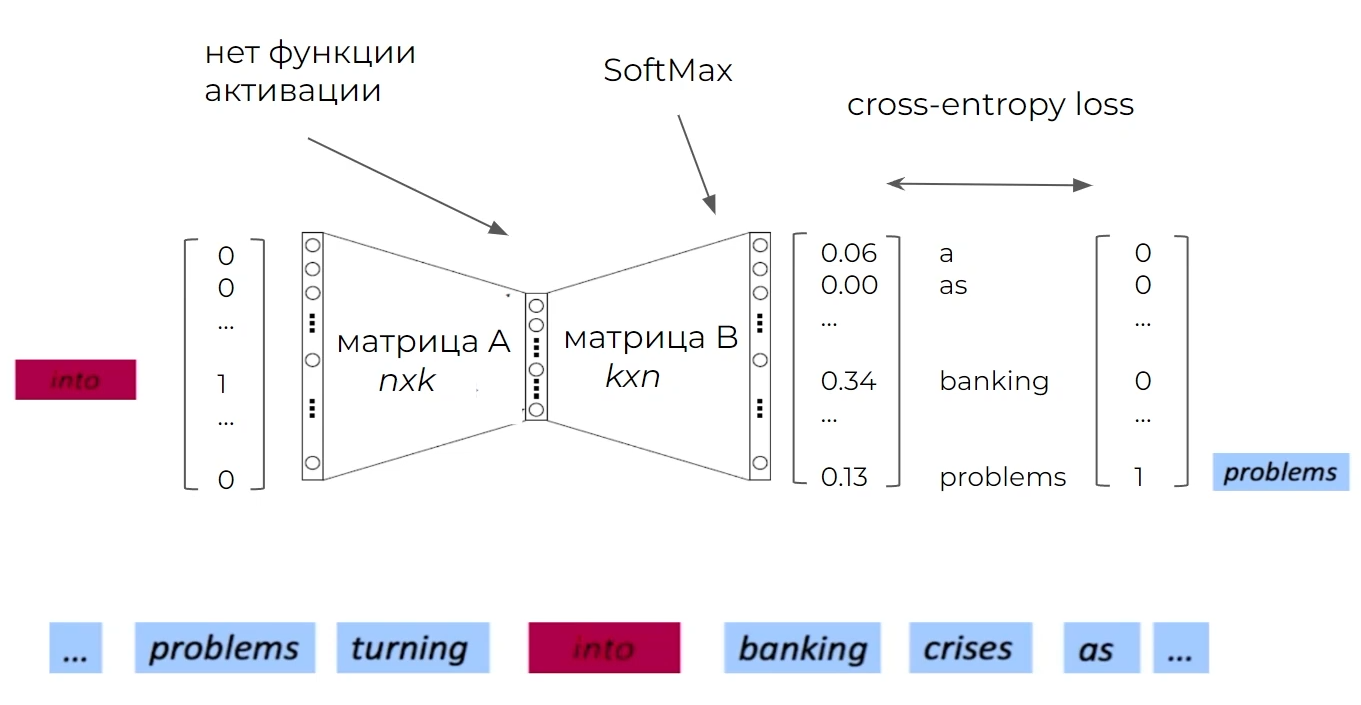
**

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 косинусного расстояния.

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

На данный момент, большинство общедоступных LLM при создании промптах 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 «подмешиваете» 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.

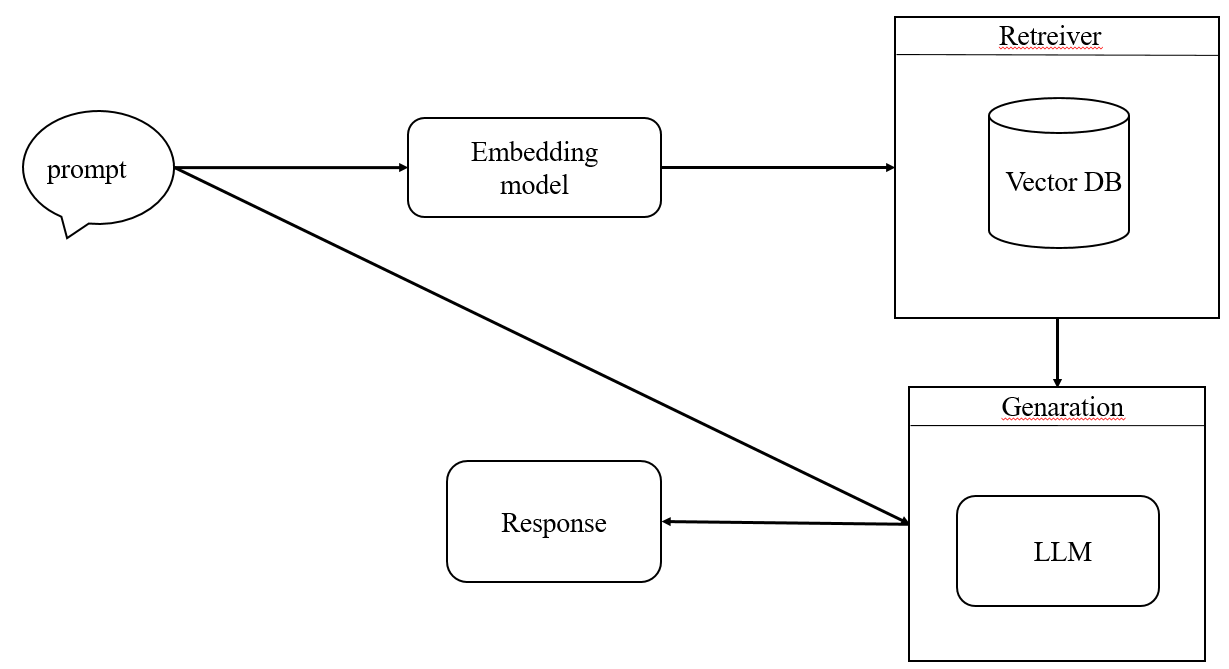


Рисунок 1.12 – концепция RAG

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.

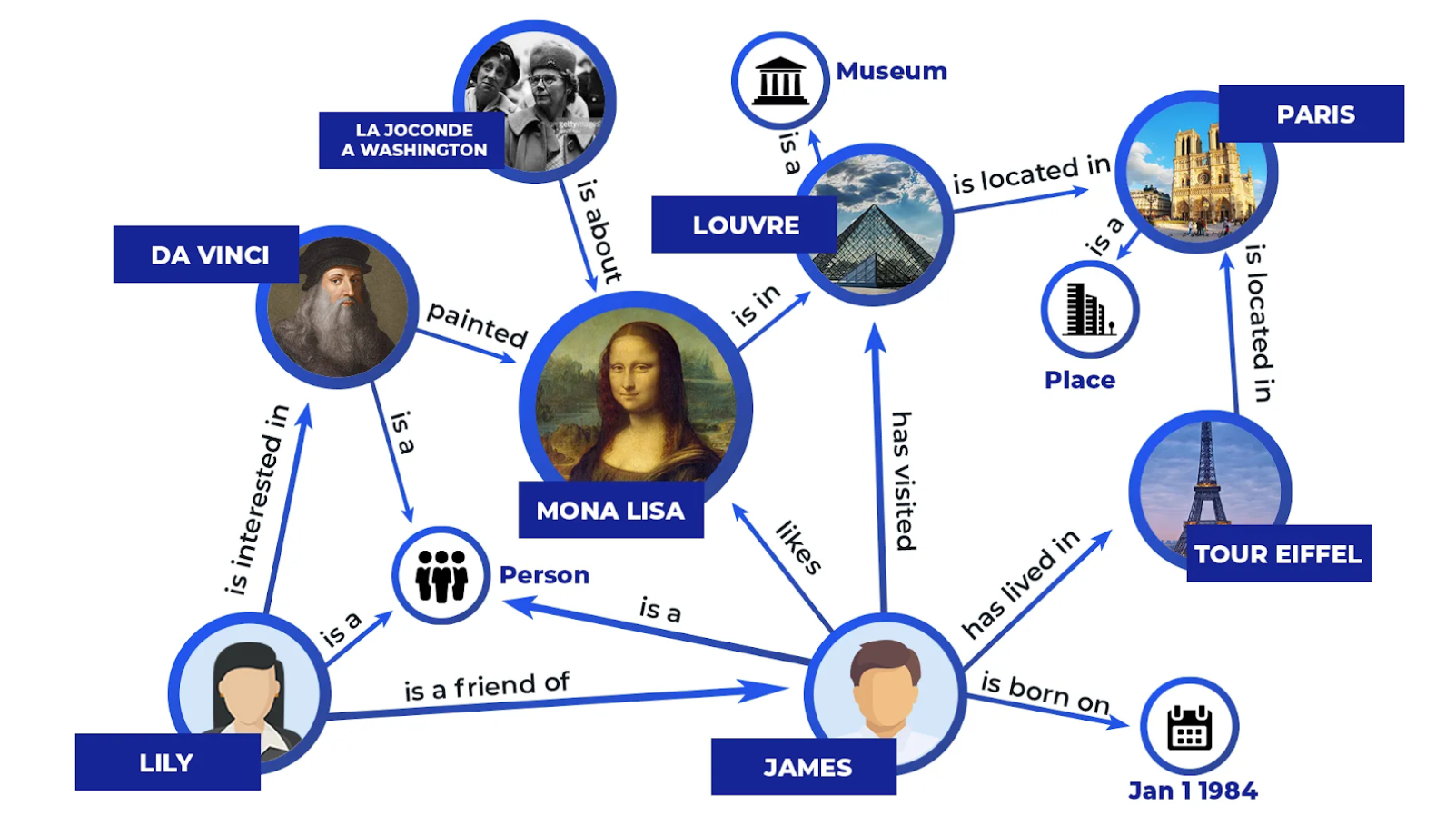


Рисунок 1.13 – Граф знаний

## **1.3 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.

## **1.4 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 [].

# **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**