

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

**«Дальневосточный федеральный университет»**

(ДВФУ)

Институт математики и компьютерных технологий

Департамент программной инженерии и искусственного интеллекта

Хмелевский Егор Дмитриевич

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

**магистерская диссертация**

вид ВКР

АВТОМАТИЗАЦИЯ ПРОЦЕССА РАЗМЕТКИ ТЕКСТОВЫХ ДОКУМЕНТОВ ВОПРОСНО-ОТВЕТНОЙ СИСТЕМЫ С РАЗМЕЧЕННОЙ КОЛЛЕКЦИЕЙ ДОКУМЕНТОВ

по направлению подготовки (специальности) 01.04.02 «Прикладная математика и информатика» магистерская программа «Перспективные методы искусственного интеллекта в сетях передачи и обработки данных»

Владивосток

2025

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| В материалах данной выпускной  квалификационной работы не содержатся  сведения, составляющие государственную  тайну, и сведения, подлежащие экспортному  контролю | | | | | |  | Автор работы | | |  | |
|  |  | | | подпись | |
|  | группа | | М9123-01.04.02пми | | |
|  | « |  | » |  | 2025 г. |
|  |  | | | | |
|  | | | | | |  |  | | | | |
| Уполномоченный по экспортному контролю | | | | | |  | Руководитель ВКР | | | | |
|  | | | | | |  | доцент, канд. техн. Наук | | | | |
|  | | | | | |  | должность, ученое звание | | | | |
|  | | | И.Л. Артемьева | | |  |  | | | С.Н. Остроухова | |
| подпись | | | И.О. Фамилия | | |  | подпись | | | И.О. Фамилия | |
| « |  | » |  | 2025 г. | |  | « |  | » |  | 2025 г. |
|  | | | | | |  |  | | | | |
|  | | | | | |  | Консультант | | | | |
|  | | | | | |  |  | | |  | |
|  | | | | | |  | подпись | | | И.О. Фамилия | |
|  | | | | | |  | « |  | » |  | 2023 г. |
|  | | | | | |  |  | | | | |
|  | | | | | |  | Назначен рецензент | | | | |
|  | | | | | |  |  | | | | |
|  | | | | | |  | ученое звание | | | | |
|  | | | | | |  |  | | | | |
|  | | | | | |  | фамилия, имя, отчество | | | | |
|  | | | | | |  |  | | | | |
| Защищена в ГЭК с оценкой | | | | | |  | **«Допустить к защите»** | | | | |
|  | | | | | |  | и.о. директора департамента | | | | |
|  | | | | | |  |  | | | | |
| Секретарь ГЭК | | | | | |  | ученая степень, ученое звание | | | | |
|  | | | Е.В. Моисеенко | | |  |  | | | О.А. Крестникова | |
| подпись | | | И.О. Фамилия | | |  | подпись | | | И.О. Фамилия | |
| « |  | » |  | | 2025 г. |  | « |  | » |  | 2025 г. |
|  |  |  |  | |  |  |  |  |  |  |  |

Оглавление

[Введение 4](#_Toc200903603)

[Глава 1 Обзор методов подходов и методов разметки текстовых документов 6](#_Toc200903604)

[1.1 Вопросно-ответная система 6](#_Toc200903605)

[1.1.1 Вопросно-ответные системы, использующие веб-поиск 7](#_Toc200903606)

[1.1.2 Вопросно-ответные системы с коллекцией вопросов и ответов 8](#_Toc200903607)

[1.1.3 Экспертные вопросно-ответные системы 8](#_Toc200903608)

[1.1.4 Вопросно-ответные системы с коллекцией индексированных документов 9](#_Toc200903609)

[1.2 RAG 10](#_Toc200903610)

[1.3 Общий подход к извлечению именованных сущностей из текста 13](#_Toc200903611)

[1.3.1 Поиск кандидатов в именованные сущности 14](#_Toc200903612)

[1.3.2 Выделение признаков 15](#_Toc200903613)

[1.3.3 Ранжирование и отсечение 16](#_Toc200903614)

[1.4 Подходы к извлечению именованных сущностей 17](#_Toc200903615)

[1.4.1 Статический подход 17](#_Toc200903616)

[1.4.2 Лингвистический подход 25](#_Toc200903617)

[1.4.3 Машинное обучение 26](#_Toc200903618)

[1.4.4 Применимость подходов 28](#_Toc200903619)

[1.5 Выводы 30](#_Toc200903620)

[Глава 2 Анализ предметной области «Разметка текстовых документов» 31](#_Toc200903621)

[2.1 Термины предметной области 31](#_Toc200903622)

[2.2 Анализ профессиональной деятельности 32](#_Toc200903623)

[2.3 Модель предметной области 40](#_Toc200903624)

[2.4 Методы решения задач 44](#_Toc200903625)

[Глава 3 Проект программной системы 49](#_Toc200903626)

[3.1 Архитектурно-контекстная диаграмма 49](#_Toc200903627)

[3.2 Модуль NER 50](#_Toc200903628)

[3.3 Модуль формирования словаря 51](#_Toc200903629)

[3.4 Модуль поиска 52](#_Toc200903630)

[3.5 Диаграмма Use-Case 53](#_Toc200903631)

[3.6 Спецификация остальных требований 54](#_Toc200903632)

[3.6.1 Требования к удобству использования ПС 54](#_Toc200903633)

[3.6.2 Требования к удобству использования ПС 55](#_Toc200903634)

[3.7 Проект верхнего уровня 55](#_Toc200903635)

[3.7.1 Структура базы данных 55](#_Toc200903636)

[3.7.2 Проект интерфейса 58](#_Toc200903637)

[Глава 4 Реализация и тестирование системы 68](#_Toc200903638)

[4.1 Технологический стек 68](#_Toc200903639)

[4.1.1 Frontend 68](#_Toc200903640)

[4.1.2 Backend 68](#_Toc200903641)

[4.1.3 База данных 69](#_Toc200903642)

[4.1.4 Сборка проекта 69](#_Toc200903643)

[4.1.5 Среда разработки 69](#_Toc200903644)

[4.2 Тестирование 70](#_Toc200903645)

[4.3 Эксперимент 70](#_Toc200903646)

[4.1.1 Эксперимент №1 70](#_Toc200903647)

[4.1.2 Эксперимент №2 72](#_Toc200903648)

[4.1.3 Выводы к экспериментам 74](#_Toc200903649)

[Заключение 75](#_Toc200903650)

[Список литературы 76](#_Toc200903651)

[Приложение А 80](#_Toc200903652)

# Введение

Объём информации во всём мире стремительно растёт, и задача быстрого поиска ответов становится всё актуальнее. Одной из ключевых задач в этой области является автоматизация процесса разметки текстовых документов, в частности для вопросно-ответных систем, где требуется не только выделять значимые фрагменты текста, но и обеспечивать их связь с потенциальными вопросами пользователей.

Для решения задачи разметки важно разметить текстовый документ таким образом, чтобы по нему можно было осуществлять поиск релевантной информации. Для этого прибегают к одной из задач NLP (Natural Language Processing) – извлечению именованных сущностей (NER), что позволяет автоматически находить в тексте ключевые словосочетания. Для точного извлечения именованных сущностей важную роль играют термины, а также их синонимы и значения, которые человек способен интуитивно выделять, читая текст. Они позволяют более точно характеризовать содержание документа. Автоматизация этого процесса позволяет не только ускорить обработку больших массивов данных, но и повысить качество поиска информации, что особенно важно для построения эффективных вопросно-ответных систем.

Одним из современных подходов, позволяющих объединяющих разметку текстовых документов и поиск по размеченным данным является Retrieval-Augmented Generation (RAG). Благодаря этому подходу значительно повышается точность и полнота выдачи в вопросно-ответных сервисах.

Постоянное увеличение объёма информации, а также появление новых областей знаний требуют непрерывного совершенствования методов автоматической разметки и поиска. В современных условиях особую актуальность приобретает задача построения систем, способных автоматически выделять и структурировать ключевые элементы текста, обеспечивая тем самым эффективную работу вопросно-ответных сервисов и других интеллектуальных систем обработки информации.

**Целью** данной работы является автоматизация процесса разметки текстовых документов для вопросно-ответных систем на основе размеченной коллекции документов.

Для достижения поставленной цели необходимо решить следующие **задачи**:

1. Обзор подходов и методов разметки текстовых документов.
2. Анализ предметной области «Разметка текстовых документов».
3. Проектирование и реализация прототипа системы.
4. Тестирование и экспериментальное исследование.

# Обзор методов подходов и методов разметки текстовых документов

В современных вопросно-ответных системах (QA) ключевую роль играет не только генерация релевантных ответов, но и эффективное извлечение информации из больших массивов текстовых данных. Одним из современных и перспективных подходов в этой области является Retrieval-Augmented Generation (RAG) – технология, которая объединяет два этапа: **retrieval** (поиск и извлечение релевантной информации) и **generation** (генерация ответа на основе найденных данных) [1].

Центральным этапом в архитектуре RAG выступает компонент Retrieval. Именно качество и точность retrieval-этапа определяют, насколько полно и корректно система сможет удовлетворить информационные потребности пользователя.

Для успешной реализации retrieval-компонента необходимо автоматизированно извлекать из текстов именованные сущности –последовательности слов, отражающие ключевые характеристики и темы документа. Эти сущности служат основой для индексирования, поиска и последующего связывания пользовательских запросов с релевантными текстовыми фрагментами. Поэтому обзор и анализ методов извлечения именованных сущностей становится фундаментальной задачей при построении современных QA-систем на основе RAG.

В данной главе основное внимание уделяется видам QA-систем, концепции RAG и разметке текстовых документов в виде извлечения именованных сущностей.

## Вопросно-ответная система

**Вопросно-ответная система (QA)** – информационная система, способная принимать вопросы и отвечать на них на естественном языке.

Различают 4 вида QA-систем по тому, на чём они основываются [27]:

* Вопросно-ответные системы, использующие веб-поиск.
* Вопросно-ответные системы с коллекцией вопросов и ответов.
* Экспертные вопросно-ответные системы.
* Вопросно-ответные системы с коллекцией индексированных документов.

### Вопросно-ответные системы, использующие веб-поиск

Вопросно-ответные системы на базе веб-поиска используют результаты поисковых систем (например, Google или Яндекс) как источник знаний. Пользователь задаёт вопрос на естественном языке, система преобразует его в поисковый запрос, получает список ссылок с краткими текстовыми фрагментами, анализирует их с помощью методов обработки естественного языка и формирует ответ.

**Преимущества** – отсутствие необходимости в собственной базе знаний и решении задачи поиска релевантных документов.

**Недостатки** – зависимость качества ответов от поисковой системы.

На рисунке 1 представлена схема QA-системы на основе веб-поиска.

Изображение выглядит как текст, снимок экрана, линия, диаграмма

Содержимое, созданное искусственным интеллектом, может быть неверным.  
Рисунок 1 – Схема QA-системы на основе веб-поиска

### Вопросно-ответные системы с коллекцией вопросов и ответов

Вопросно-ответные системы на основе коллекции заранее подготовленных вопросов и ответов используют локальную базу знаний для поиска наиболее похожего вопроса и предоставления соответствующего ответа.

**Преимущества** – простая реализация поиска и возможность давать развернутые ответы.

**Недостатки** – необходимость предварительного сбора базы и невозможность ответить на уникальные вопросы, отсутствующие в коллекции.

На рисунке 2 представлена схема QA-системы на основе коллекций вопросов и ответов.

Изображение выглядит как текст, снимок экрана, диаграмма, линия

Содержимое, созданное искусственным интеллектом, может быть неверным.  
Рисунок 2 – Схема QA-системы на основе коллекций вопросов и ответов

### Экспертные вопросно-ответные системы

Экспертные вопросно-ответные системы работают на основе структурированной базы знаний, где вопросы анализируются и решаются с помощью фактов и правил. Такие системы обычно узкоспециализированы из-за сложности создания и поддержки базы знаний, которая требует достоверной информации и участия экспертов.

**Преимущество** – высокая точность ответов.

**Недостатки** – необходимость большой базы знаний и привлечения специалистов.

На рисунке 3 представлена схема экспертной QA-системы.

Изображение выглядит как текст, диаграмма, линия, снимок экрана

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 3 – Схема экспертной QA-системы

### Вопросно-ответные системы с коллекцией индексированных документов

В настоящее время получил развитие подход к разработке QA-системс коллекцией индексированных документов.

Вопросно-ответные системы с индексированной коллекцией документов строят специальные индексы, используя именованные сущности и синтаксические связи, выделяемые методами обработки естественного языка. Это позволяет эффективно находить релевантные документы по запросу пользователя.

**Преимущества** – самостоятельное управление индексированием и быстрый поиск.

**Недостатки** – высокие вычислительные затраты на индексацию и необходимость менять алгоритмы поиска при изменении индекса.

На рисунке 4 представлена схема QA-системы на основе коллекций индексированных документов.

Изображение выглядит как текст, диаграмма, снимок экрана, линия

Содержимое, созданное искусственным интеллектом, может быть неверным.  
Рисунок 4 – Схема QA-системы на основе коллекций индексированных документов

Данный тип основывается на подходе RAG.

## RAG

**Retrieval-Augmented Generation (RAG)** – это архитектурный подход, который объединяет две фундаментальные идеи: поиск релевантной информации и генерацию новых ответов на её основе. В отличие от традиционных методов, где система либо просто ищет существующие фрагменты текста, либо генерирует ответы, опираясь только на свои внутренние знания, RAG сочетает оба принципа, что позволяет создавать более точные, информативные и актуальные ответы [2,3].

**Шаги разметки RAG:**

1. Сбор и подготовка данных.

* Сбор источников (документы, статьи)
* Очистка данных
* Разделение на фрагменты

1. Аннотирование и структурирование данных

* Аннотация: маркировка фрагментов по темам, категориям и т.п.
* Извлечение сущностей и связей

1. Индексация и векторизация
2. Построение RAG

Основными шагами в RAG являются:

* Разделение на фрагменты – текст делится на текстовые блоки как правило одинаковой длины.
* Извлечение сущностей и связей во всех текстовых блоках.
* Индексация и векторизация, позволяющая сформировать структуру для поиска по векторным представлениям текста.

**Построение RAG:**

* **Извлечение (Retrieval)**

Система анализирует пользовательский запрос и ищет наиболее релевантные фрагменты информации во внешних источниках –например, в базе документов, корпоративных архивах или открытых данных. Этот поиск может быть реализован с помощью различных алгоритмов, включая векторный поиск, ключевые слова или работу с графами знаний [2,4].

* **Дополнение (Augmentation)**

Найденные фрагменты не используются напрямую, а интегрируются в дальнейший процесс. То есть, система «дополняет» исходный запрос пользователя свежей, контекстуально подходящей информацией. Это позволяет учесть как специфику запроса, так и актуальные данные, которые могли появиться уже после обучения базовой модели [4,5].

* **Генерация (Generation)**

Система формирует ответ, используя как внутренние знания, так и найденную внешнюю информацию. Такой подход позволяет не только избегать устаревших или неточных ответов, но и давать развернутые, аргументированные отклики, опираясь на реальные источники [3].

**Почему RAG – это не только про машинное обучение**

Концепция RAG выходит за пределы чисто алгоритмических или технических решений. Она фокусируется на том, как человек получает знания: сначала ищет информацию, затем осмысливает её и только после этого формулирует ответ. В этом смысле RAG – это попытка приблизить искусственные системы к естественным когнитивным процессам:

* **Гибкость и адаптивность**

Система может оперировать как внутренними знаниями, так и внешними источниками, что делает её более универсальной и способной быстро адаптироваться к новым данным без необходимости полного переобучения [5].

* **Контекстуальность и прозрачность**

Ответы формируются не только на основе «заготовленных» знаний, но и с учётом актуального контекста, что важно для доверия и прозрачности: пользователь может узнать, на каких данных основан ответ [6].

* **Снижение риска ошибок**

За счёт привлечения внешних источников система снижает вероятность ошибок, связанных с устаревшей или неполной информацией, и может ссылаться на конкретные документы или фрагменты текста [5].

* **Применимость и роль в современных системах**

RAG особенно востребован в задачах, где важна не только скорость, но и качество ответа: интеллектуальные поисковые системы, корпоративные помощники, диалоговые интерфейсы, автоматизация поддержки пользователей. Такой подход позволяет строить системы, которые не просто «угадывают» ответ, а действительно опираются на актуальные и проверяемые данные [3,5].

В целом, концепция RAG – это не столько про отдельные алгоритмы или машинное обучение, сколько про интеграцию поиска и генерации как единого интеллектуального процесса, приближенного к реальному человеческому мышлению [6].

## Общий подход к извлечению именованных сущностей из текста

**Именованная сущность (Named Entity)** – последовательность слов, являющихся важными в рамках текстового документа.

В начале для извлечения именованных сущностей из текста рассмотрим общий подход.

Современные алгоритмы чаще всего состоят из 3 последовательных шагов, приведённых ниже [7]:

1. Поиск кандидатов в именованные сущности.
2. Выделение признаков.
3. Ранжирование и отсечение.

Так же обычно выделенные именованные сущности сразу классифицируют на личности, организации, даты и прочее, но в рамках данной работы это не является важным, так как при разметке текстовых документов важно выделить как можно больше информации.

Разберём каждый из шагов алгоритма, описанного выше, подробнее.

### Поиск кандидатов в именованные сущности

Этап поиска кандидатов в именованные сущности можно разделить на несколько задач [7, 18]:

1. Предобработка слов.
2. **Сттемминг** – это грубый эвристический процесс, который отрезает «лишнее» от корня слов, часто это приводит к потере словообразовательных суффиксов [8].
3. **Лемматизация** – это более тонкий процесс, который использует словарь и морфологический анализ, чтобы в итоге привести слово к его канонической форме – лемме [8].
4. **Выделение коллокаций** – это процесс автоматического или полуавтоматического обнаружения в тексте устойчивых словосочетаний, которые имеют семантическую целостность и часто употребляются вместе.
5. **Устранение неоднозначностей (Disambiguation)** – это процесс определения точного значения слова или фразы в контексте, когда они могут иметь несколько интерпретаций. Эта задача особенно важна в обработке естественного языка (NLP), так как многие слова являются омонимами (имеют разные значения) или полисемантичны (многозначны).
6. Разделение на словосочетания (в том числе односложные)
7. **Токенизация**. Базовый этап – разбиение текста на отдельные слова (токены) и предложения. Это позволяет работать с текстом на уровне элементарных единиц и формировать из них более сложные структуры [8].
8. **Использование N-грамм**.Один из самых распространённых методов выделения словосочетаний – построение N-грамм. N-грамма – это последовательность из N подряд идущих слов. Например, биграммы (2-граммы) - пары слов, триграммы - тройки и так далее. Этот подход позволяет автоматически выделять часто встречающиеся устойчивые словосочетания, которые могут быть именованными сущностями [9].
9. **Скользящее окно**. Более сложный способ – использование скользящего окна заданной ширины (например, 2–3 слова), в рамках которого формируются все возможные комбинации словосочетаний. Этот метод позволяет учесть лексический контекст и выявить наиболее значимые сочетания [7].
10. **Частеречная фильтрация**. Для повышения качества выделения фраз часто применяется фильтрация по частям речи. Например, выделяют только те N-граммы, которые соответствуют определённым грамматическим шаблонам (например, прилагательное + существительное, существительное + существительное), что позволяет отсеять случайные или малозначимые сочетания [7].

### Выделение признаков

На втором этапе для каждого выбранного кандидата в именованные сущности определяется набор характеристик, позволяющих впоследствии оценить их значимость. Эти характеристики условно делят на три основные группы [7, 18]:

* **Синтаксические признаки.**

В эту категорию входят сведения о частях речи, которые получают с помощью POS-теггинга (разметки по частям речи), а также дополнительные данные из специализированных лексических ресурсов и онтологий. Следует отметить, что данный подход зависит от языка анализируемого текста.

* **Статистические признаки.**

Здесь учитывается частота появления именованной сущности как в отдельном тексте, так и в коллекции текстов. Кроме того, анализируется длина сущности и её схожесть с другими кандидатами. Как правило, чем чаще сущность встречается в тексте, тем выше её значимость.

* **Структурные признаки.**

Значимость именованной сущности может определяться её расположением в структуре документа. Ключевые слова часто встречаются в заголовках или в первых абзацах. Для количественной оценки структурного признака используют, например, относительное положение слова в тексте – рассчитывают долю слов от начала текста до рассматриваемого словосочетания по отношению к общему объёму текста.

### Ранжирование и отсечение

Третьим этапом извлечения ключевых слов является ранжирование и отсечение. На данном этапе с помощью полученных признаков кандидатов в ключевые слова осуществляется их отбор. Обычно применяют один из двух подходов или использование каких-либо эвристических формул, которые позволяют определить, является ли слово ключевым, или использование методов машинного обучения. Стоит отметить, что для машинного обучения с учителем необходим предварительно размеченный корпус документов с выделенными ключевыми словами. Изначально применение машинного обучения для выделения ключевых слов сводилось к решению задачи бинарной классификации путем различных подходов к обучению классификатора. Использовались наивные байесовские классификаторы, деревья принятия решений, бустинг. Однако такой подход не позволял сравнивать найденные ключевые слова друг с другом и выбирать лучшие из них. Поэтому, впоследствии стали применяться алгоритмы, позволяющие ранжировать ключевые слова попарно (например, алгоритм KEA) [7, 18].

Описанные выше этапы, применяемые для выделения именованных сущностей приведены на рисунке 5.

Изображение выглядит как текст, снимок экрана, диаграмма, линия

Контент, сгенерированный ИИ, может содержать ошибки.  
Рисунок 5 – Этапы извлечения ключевых слов

## Подходы к извлечению именованных сущностей

Способы решения задачи извлечения именованных сущностей (NER) можно классифицировать по подходам, на которых они основываются:

* Статистичекий.
* Лингвистический.
* Машинное обучение.

### Статический подход

**Статистический подход** основан на извлечении слов и словосочетаний с учетом их статистической значимости в текстах, придавая приоритет существительным и именным словосочетаниям.

Методы извлечения именованных сущностей из сегментированных текстов чаще всего базируются на таких хорошо зарекомендовавших себя алгоритмах, как TF-IDF, «YAKE!», TextRank. Рассмотрим подробнее каждый из алгоритмов.

#### TF-IDF

**TF-IDF** – статистический показатель, применяемый для оценки важности слова в контексте категории, документа или коллекции документов. Используется при анализе текстовых данных.

Как правило, TF-IDF определяется для каждого слова. Чем выше значение данного показателя, тем значимее слово в контексте категории, документа, коллекции. При этом данный показатель также позволяет учесть и широко употребляемые слова, понизив их значимость в контексте объекта для анализа.

Формула для определения показателя имеет следующий вид:

где – частота слова в конкретной документе, – обратная частота документа (популярность слова).

Частота слова в категории определяется по формуле:

где количество отдельных слов в документе, общее количество всех слов в документе.

Обратная частота документа (также часто называют инверсией частоты) определяется по формуле:

где – количество документов всего, – количество документов, в которых содержится интересующее слово.

Первый компонент формулы для вычисления TF-IDF фактически всегда не меняется. Метод расчета инвариации частоты может различаться в зависимости от специфики задачи, объема данных для анализа, количества категорий. При этом основной смысл показателя остается без изменений, и он позволяет снизить «вес» широко употребляемых слов.

При анализе текстовых данных метрику TF-IDF лучше всего рассчитывать после проведения процессов [токенизации](https://wiki.loginom.ru/articles/tokenization.html), а также [лемматизации](https://wiki.loginom.ru/articles/lemmatisation.html) или [стемминга](https://wiki.loginom.ru/articles/stemming.html) [12].

**Плюсы TF-IDF:**

* Отсутствие необходимости в обучении:

YAKE! не требует предварительного обучения на больших объемах данных, что упрощает его использование и настройку.

* Эффективность в выявлении именованных сущностей:

TF-IDF выделяет значимые слова, часто встречающиеся в документе, но редко – в других документах корпуса, что помогает выявлять наиболее важные именованные сущности.

* Язык-независимость:

Метод TF-IDF применим к текстам на различных языках без необходимости адаптации.

**Минусы TF-IDF:**

* Игнорирование контекста:

TF-IDF не учитывает контекст использования слов, что может приводить к выделению именованных сущностей, важных только с точки зрения частоты, но не смысла.

* Неэффективность с короткими текстами:

TF-IDF менее эффективен при анализе коротких текстов, где статистическая значимость слов сложно оценить из-за малого объема данных.

* Неэффективность в выделении именованных сущностей:

TF-IDF без модификаций способен выделять только единичные слова, хотя именованными сущностями чаще всего являются не отдельно взятые слова, а словосочетания.

#### YAKE!

В данном методе используется стандартная для текстового анализа методика выделения слов и фраз с помощью токенизации. Фактически такая методика позволяет проверить все сочетания слов на их важность, а не только разделенные стоп-словами. YAKE! собирается из 5 отдельных метрик.

**Casing**

Метрика Casing основана на идее о том, что ключевые слова зачастую могут быть названиями или аббревиатурами. Она измеряет количество раз, когда слово в тексте встречается с большой буквы или является аббревиатурой (написано полностью большими буквами).

где:

* – количество раз, когда слово начинается с большой буквы.
* – количество раз, когда слово отмечается алгоритмом, как аббревиатура (состоит из больших букв).
* – общая частота слова.

**Word Position**

Авторы утверждают, что ключевые слова чаще стоят в начале текста. Из-за этого вводится метрика Word Position, которая учитывает положение слова относительно других.

где – множество позиций слова в документе.

**Word Frequency**

В данном случае частота слова нормируется с учетом среднего и стандартного отклонения частоты:

где – частота слова в тексте.

**Word Relatedness to Context**

Авторы утверждают, что данная метрика способна оценивать, насколько слово похоже на стоп-слово – насколько оно важно для контекста. Метрика использует количество слов, появляющихся слева и справа от слова-кандидата.

где:

* – отношение количества слов слева от кандидата к количеству всех слов, которые появляются вместе с ним.
* – отношение количества слов справа от кандидата к количеству всех слов, которые появляются вместе с ним.
* – отношение количества разных слов, которые появляются слева от кандидата к .
* - отношение количества разных слов, которые появляются справа от кандидата к .

Утверждается, что стоп-слова имеют высокое значение метрики .

**DifSentence**

Эта метрика учитывает количество предложений, в которых используется слово-кандидат.

где:

* SF(w) – частота появления слова в предложениях.
* # Sentences – количество предложений в тексте.

Далее происходит сортировка ключевых слов по этой метрике и выбирается k наиболее значимых.

**Итоговая метрика**

Итоговая метрика составляется из описанных выше метрик.

Далее происходит сортировка ключевых слов по этой метрике и выбирается k наиболее значимых [13].

**Плюсы алгоритма YAKE!**

* Отсутствие необходимости в обучении:

YAKE! не требует предварительного обучения на больших объемах данных, что упрощает его использование и настройку.

* Локальная значимость:

Метод фокусируется на анализе одного документа, оценивая значимость слов в контексте данного текста, а не всего корпуса. Это позволяет лучше выявлять ключевые слова, специфичные для каждого документа.

* Учет различных факторов:

YAKE! использует несколько характеристик, таких как частота встречаемости слова, его положение в тексте, контекстные зависимости и другие, что обеспечивает более точное извлечение ключевых слов.

* Язык-независимость:

YAKE! применим к текстам на различных языках без необходимости адаптации, так как он основывается на общих характеристиках текстов.

**Минусы алгоритма YAKE!**

* Ограниченная точность для сложных текстов:

YAKE! может иметь ограничения при работе с текстами, где требуется глубокое понимание контекста и семантики. Он может не всегда правильно выделять ключевые слова в сложных и многозначных контекстах.

* Зависимость от параметров:

Эффективность метода может зависеть от настроек и параметров, таких как количество ключевых слов, что требует экспериментирования для оптимальных результатов.

#### TextRank

Метод TextRank наиболее сильно отличается от двух предыдущих. Он использует идею, что любой текст можно представить в виде графа, где слова являются вершинами, а связи между ними – ребрами графа. После переведения текста в графовое представление используется классическая метрика важности вершин графа PageRank.

**Построение графа**

Для построения графа вокруг каждого слова берется контекст – берутся все слова, которые находятся на расстоянии n-слов от главного. Например, для контекста размера 2 берутся два слова слева и два слова справа от текущего. Все слова в контексте текущего связываются с ним ребрами графа.

**PageRank**

Рассмотрим метрику, которая используется для выделения важных вершин на графе.

где:

* – важность i-ой вершины.
* – множество вершин, имеющих входящие в i-ую вершину ребра.
* – множество вершин, связанных с i-ой вершиной исходящими из нее ребрами
* – коэффициент затухания, выбирается пользователем

Важность инициализируется случайными числами и потом итеративно сходится к правильным значениям. Таким образом, важность слова определяется связью с другими важными словами [13].

**Плюсы TextRank**

* Отсутствие необходимости в обучении:

TextRank не требует предварительного обучения на больших объемах данных, что упрощает его использование и настройку.

* Учет контекста и связей:

Метод строит граф на основе текстовых данных, где узлы представляют собой слова или фразы, а ребра – их связи и соавторство в тексте. Это позволяет учитывать контекст и семантические связи между словами.

* Язык-независимость:

TextRank может быть применен к текстам на различных языках без необходимости адаптации, так как он основан на универсальных принципах графовых моделей.

**Минусы TextRank**

* Неэффективность с короткими текстами:

TextRank может быть менее эффективен для очень коротких текстов, где недостаточно данных для построения значимого графа.

* Время выполнения:

Построение и обработка графа могут требовать значительных вычислительных ресурсов и времени для очень больших текстовых корпусов, что ограничивает применение метода в реальном времени. Однако, он всё равно будет работать быстрее чем алгоритмы, основанные на машинном обучении.

* Зависимость от параметров:

Эффективность TextRank может зависеть от настроек параметров, таких как размер окна для создания связей между словами, что требует экспериментирования для оптимальных результатов.

* Игнорирование многозначности слов:

Как и многие другие статистические методы, TextRank не учитывает многозначность слов, что может приводить к выделению нерелевантных именованных сущностей в сложных и контекстно насыщенных текстах.

TextRank является мощным и универсальным методом для извлечения ключевых слов и фраз, который не требует предварительного обучения и учитывает контекстные связи между словами. Однако его ограниченная эффективность с короткими текстами и зависимость от параметров могут потребовать дополнительных усилий для настройки и оптимизации.

#### Выводы

Все три алгоритма решают одну и ту же задачу с разных сторон и с использованием разной логики. Результаты работы алгоритмов из-за этого отличаются. Нельзя однозначно сказать, какой из них лучше решает конкретную задачу. На отдельной задаче имеет смысл тестировать качество каждого из алгоритмов и делать выбор исходя из этого.

### Лингвистический подход

Лингвистический подход основывается на поиске именованных сущностей по заданным шаблонам. В качестве таких шаблонов могут выступать, например, последовательности существительных **«**голос солистки**»**, **«**подача блюд**»**, пары прилагательное-существительное **«**живая музыка**»**, **«**апельсиновый сок**»**, последовательности существительных и предлогов **«**блюда из мяса на мангале**»** и т.д.

Такие шаблоны могут быть сформированы заранее или построены автоматически. Например, в работе [14] определялись части речи слов и извлекались фразы, состоящие из двух слов, соответствующие шаблонам. Для автоматического конструирования шаблонов в тексте находятся слова близкие к заранее заданной категории, а затем рассматриваются их соседние слова и определяются их части речи, синтаксические связи. Конструкции, имеющие наибольшую частоту, принимается в качестве шаблонов.

Однако такой подход зачастую выделяет слова, не относящиеся к именованным сущностям. Для фильтрации кандидатов в именованные сущности можно использовать статистические характеристики.

Так же можно использовать набор правил для дополнения уже извлеченных именованных сущностей, например методами машинного обучения.

Тем не менее, есть минус, при использовании такого подхода с шаблонами, нужно заранее прописать эти шаблоны, что усложняет процесс предварительной настройки такого решения, ведь это прямым образом будет влиять на результат.

### Машинное обучение

В последнее время широкое признание получили методы, в основе которых лежит использование алгоритмов машинного обучения. Как правило, они подразумевают два этапа: извлечение цепочек слов, которые потенциально могут быть именованными сущностями, и дальнейшее уточнение их границ.

В общем случае тексты делятся на значимые интервалы, часто совпадающие с отдельными словами и называемые токенами. Каждому токену соответствует вектор. С множеством таких векторов далее работает модель. В зависимости от специфики задачи при вычислениях весов модели делается упор на предсказание либо токена в контексте, либо контекста для токена. Предварительно обученная на объемном корпусе текстов модель выделяет сущности-кандидаты, которые далее могут быть верифицированы. Для определения, является ли последовательность слов именованной сущностью, могут быть использованы разные признаки: общелингвистическая информация (частеречная принадлежность слов, главное слово фразы, количество имен существительных во фразе и др.), статистические (длина фразы, ТЕ, IDF, TF-IDF или частота встречаемости фразы в корпусе научных текстов) и гибридные признаки.

Применяя методы машинного обучения для решения задачи извлечения терминологии из текстов, можно их расширять и совершенствовать.

**CNN**

Традиционно многие системы выделения именованных сущностей основаны на гибридном подходе: сначала применяется лингвистическая/статистическая фильтрация для определения правдоподобных сущностей-кандидатов, затем кандидаты оцениваются и классифицируются с использованием статистических функций, специальных метрик или машинного обучения [15].

Так как стоит задача определения именованных сущностей, то очень важно чтобы выбранный метод машинного обучения учитывал контекст. Под контекстом можно понимать фиксированное количество соседних слов (например, рассматривать предыдущих слов и последующих слов). В такой интерпретации для классификации можно использовать CNN (свёрточную нейронную сеть), на вход которой подавать векторные представления слов [11].

**RNN**

Предложение представляет собой упорядоченный набор слов, и эта последовательность имеет произвольную длину. Для работы на таких данных широко применяются RNN (рекуррентные нейронные сети). RNN в отличие от сетей прямого распространения (многослойный перцептрон, CNN) имеют рекуррентные связи, т.е. обратные связи между нейронами одного или различных слоев. За счет них сети получают некоторое внутреннее состояние, в котором сохраняется и обновляется информация о контексте. Это внутреннее состояние отражает информацию о предыдущих элементах последовательности, причем количество этих элементов не фиксировано.

**Transformers**

Наиболее современным и эффективным подходом к обработке последовательностей, включая задачи извлечения терминологии, является использование архитектуры трансформеров. В отличие от RNN, которые последовательно обрабатывают элементы последовательности, трансформеры работают с последовательностями целиком, благодаря чему они могут учитывать долгосрочные зависимости и параллельно обрабатывать данные. Основу трансформеров составляет механизм внимания (attention), который позволяет модели фокусироваться на значимых частях входной последовательности при обработке каждого слова.

Механизм внимания позволяет модели трансформеров взвешивать важность каждого токена в контексте других токенов в последовательности, что делает их особенно полезными для задач, где контекст играет ключевую роль. Эти модели, как например BERT и GPT, предварительно обучаются на больших корпусах текстов, что позволяет им эффективно захватывать и использовать информацию о языке для выполнения различных задач, включая извлечение терминологии.

Благодаря возможности параллельной обработки и высокой эффективности механизма внимания, трансформеры могут обрабатывать большие объемы данных быстрее и точнее, чем традиционные методы. Это делает их предпочтительным выбором для многих современных приложений в области обработки естественного языка. Кроме того, трансформеры могут быть дополнительно дообучены на специализированных корпусах текстов, что позволяет адаптировать их под конкретные задачи и домены, обеспечивая высокую точность и релевантность извлеченных именованных сущностей.

### Применимость подходов

Каждый из подходов по-своему решает поставленную задачу.

Так, статический подход отлично подходит для решения задачи при необходимости быстрого, по сравнению с другими подходами, выделения именованных сущностей. Однако, из-за того, что такой подход опирается исключительно на статистические особенности, на специфических текстах он может выдавать неверные результаты.

Лингвистический подход хорошо себя показывает в совокупности с другими подходами и сразу “из коробки” может выделять связанные словосочетания.

Машинное обучение показало себя наиболее хорошо в перспективе благодаря механизму внимания, который позволит учитывать только наиболее актуальную информацию. Однако, для такого подхода необходимо большое количество сформированных токенов, каждому из которых соответствует вектор. Для того, чтобы формировать такие токены с векторами, можно воспользоваться статистическим подходом, который не требует настройки и поможет автоматизировать этот процесс.

## Выводы

В первой главе проведён комплексный анализ методов автоматической разметки текстовых документов для вопросно-ответных систем на основе архитектуры RAG.

Рассмотрены три основных подхода к извлечению именованных сущностей: статистический, лингвистический и методы машинного обучения. В результате анализа выявлены следующие особенности:

* **Статистический подход** обеспечивает быструю обработку больших объёмов данных, не требует предварительного обучения и хорошо адаптируется к различным тематикам.
* **Лингвистический подход** позволяет учитывать синтаксическую структуру и выявлять сложные терминологические конструкции, что повышает качество извлечения релевантных фрагментов для последующей генерации ответа.
* **Методы машинного обучения** демонстрируют высокую точность, но требуют наличия больших размеченных корпусов, что не всегда реализуемо.

С учётом требований к скорости разработки, отсутствия размеченных данных, для дальнейшей работы были выбраны **статистический и лингвистический подходы**. Их комбинация позволяет эффективно формировать базу знаний для RAG-системы, обеспечивая релевантность и полноту извлекаемой информации без существенных затрат на обучение моделей.

# Анализ предметной области «Разметка текстовых документов»

Данная глава содержит анализ предметной области «Разметка текстовых документов» и задач, решаемых в рамках системы.

## Термины предметной области

**Текстовый документ** – это форма представления информации, где информация записана в текстовом формате в электронном виде или на другом материальном носителе. Это может быть любое произведение, написанное с помощью текстового редактора, включая статьи, отчеты, письма и многое другое [14, 15].

**TF-IDF** – это мера, отражающая важность слова в документе относительно корпуса текстов [16].

**Токен** – это отдельная единица текста, обычно слово, полученное в результате процесса токенизации, то есть разбиения текста на отдельные слова или символы для дальнейшего анализа. Именно для каждого токена (слова) вычисляется значение TF-IDF, отражающее его значимость в документе относительно всего корпуса текстов [17].

**Словосочетание** – последовательность слов, состоящая из одного или более слов, связанных по смыслу и грамматически.

**Именованная сущность** – это словосочетание в тексте, обозначающие уникальные объекты реального мира: имена людей, названия организаций, географические объекты и другие именования [19].

**n-грамм** в контексте TF-IDF – это последовательность из n слов, следующих друг за другом в тексте.

**Существительное** – это самостоятельная часть речи, которая обозначает предмет, лицо или явление и отвечает на вопросы «кто?» или «что?» [18].

**Прилагательное** – это самостоятельная часть речи, которая обозначает признак предмета и отвечает на вопросы «какой?», «какая?», «чей?» и так далее.

**Тип словосочетания** – это характеристика структуры и грамматической организации словосочетания, отражающая морфологические и синтаксические свойства главного и зависимых компонентов, а также их количество и форму связи.

**Шаблон** – синтаксическая схема построения словосочетаний [18].

**Термин** – название определённого понятия, какой-либо предметной области, может выступать как отдельным словом, так и словосочетанием [18].

**Синоним термина (синоним)** – это словосочетание, имеющее то же или близкое значение, что и данный термин, и используемое для обозначения того же понятия в рамках определённой предметной области.

**Значение термина (значение)** – это содержание или смысл, который вкладывается в термин, то есть совокупность признаков и свойств, определяющих соответствующее понятие в данной предметной области.

**Словарь** – совокупность именованных сущностей, а также терминов и его синонимов/значений.

**Косинусная схожесть** – это мера сходства между двумя векторами, основанная на косинусе угла между ними. Она показывает, насколько направления векторов совпадают, и не зависит от их длины. Значение косинусной схожести варьируется от 0 (векторы ортогональны, то есть не имеют общих признаков) до 1 (векторы полностью совпадают по направлению)

## Анализ профессиональной деятельности

Задачей разметки текстовых документов является предобработка текстовых документов. Этот этап позволяет перевести неструктурированную информацию в структурированный вид и как следствие упростить задачу поиска информации в текстовых документах.

Категории пользователей:

**Специалисты** – группа пользователей, которая может инициировать извлечение именованных сущностей для текстовых документов и в последствии формировать словари на их основе для последующего поиска релевантных терминов с их позициями в исходном текстовом документе.

**Задачи специалистов:**

* Инициация выделения именованных сущностей из текстового документа.
* Формирование словаря – создание/редактирование и объединение словарей на основе полученных именованных сущностей.
* Просмотр списка словарей.
* Поиск по терминам.
* Поиск пересечений терминов.

**Первоначальной задачей специалистов** является разметка текстовых документов для последующего поиска релевантных словосочетаний в словарях с их позициями в исходных предложениях текстов.

Разметка текстовых документов состоит из выделения именованных сущностей и формирования словаря.

**Извлечение именованных сущностей (или анализ)** происходит в момент, когда специалист предоставляет системе текст или текстовый документ для разметки. Этот этап состоит из:

* **Разбиение** **исходного текста** на блоки меньшей длины чем исходный документ.
* **Извлечение кандидатов в именованные сущности**

Для выделения кандидатов необходимо выполнить несколько шагов:

* Токенизация текста

Выделяемые из текстов токены состоят из набора символов русского, латинского алфавита, а также из цифр и символа тире.

* Генерация n-грамм

Для каждого выделенного токена берётся «окно» из n токенов. Окно размером 1, 2, 3.

* Определение типа словосочетания

Для каждой n-грамм определяется относится ли она к поддерживаемому типу словосочетания: однословное, адъективное и так далее. Этот этап был доработан по сравнению с ВКР Сидельниковой Маргариты «Разработка системы для извлечения терминов из коллекции текстовых документов» и теперь учитывает падежи, что позволяет учитывать главное слово в словосочетании [18]. Более подробно про возможные типы словосочетаний будет описано далее в этой же главе.

* Лемматизация
* **Расчёт TF-IDF**

Для каждого выделенного кандидата в именованную сущность вычисляется его частота TF-IDF.

* **Ранжирование**

Полученные на предыдущем шаге именованные сущности сортируются по убыванию TF-IDF-весов, после чего выбираются верхние 10000.

Извлечение кандидатов в именованные сущности для расчёта TF-IDF и его расчёт для них в дальнейшем будем называть *компонентой retrieval*.

**Результат выделения именованных сущностей** в виде непосредственно выделенных словосочетаний с их типами словосочетаний и значениями TF-IDF для каждого, а также исходный текст и блоки, на которые текст был разбит, сохраняется для дальнейшего формирования словаря и последующего поиска по размеченному текстовому документу.

Выделенные словосочетания могут быть одним из следующих типов [18]:

* **Однословное –** отдельное слово, не образующее сочетания с другими словами.
* **Адъективное –** словосочетание, состоящее из прилагательного и существительного. Главным в этом словосочетании является существительное. Это самая распространённая модель словосочетания.Пример представлен на рисунке 6.

Изображение выглядит как снимок экрана, текст, черный, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.  
Рисунок 6 – Адъективное словосочетание

* **Адъективное многословное –** словосочетание, состоящее из двух прилагательных и одного существительного. Существительное является в этом словосочетании главным словом, а прилагательные зависимыми. Пример представлен на рисунке 7.

Изображение выглядит как снимок экрана, текст, черный, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.  
Рисунок 7 – Адъективное многословное словосочетание

* **Генитивное –** словосочетание, состоящее из двух существительных. Зависимое слово стоит в родительном падеже. Пример представлен на рисунке 8.

Изображение выглядит как снимок экрана, текст, черный, Графика

Контент, сгенерированный ИИ, может содержать ошибки.  
Рисунок 8 – Генитивное словосочетание

* **Генитивное многословное –** словосочетание, состоящее из трёх существительных, несколько из которых являются зависимыми в родительном падеже. Пример представлен на рисунке 9.

Изображение выглядит как снимок экрана, черный, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.  
Рисунок 9 – Генитивное многословное словосочетание

* **Адъективно-генитивное –** словосочетание,состоящее из двух существительных, разделённых прилагательным. Прилагательное и зависимое слово в родительном падеже одновременно определяют главное слово. Пример представлен на рисунке 10.

Изображение выглядит как снимок экрана, текст, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.  
Рисунок 11 – Адъективно-генитивное словосочетание

* **Генитивно-адъективное** – словосочетание, состоящее из прилагательного и двух существительных. Главное слово связано с зависимым в родительном падеже и при этом дополнительно определяется прилагательным. Пример представлен на рисунке 12.

Изображение выглядит как снимок экрана, текст, черный, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.  
Рисунок 12 – Генитивно-адъективное словосочетание

После формирования именованных сущностей происходит **ручная разметка**. Она состоит из создания/редактирования словаря, а также из объединения двух словарей в один.

Создание словаря:

При создании словарь содержит отсортированные в алфавитном порядке именованные сущности. При редактировании словарь уже содержит отсортированные в алфавитном порядке не только именованные сущности, но и термины, а также его синонимы и значения.

* **Установка порогового значения TF-IDF**

Словосочетания ниже этого порога не попадут в конечный словарь.

* **Формирование словаря**

На основе именованных сущностей, выделенных во время этапа выделения именованных сущностей, пользователь определяет термины, а также его синонимы и значения, относящиеся одновременно только к одному термину.

После этого можно либо объединить словарь с существующим, либо сохранить текущий.

Редактирование словаря:

При редактировании словарь уже содержит отсортированные в алфавитном порядке не только именованные сущности, но и термины, а также его синонимы и значения.

Все прочие пункты аналогичны созданию словаря за исключением того, что действия совершаются над уже созданным ранее словарём.

Объединение словарей

Объединение словарей происходит посредством объединения именованных сущностей и терминов, а также их синонимов и значений. При их объединении в один словарь приоритет имеет целевой для объединения словарь. К примеру, если в целевом словаре словосочетание уже присутствовало, то словосочетание из объединяемого словаря не попадёт в целевой, как и его связи (синонимы и значения). Пороговое значение TF-IDF в итоговом словаре будет равно нулю.

Нужно отметить, что объединяемые словари могли быть сформированы из разных текстовых документов. В таком случае последующий поиск словосочетаний и их места встречи в исходном тексте будут происходить во всех текстовых документах, которые после объединения ассоциируются с новообразованным словарём.

**После окончания всех этапов** разметки текстовых документов специалист может осуществить поиск по размеченным данным.

Процедуру поиска после того, как специалист ввёл искомое словосочетание, можно разделить на несколько этапов:

* Поиск последних трёх добавленных словарей.
* Поиск в найденных словарях схожих словосочетаний.

С помощью компоненты retrieval происходит разметка всех терминов, синонимов и значений из текущего обходимого словаря образуя матрицу. То же самое проделывается над словосочетанием-запросом. Далее с помощью косинусной схожести находятся 3 словосочетания с наибольшей степенью схожести, отличной от нуля.

* Поиск предложений, в которых встречались найденные словосочетания.

Этот этап производится над каждым словосочетанием в совокупности со всеми связями (терминами, синонимами, значениями). При помощи всё той же компоненты retrieval для выделенных блоков текста словаря выделяются словосочетания со своими значениями TF-IDF, образуя вместе матрицу. То же самое производится для словосочетаний и так же через косинусную схожесть находятся предложения, в которых с вероятностью больше нуля встречаются эти словосочетания. Таких предложений выбирается только 3 с наибольшим значением схожести.

Помимо поиска по одному словосочетанию система подразумевает возможность поиска пересечений словосочетаний в предложениях. К примеру, если в одном предложении были встречены первое и второе искомое словосочетание, то система выдаст этот текст.

## Модель предметной области

**–** текстовый документ.

– множество блоков текста, на которые разделён исходный текстовый документ, где m – количество блоков в тексте.

, где , где не может стоять раньше .

– множество токенов в текстовом документе, где n – количество различных токенов в тексте.

– кортеж из value – значения, POS – одна из категорий части речи, case – одна из категорий падежа.

Введём условия:

* Любой токен встречается хотя бы в одном блоке текста
* Любой блок текста содержит хотя бы один токен

– поддерживаемые типы словосочетаний.

, где , m – количество слов проверяемого словосочетания.

– кортеж из и .

Всевозможные типы словосочетаний:

– однословное словосочетание

–адъективное словосочетание.

–генитивное словосочетание.

–адъективное многословное словосочетание.

– генитивное многословное словосочетание.

–адъективно-генитивное словосочетание.

–генитивно-адъективное словосочетание.

, где , i – количество слов, из которого состоит словосочетание.

Введём правило нормализации фразы:

* Для каждого токена в Phrase выполняется морфологическое преобразование:

Обозначим это преобразование как функцию

– последовательность для словосочетания, где , n – количество слов в словосочетании.

Cловосочетание кандидат для разметки с помощью TFIDF нужно добавить, что каждый токен фраза нормализуется после определения типа.

–именованная сущность, где TFIDF – важность словосочетания в текстовом документе, подробное описание получения TFIDF описано в методах решения задач.

– словарь, где:

* – название словаря, состоит из символов русского и латинского алфавитов.
* – множество текстовых документов, на основе которых составлен словарь, где n – их кол-во.
* – дата и время создания словаря.
* – пороговое значение отображения словосочетаний словаря, .
* – множество именованных сущностей словаря, где n – их количество. Введём условие:
  + Все именованные сущности разные.
* – множество терминов, где n – их количество.

– кортеж из именованной сущности, выбранной в качестве термина и его синонимов и значений, где:

* – именованная сущность, выбранная в качестве термина, .
* – множество синонимов, где n – их количество.
* – множество значений, где n – их количество.

Введём условия:

* Каждый термин содержит свой набор синонимов и словосочетаний.
* Синонимы и значения термина содержат разные именованные сущности.

## Методы решения задач

**Метод выделения именованных сущностей**

Вход:

Выход:

Описание:

*Шаг 1.* Извлечение из текстового документа Text блоков текста .

*Шаг 2.* Для каждого :

* + Выполняется токенизация (извлечение ).
  + Формирование из токенов n-грамм
  + Получение для каждого (проверка типа словосочетания и нормализация).
  + Получение для из предыдущего шага.

Для этого определим функцию стандартного расчёта TFIDF:

*Подробное описание расчёта не приведено, так как оно имеет стандартный вид TFIDF.*

*Шаг 3.* Ранжирование полученного массива по убыванию значения для каждого ,

*Шаг 4.* Отбор первых 10000 именованных сущностей.

**Метод формирования словаря. Создание словаря.**

Вход:

Выход:

Описание:

*Шаг 1.* , где Text – текст, используемый в методе извлечения именованных сущностей.

*Шаг 2.* Задание названия и порогового значения , задаётся автоматически при создании.

*Шаг 2.* Определение множества на основе .

**Метод формирования словаря. Редактирование словаря.**

Вход:

Выход:

Описание:

*Шаг 1.* Обновление .

*Шаг 2.* Обновление массива .

**Метод формирования словаря. Объединение словарей.**

Вход:

Выход:

Описание:

*Шаг 1.* Определение множества на основе .

*Шаг 2.* Объединение данных текущего словаря с . При возникновении коллизий приоритет в данных отдаётся . проставляется в 0.0, а берётся из словаря .

*Шаг 3.* В текст добавляется текстовый документ из нового словаря.

**Метод поиска по одному словосочетанию**

Вход: , где n – кол-во всех созданных ранее словарей; – словосочетание для поиска терминов.

Выход: , где –искомый термин, а – множество найденных блоков текста, в которых встречается.

Описание:

*Шаг 1.* Поиск последних 3-х добавленных словарей на основе – . Если словарей 0, то вернуть пустое множество, иначе – шаг 2.

*Шаг 2.* Для каждого словаря осуществляется поиск терминов, их синонимов и значений, похожих на .

Пусть – TF-IDF-вектор объекта x.

TF-IDF-вектор формируется на основе корпуса, включающего все исходного текстового документа, и все выделенные кандидаты в именованные сущности.

Схожесть между двумя объектами определяется как косинусное сходство их TF-IDF-векторов:

где:

* – скалярное произведение TF-IDF-векторов,
* – евклидова норма TF-IDF-вектора

Следовательно, для объектов математической модели:

Для и поиск схожих терминов посредством косинусного сходства будет выглядеть:

*Шаг 3.* Ранжирование и отсев терминов.

Полученные термины сортируются в порядке убывания косинусной схожести и отбираются первые 3, у которых сходство отлично от нуля.

*Шаг 4.* Для каждого найденного ищется блок текста , в котором он встречался через косинусное сходство.

*Шаг 5.* Ранжирование и отсев блоков текстов.

Полученные предложения сортируются в порядке убывания косинусной схожести и отбираются первые 3, у которых сходство отлично от нуля.

*Шаг 6.*  вместе с блоками текста, в которых они встречались, записываются в виде

**Метод поиска по нескольким словосочетаниям с их пересечением**

Вход: , где n – кол-во всех созданных ранее словарей; – словосочетания для поиска терминов.

Выход: , где – искомые термины, а – множество найденных блоков текста, в которых множество Term’ов пересекается.

Описание:

*Шаг 1.* Поиск последних 3х добавленных словарей на основе – . Если словарей 0, то вернуть пустое множество, иначе – шаг 2.

*Шаг 2.* Для каждого словаря и для каждого осуществляется поиск терминов, их синонимов и значений, похожих на .

Пусть  – TF-IDF-вектор объекта x.

Схожесть между и каждым термином из словаря определяется как:

*Шаг 3.* Ранжирование и отсев терминов.

, сортируются в порядке убывания косинусной схожести и отбираются первые 3, у которых сходство отлично от нуля.

*Шаг 4.* Для каждого найденного ищется блок текста , в котором он встречался через косинусное сходство.

*Шаг 5.* Ранжирование и отсев блоков текстов.

Полученные предложения сортируются в порядке убывания косинусной схожести и отбираются первые 3, у которых сходство отлично от нуля.

*Шаг 6.* Поиск пересечения в блоках текста множеств найденных терминов по всем .

*Шаг 7.* вместе с блоками текста, в которых встречались их пересечения, записываются в виде

# Проект программной системы

Для ПО «Разметка текстовых документов» необходимо разработать программную систему, содержащую функционал:

1. Извлечения именованных сущностей (NER).
2. Формирования словаря.
3. Поиска.

## Архитектурно-контекстная диаграмма

Для реализации системы «Разметки текстовых документов» была выбрана архитектура web-приложения только с серверной частью.

В процессе проектирования программной системы была разработана архитектурно-контекстная диаграмма, представленная на рисунке 13.

Изображение выглядит как текст, снимок экрана, диаграмма

Содержимое, созданное искусственным интеллектом, может быть неверным.  
Рисунок 13 – Архитектурно-контекстная диаграмма ПС

Жёлтым цветом на диаграмме выделены модули, отвечающие за разметку текстовых документов, а также поиск.

**Компоненты:**

* **Специалист** – конечный пользователь.
* **Интернет** – информационно-коммуникационная сеть, через которую осуществляется взаимодействие с ПС.
* **Веб-сервер** – сервер, принимающий HTTP-запросы от клиентов, обычно от веб-браузеров, и отдающий им HTTP-ответы в виде json данных или html-страниц.
* **Серверная часть** – часть приложения, отвечающая за обработку запросов от web-сервера и логику работы приложения с данными. Содержит контроллеры, сервисы и репозитории.
* **Контроллеры –** компоненты серверной части, принимающие входящие запросы от веб-сервера, обрабатывающие их и передающие данные в соответствующие сервисы для выполнения бизнес-логики.
* **Сервисы** – модули, реализующие бизнес-логику приложения, обрабатывающие данные, полученные от контроллеров, и взаимодействующие с репозиториями для доступа к данным.
* **Репозитории** – компоненты, отвечающие за взаимодействие с базой данных, выполнение операций чтения, записи, обновления и удаления информации.
* **База данных ПС** – централизованное хранилище информации.

## Модуль NER

На рисунке 14 представлена диаграмма метода извлечения именованных сущностей.

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.  
Рисунок 14 – Диаграмма модуля NER

Реализация модуля соответствует методу, описанному в предыдущей главе.

## Модуль формирования словаря

На рисунке 15 представлена диаграмма метода формирования словаря.

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 15 – Диаграмма модуля формирования словаря

Реализация модуля соответствует методу, описанному в предыдущей главе.

## Модуль поиска

На рисунке 16 представлена диаграмма метода поиска.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Содержимое, созданное искусственным интеллектом, может быть неверным.  
Рисунок 16 – Диаграмма модуля поиска

Реализация модуля соответствует методам поиска по одному и нескольким словосочетаниям, описанному в предыдущей главе.

## Диаграмма Use-Case

В рамках данной работы чтобы определить функциональные требования к системе «Разметка текстовых документов» была построена Use-Case диаграмма, представленная на рисунке 17.

Изображение выглядит как текст, снимок экрана, круг, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.  
Рисунок 17 – Use-Case диаграмма  
системы «Разметка текстовых документов»

Исходя из Use-Case диаграммы можно выделить основные функциональные требования:

* Разметка текстовых документов: выделение именованных сущностей и формирование словаря.
* Просмотр списка словарей.
* Поиск.

## Спецификация остальных требований

В данном параграфе описаны требования, которые не были представлены в Use-Case диаграмме.

### Требования к удобству использования ПС

1. ПС должна быть предназначена для специалистов разных направленностей.
2. Все части ПС, доступные для взаимодействия с пользователем, должны быть на русском языке.
3. Тексты сообщений и подсказки должны быть понятными.

### Требования к удобству использования ПС

ПС должна работать в следующих браузерах, версии которых выпущены не раньше 2017 года:

* Opera, Opera GX.
* Google Chrome.
* Microsoft Edge.
* Yandex.
* Mozilla Firefox.

## Проект верхнего уровня

В данном разделе работы будет рассмотрено описание структуры базы данных и проекта интерфейса.

### Структура базы данных

Была выбрана реляционная модель данных.

**Всего выделено 6 объектов базы данных:**

* document;
* analysis\_result;
* dictionary;
* dictionary\_analysis\_result;
* term;
* connection;

На рисунке 18 представлена структура базы данных.

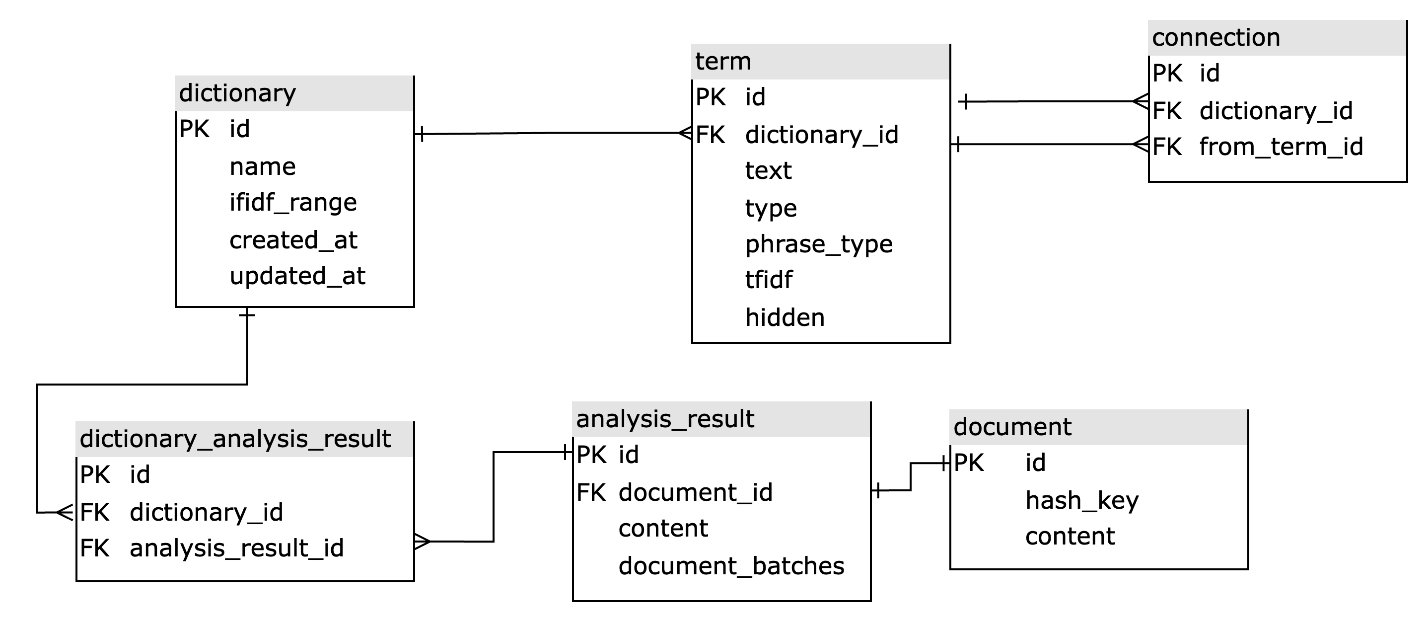
  
Рисунок 18 – UML-диаграмма  
базы данных «Разметка текстовых документов»

Таблица **document** служит для хранения информации о текстовых документах.

**Атрибуты:**

* id – уникальный идентификационный номер (primary key).
* hash\_key – значение, получаемое в результате применения хэш-функции над content.
* content – содержимое текстового документа.

Таблица **analysis\_result** служит для хранения информации о результатах выполнения этапа извлечения именованных сущностей.

**Атрибуты:**

* id – уникальный идентификационный номер (primary key).
* document\_id – внешний ключ document (foreign key).
* content – результат извлечения именованных сущностей в формате json.
* document\_batches – текстовые блоки, сформированные из содержимого текстового документа.

Таблица **dictionary** служит для хранения информации о результатах формирования словаря.

**Атрибуты:**

* id – уникальный идентификационный номер (primary key).
* name – название словаря.
* tfidf\_range – пороговое значение tfidf словаря.
* created\_at – дата и время создания словаря.
* updated\_at – дата и время обновления словаря.

Таблица **dictionary\_analysis\_result** выступает в виде связующего звена между dictionary и analysis\_result (позволяет избавиться от связи многие ко многим).

**Атрибуты:**

* id – уникальный идентификационный номер (primary key).
* dictionary\_id – внешний ключ словаря (foreign key).
* analysis\_result\_id – внешний ключ результатов анализа (foreign key).

Таблица **term** служит для хранения информации обо всех именованных сущностей.

**Атрибуты:**

* id – уникальный идентификационный номер (primary key).
* dictionary\_id – внешний ключ словаря (foreign key).
* text – значение термина.
* type – сущность/термин/синоним/значение.
* phrase\_type – тип словосочетания.
* tfidf – значение tfidf для сущности.
* hidden – нужно ли учитывать сущности при отображении словаря и поиске.

Таблица **connection** служит для хранения информации о связях термина.

**Атрибуты:**

* id – уникальный идентификационный номер (primary key).
* dictionary\_id – внешний ключ словаря (foreign key).
* from\_term\_id – внешний ключ сущности (foreign key).

### Проект интерфейса

Перейдя на сайт ПС, специалист увидит главную страницу.

Изображение выглядит как текст, Шрифт, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 19 – Главная страница

На этой странице он может вбить содержимое текстового документ в текстовое поле, либо загрузить текстовый документ (рисунок 20).

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.  
Рисунок 20 – Загрузка содержимого текстового документа

Нажав на кнопку «Анализировать текст» происходит извлечение именованных сущностей, результаты пользователь увидит на той же странице ниже (рисунок 21).

Изображение выглядит как текст, снимок экрана, число, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.  
Рисунок 21 – Результаты извлечения именованных сущностей

При попытке анализа пустого текста пользователь увидит сообщение об ошибке (рисунок 22).

Изображение выглядит как текст, снимок экрана, веб-страница, Веб-сайт

Контент, сгенерированный ИИ, может содержать ошибки.  
Рисунок 22 – Сообщение о пустом тексте  
или не выбранном файле для анализа

При успешном анализе пользователь может нажать кнопку «Сформировать словарь» – в таком случае он попадает на страницу создания словаря (формирования словаря)– рисунок 23.

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Контент, сгенерированный ИИ, может содержать ошибки.  
Рисунок 23 – Создание словаря (ручная разметка)

С помощью интерфейса, представленного на рисунке 23, пользователь может создать словарь. Для этого ему обязательно нужно задать название. Если он не задаст название и попробует создать словарь, то получит сообщение об ошибке, представленное на рисунке 24.

Изображение выглядит как текст, программное обеспечение, Мультимедийное программное обеспечение, Значок на компьютере

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 24 – Сообщение о попытке сохранить  
словарь с пустым названием

После того как пользователь ввёл название, он может задать порог TF-IDF, который уменьшит количество именованных сущностей на странице (рисунок 25).

Изображение выглядит как текст, снимок экрана, число, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 25 – Изменение порога TF-IDF

После задания порога пользователь может задать термин и его синонимы/значения (рисунок 26).

Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница

Контент, сгенерированный ИИ, может содержать ошибки.Рисунок 26 – Задание терминов и его синонимов/значений

При добавлении нового термина фокус перейдёт на новый и все связи предыдущего термина скроются (рисунок 27).

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Контент, сгенерированный ИИ, может содержать ошибки.Рисунок 27 – Переход фокуса на новый термин

После наполнения словаря пользователь может сохранить словарь или объединить его с существующим. Для начала сохраним словарь (рисунок 28).

Изображение выглядит как текст, снимок экрана, Шрифт, мультимедиа

Контент, сгенерированный ИИ, может содержать ошибки.Рисунок 29 – Уведомление об успешном сохранении словаря

После сохранения словаря пользователь попадает на страницу редактирования словаря (рисунок 30).

Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница

Контент, сгенерированный ИИ, может содержать ошибки.Рисунок 30 – Страница редактирования словаря.

На этой странице можно редактировать словарь, проделав аналогичные действия, что и при создании. Попробуем объединить один словарь с другим, для этого создадим новый словарь и нажмём копку «Объединить» (рисунок 31).

Изображение выглядит как текст, Шрифт, логотип, Цвет электрик

Контент, сгенерированный ИИ, может содержать ошибки.  
Рисунок 32 – Кнопка объединения словаря

Перед пользователем появляется выпадающий список, в котором он может выбрать словарь, с которым хочет объединить текущий (рисунок 33).

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.  
Рисунок 33 – Окно объединения со словарём

При выборе словаря и нажатии на кнопку «Подтвердить» пользователь увидит уведомление (рисунок 34) об успешном объединении, после чего он попадёт на страницу редактирования полученного словаря.

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Контент, сгенерированный ИИ, может содержать ошибки.  
Рисунок 34 – Страницу редактирования словаря  
после объединения

В заголовочной части сайта есть ещё две вкладки «Словари» и «Поиск», перейдём на страницу словарей (рисунок 35).

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Контент, сгенерированный ИИ, может содержать ошибки.Рисунок 35 – Страница списка всех добавленных словарей

На этой странице пользователь видит все ранее добавленные словари, а также может перейти к редактированию конкретного словаря или удалить его. При нажатии на кнопку удалить словарь пользователь увидит сообщение (рисунок 36).

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.  
Рисунок 36 – Сообщение с подтверждением удаления словаря

После этого словарь пропадёт из общего списка.

Теперь перейдём на страницу поиска (рисунок 37).

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.  
Рисунок 37 – Страница поиска

На этой странице пользователь может осуществить поиск по одному или нескольким словосочетаниям. Осуществим поиск по одному словосочетанию, для этого введём его в верхнее поле для ввода и нажмём кнопку «Найти» (рисунок 38)

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.  
Рисунок 38 – Результат поиска по одному словосочетанию

# Реализация и тестирование системы

В данной главе описываются инструменты разработки программной системы «Разметка текстовых документов», приводятся проведенные основные тесты, составленные на основе плана тестирования.

## Технологический стек

Для реализации системы «Разметка текстовых документов» было принято решение разрабатывать web-приложение, что обеспечивает большую гибкость в проектировании пользовательского интерфейса и взаимодействия с пользователем по сравнению с desktop приложениями.

### Frontend

В данном разделе описываются технологии, применяемые при создании клиентской части системы.

* Основным языком разработки выбран JavaScript. От использования специализированных фреймворков для построения пользовательского интерфейса было решено отказаться, чтобы не усложнять архитектуру и сосредоточиться на экспериментальной части работы.
* Для верстки страниц используются HTML и CSS.
* Для ускорения и упрощения разработки интерфейса, а также обеспечения адаптивности и современного внешнего вида, в проекте применён интерфейсный инструментарий Bootstrap 5 [20].

### Backend

В качестве основного языка программирования для backend был выбран Python [21], что обусловлено богатым набором библиотек для обработки естественного языка и современными фреймворками для веб-разработки.

**FastAPI**

В качестве веб-фреймворка выбран FastAPI [22]. FastAPI позволяет легко создавать как REST API для взаимодействия с фронтендом и внешними сервисами, так и серверные маршруты для обработки пользовательских запросов через веб-интерфейс.

**Обработка и анализ текста**

Для выделения типов словосочетаний и лемматизации русского текста применяется библиотека **pymorphy3** [23].

Извлечение именованных сущностей реализовано с помощью **scikit-learn**: класс TfidfVectorizer для вычисления меры TF-IDF; cosine\_similarity для вычисления схожести векторов двух строк [24].

**Шаблонизация и серверный рендеринг**

Для генерации HTML-страниц на серверной части приложения применяется Jinja2 [25] – шаблонизатор, который интегрируется с FastAPI и позволяет динамически формировать html-страницы.

### База данных

В качестве базы данных был выбран postgres из-за поддержки работы с массивами, JSON-данными, а также векторными данными для возможности вычисления сходства строк.

### Сборка проекта

Сборка проекта осуществляется с помощью **Docker Compose**. Это позволяет создавать и запускать приложение с несколькими контейнерами Docker, включая базу данных и бэкенд. Docker Compose использует файл docker-compose.yml для описания конфигурации приложения и запускает все контейнеры в одной сети, что обеспечивает легкую настройку и масштабирование приложения [26].

### Среда разработки

В качестве основной среды разработки использовалась **PyCharm** – интегрированная среда разработки для Python, предоставляющая широкий набор инструментов для автодополнения кода, отладки, тестирования, работы с системами контроля версий и управления зависимостями, а также поддерживающая средства для работы с базами данных, что делает её удобным выбором для разработки приложений на Python.

## Тестирование

В данном подразделе описываются тесты, покрывающий функционал, касающийся модулей разметки и поиска.

Для описания тестов построена таблица, в которой представленная следующая информация:

* Информация о номере тестовой ситуации.
* Информация о тестовой ситуации.
* Информация о входных данных.
* Информация о выходных данных.

Тестовые ситуации описаны в Приложении А (таблица 1).

В результате тестирования, в соответствии с приложением А (таблица 1), было проведено 20 тестов, которые были успешно пройдены. Успешность тестирования говорит о том, что модули работают корректно и соответствуют требованиям, предъявляемым к ним.

## Эксперимент

Ниже будет описан ряд экспериментальных исследований предложенного метода разметки коллекций текстовых документов.

### Эксперимент №1

**Цель эксперимента:** сравнить извлечение именованных сущностей, учитывающее падеж во время выделения кандидатов в именованные сущность с извлечением, которое учитывает только части речи.

**Входные данные:** Сказка Шарля Перо «Красная шапочка» [28].

**Выходные данные:**

После обработки результатов и отбора терминов, значений и синонимов вручную в словарь (формирование словаря) были получены результаты, часть из которых представлена в таблице 2.

Таблица 2 – Часть сформированного словаря для сказки «Красная шапочка» (новый метод).

|  |  |  |
| --- | --- | --- |
| Термин | Синоним | Значение |
| бабушка | бабушка любимый  твой бабушка | старушка |
| болезнь |  | осип |
| букет |  |  |
| верёвочка |  |  |
| красный шапочка | любимый внучка  бедный девочка  послушный девочка  внучка ваш  свой дочка  шапочка  красота невиданный | девочка  внучка  дочка |
| глаз | большой глаз  такой большой глаз |  |
| дорога | короткий дорога  сам короткий дорога  этот дорога |  |

Итого 89 отобранных именованных сущностей для формирования словаря из 96. То есть процент значимых именованных сущностей примерно 93%.

После обработки результатов старым методом были получены результаты, часть из которых представлена в таблице 3.

Таблица 3 – Часть сформированного словаря для сказки «Красная шапочка» (старый метод).

|  |  |  |
| --- | --- | --- |
| Термин | Синоним | Значение |
| бабушка | твой бабушка | старушка |
| болезнь |  |  |
| букет |  |  |
| верёвочка |  |  |
| красный шапочка | любимый внучка  бедный девочка  послушный девочка  внучка ваш  свой дочка  шапочка  красота невиданный | девочка  внучка  дочка |
| глаз | большой глаз |  |
| дорога | короткий дорога  этот дорога |  |

Итого 98 отобранных именованных сущностей для формирования словаря из 116. То есть процент значимых именованных сущностей – 84%.

**Обоснование результатов:**

Если сравнивать новый метод со старым, то прирост отбора релевантных именованных сущностей на рассматриваемом текстовом документе составляет 9%. Такой результат объясняется тем, что при учёте падежей удаётся точнее учитывать главное и зависимые слова в словосочетаниях таких типов как генитивно-адъективное, адъективное многословное и прочих.

### Эксперимент №2

**Целью эксперимента** является демонстрация того, что поиск по размеченным коллекциям документов, осуществляемый с использованием словарей терминов, синонимов и значений, извлечённых с помощью методов извлечения именованных сущностей (NER) и лингвистических шаблонов, обеспечивает более релевантные и осмысленные результаты по сравнению с классическим полнотекстовым поиском.

**Входные данные:** ПДД РФ, 2. Общие обязанности водителей [29]; запрос – «средство»

**Выходные данные:**

После извлечения именованных сущностей, формирования словаря и выполнения поиска по словосочетанию «средство» для предложенного метода и классического полнотекстового поиска представлен в таблице 4.

Таблица 4 – Результат поиска для текстового документа «ПДД РФ, 2. Общие обязанности водителей».

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Метод | Термин | Синоним | Значение | Кол-во текстовых фрагментов |
| Размеченная коллекция документов | транспортный средство | автомобиль | грузовой автомобиль, грузовой автомобиль автобус, автобус | 35 |
| Классический полнотекстовый поиск | транспортное средство |  |  | 9 |

**Обоснование результатов:**

Поиск по размеченной коллекции документов позволяет учитывать не только термины, но и его синонимы и значения, а также реализованный поиск по косинусному сходству векторов позволяет находить вариации форм написания словосочетания.

### Выводы к экспериментам

Предложенный способ разметки текстовых документов с использованием подхода RAG, а так же комбинации извлечения именованных сущностей с помощью статистического и лингвистического подходов и последующим ручным формированием словарей позволяет сформировать легковесную структуру для конкретных предметных областей с возможностью релевантного поиска информации в исходных текстовых документах.

# Заключение

В ходе выполнения выпускной работы были получены следующие результаты:

1. Был проведён обзор подходов и методов разметки текстовых документов.
2. Был проведён анализ предметной области «Разметка текстовых документов».
3. Спроектирован и реализован прототип системы.
4. Проведено тестирование и экспериментальное исследование.

Таким образом, были решены все задачи, выделенные для достижения цели, поставленной в выпускной работе.

# Список литературы

1. Хабр. Часть 1. Обзор подходов RAG // Habr [Электронный ресурс]. – 2024. – URL: <https://habr.com/ru/articles/893650/> (дата обращения: 10.05.2025).
2. Хабр. Что такое Retrieval-Augmented Generation (RAG) в языковых моделях и как оно работает? // Habr [Электронный ресурс]. – 2024. – URL: <https://habr.com/ru/articles/841428/> (дата обращения: 10.05.2025)
3. Microsoft. RAG и генерированный ИИ - Azure AI Search // Microsoft Learn [Электронный ресурс]. – 2024. – URL: <https://learn.microsoft.com/ru-ru/azure/search/retrieval-augmented-generation-overview> (дата обращения: 10.05.2025).
4. Zerocoder. Что такое Retrieval Augmented Generation (RAG) в Prompt Engineering? // Zerocoder [Электронный ресурс]. – 2024. – URL: <https://ya.zerocoder.ru/pgt-chto-takoe-retrieval-augmented-generation-rag-v-prompt-engineering/> (дата обращения: 10.05.2025).
5. Ultralytics. Поколение, дополненное извлечением (Retrieval Augmented Generation, RAG) // Ultralytics [Электронный ресурс]. – 2025. – URL: <https://www.ultralytics.com/ru/glossary/retrieval-augmented-generation-rag> (дата обращения: 10.05.2025).
6. «RAG (Retrieval Augmented Generation) - простое и понятное объяснение» // Habr [Электронный ресурс]. – 2023. – URL: <https://habr.com/ru/articles/779526/> (дата обращения: 10.05.2025).
7. Недильченко, О. С. Этапы и методы автоматического извлечения ключевых слов / О. С. Недильченко. – Текст : непосредственный // Молодой ученый. – 2017. - Nº 22 (156). - C. 60-62. - URL: https://moluch.ru/archive/156/44044/ (дата обращения: 10.06.2024).
8. Основы Natural Language Processing для текста // Habr [Электронный ресурс]. – 2018. – URL: <https://habr.com/ru/companies/Voximplant/articles/446738/> (дата обращения: 10.05.2025).
9. Обработка естественного языка [Электронный ресурс]: NEERC Wiki. – URL: [https://neerc.ifmo.ru/wiki/index.php?title=Обработка\_естественного\_языка](https://neerc.ifmo.ru/wiki/index.php?title=%D0%9E%D0%B1%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D0%B5%D1%81%D1%82%D0%B5%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D0%BE%D0%B3%D0%BE_%D1%8F%D0%B7%D1%8B%D0%BA%D0%B0) (дата обращения: 10.05.2025).
10. Метрика TF-IDF (Term frequency–inverse document frequency) / Loginom Company : официальный сайт. – 2024 – URL: https://wiki.loginom.ru/articles/tf-idf.html (дата обращения: 03.06.2024).
11. Горлевич, Даниил. Алгоритмы для выделения ключевых слов: Rake, YAKE!, TextRank / NTA – 2022 – https://newtechaudit.ru/algoritmy-dlya-vydeleniya-klyuchevyh-slov-rake-yake-textrank/ (дата обращения: 03.06.2024)
12. Turney P. «Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classifi cation of Reviews». 2002. pp. 417-424.
13. Я.Ю. Дементьева, Е.П. Бручес, Т.В. Батура Извлечение терминов из текстов научных статей // Программные продукты и системы. 2022. №4. URL: https://cyberleninka.ru/article/n/izvlechenie-terminov-iz-tekstov-nauchnyh-statey (дата обращения: 09.06.2024).
14. Текстовые документы и технологии их создания // ЯКласс: образовательная платформа [Электронный ресурс]. – URL: <https://www.yaklass.by/p/informatika/6-klass/obrabotka-tekstovoi-informatcii-2714/tekstovye-dokumenty-i-tekhnologii-ikh-sozdaniia-2790/re-a569f2d9-840e-4d2c-8901-5816062cf490> (дата обращения: 26.05.2025).
15. Текстовые документы. Текстовые редакторы // Oblakoz.ru: конспекты уроков [Электронный ресурс]. – URL: https://oblakoz.ru/conspect/489136/Tekstovye%20dokumenty.%20Tekstovye%20redaktory (дата обращения: 26.05.2025).
16. Михайлов Д. В. Мера TF-IDF, сила связи слов и ключевые сочетания для тематических текстов // Биомедицинская инженерия и радиоэлектроника: материалы конференции. – 2020. – С. 4–19. – URL: <http://www.machinelearning.ru/wiki/images/7/79/Biomed_engin_2020_mdv_pres.pdf> (дата обращения: 26.05.2025)
17. Извлечение признаков из текстовых данных с использованием TF-IDF // Блог компании OTUS [Электронный ресурс]. –URL: <https://habr.com/ru/companies/otus/articles/755772/> (дата обращения: 26.05.2025)
18. Сидельникова М. П. Разработка системы для извлечения терминов из коллекции текстовых документов : магистерская диссертация / Дальневосточный федеральный университет. – Владивосток, 2024.
19. Распознавание именованных сущностей (NER) // Sysblok.ru: глоссарий по искусственному интеллекту [Электронный ресурс]. – URL: <https://sysblok.ru/glossary/named-entity-recognition-ner/> (дата обращения: 26.05.2025).
20. Bootstrap. Официальный сайт // getbootstrap.su [Электронный ресурс]. – 2025. – URL: https://getbootstrap.su (дата обращения: 01.06.2025).
21. Python. Официальный сайт // Python.org [Электронный ресурс]. – 2025. – URL: https://www.python.org/ (дата обращения: 01.06.2025).
22. FastAPI. Официальная документация // FastAPI.tiangolo.com [Электронный ресурс]. – 2025. – URL: https://fastapi.tiangolo.com/ (дата обращения: 01.06.2025).
23. FastAPI. Официальная документация // FastAPI.tiangolo.com [Электронный ресурс]. – 2025. – URL: https://fastapi.tiangolo.com/ (дата обращения: 01.06.2025).
24. Scikit-learn. Библиотека машинного обучения для Python // scikit-learn.org [Электронный ресурс]. – 2025. – URL: https://scikit-learn.org/stable/ (дата обращения: 01.06.2025).
25. Jinja. Официальная документация по шаблонизатору // jinja.palletsprojects.com [Электронный ресурс]. – 2025. – URL: https://jinja.palletsprojects.com/en/stable/ (дата обращения: 01.06.2025).
26. Docker Documentation / Docker Inc. : официальный сайт. – 2013-2023 – URL: https://docs.docker.com/ (дата обращения: 01.06.2025).
27. Алексеев А. А. Разработка вопросно-ответной системы для антропоморфного социального агента [Электронный ресурс] : выпускная квалификационная работа. – Казань, 2018. – URL: <https://kpfu.ru/student_diplom/10.160.178.20_264066_Alekseev.pdf> (дата обращения: 13.06.2025)
28. Красная Шапочка – сказка Шарля Перро [Электронный ресурс]. – Режим доступа: https://nukadeti.ru/skazki/sharl-perro-krasnaya-shapochka (дата обращения: 14.06.2025).
29. Конституция Российской Федерации. Постановление Правительства РФ от 23.10.1993 N 1090 (ред. от 27.03.2025) "О Правилах дорожного движения" (вместе с "Основными положениями по допуску транспортных средств к эксплуатации и обязанности должностных лиц по обеспечению безопасности дорожного движения") / ПДД РФ, 2. Общие обязанности водителей // КонсультантПлюс : справочно-правовая система. URL: https://www.consultant.ru/document/cons\_doc\_LAW\_2709/a4b879c29ebc2ff9a56a0595499b6eb2dce7980e/ (дата обращения: 15.06.2025).

# Приложение А

Таблица 1 – Описание тестовых ситуаций

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № тестовой ситуации | Класс теста | Описание тестовой ситуации | Входные данные | Ожидаемый результат | Статус теста |
| 1 | Извлечение кандидатов в именованные сущности | Одно слово | “тест” | [“тест”] | Test passed |
| 2 | Слово через дефис | “тест-тест” | [“тест-тест”] | Test passed |
| 3 | Числа. Не целые числа не учитываются | “123 5.0 2024 год” | [“123”, “2024”] | Test passed |
| 4 | Игнорирование пунктуации | “Привет, мир!” | [“привет”, “мир”] | Test passed |
| 5 | Максимальное количество n-грамм=2 | “один два три” | [“один”, “два”, “три”, “два три”] | Test passed |
| 6 | Один символ | “а” | [] | Test passed |
| 7 | Пустая строка | “” | [] | Test passed |
| 8 | Извлечение именованных сущностей | Пустой текст | “” | Выброшено исключение | Test passed |
| 9 | Одно слово | “hello” | [(“hello”, 1.0)] | Test passed |
| 10 | Повторяющиеся слова | “hello hello world” | [(“hello”, 0.9), (“world”, 0.45)] | Test passed |
| 11 | Единичные символы | “a b c d” | Выброшено исключение | Test passed |
| 12 | Извлечение топ n, n=2 | “he she it” | len(result) == 2 | Test passed |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № тестовой ситуации | Класс теста | Описание тестовой ситуации | Входные данные | Ожидаемый результат | Статус теста |
| 13 | Извлечение именованных сущностей с meta-информацией | Пустой текст | “” | {"phrases": [], "total\_phrases": 0, "unique\_phrase\_types":0} | Test passed |
| 14 | Одно словосочетание | “быстрый кот” | {"phrases": [{"text": "быстрый кот",  "type": "адъективное",  "pattern\_description": "Прил + Сущ",  "tfidf\_score": 0.8,  "length": 2}],  "total\_phrases": 1,  "unique\_phrase\_types": 1} | Test passed |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № тестовой ситуации | Класс теста | Описание тестовой ситуации | Входные данные | Ожидаемый результат | Статус теста |
| 15 | Извлечение именованных сущностей с meta-информацией | Несколько словосочетаний | “быстрый кот, кот, рыжий кот” | {"phrases": [{  "text": "быстрый кот",  "type": "адъективное",  "pattern\_description": "Прил + Сущ",  "tfidf\_score": 0.8,  "length": 2  },  {  "text": "кот",  "type": "однословное",  "pattern\_description": "Сущ",  "tfidf\_score": 0.7,  "length": 1  },  {  "text": "рыжий кот",  "type": "адъективное",  "pattern\_description": "Прил + Сущ",  "tfidf\_score": 0.5,  "length": 2}],  "total\_phrases": 3,  "unique\_phrase\_types": 2} | Test passed |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № тестовой ситуации | Класс теста | Описание тестовой ситуации | Входные данные | Ожидаемый результат | Статус теста |
| 16 | Поиск терминов, синонимов, значений по словосочетанию | Пустой список фраз для поиска | Искомое словосочетание=“query”;  Термины=[] | [] | Test passed |
| 17 | Нет схожих терминов | Искомое словосочетание=“query”;  Термины=[“фраза 1”, “фраза 2”, “фраза 3”] | len(result)==0 | Test passed |
| 18 | Есть схожие термины | Искомое словосочетание=“фраза”;  Термины=[“фраза 1”, “фраза 2”, “фраза 3”] | len(result)==3 | Test passed |
| 19 | Поиск фрагментов текста, где встречался найденный термин | Фрагмент найден | Предложения=[“Красная шапочка шла по дороге”];  Искомые словосочетания=[“шапочка”] | len(result)==1 | Test passed |
| 20 | Фрагмент не найден | Предложения=[“Красная шапочка шла по дороге”];  Искомые словосочетания=[“тест”] | len(result)==0 | Test passed |