**Санкт-Петербургский государственный электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**(СПбГЭТУ «ЛЭТИ»)**

|  |  |  |
| --- | --- | --- |
| **Направление** | 09.04.02 Информационные системы и технологии | |
| **Программа** | Управление IT проектами и продуктами | |
| **Факультет** | КТИ | |
| **Кафедра** | АПУ | |
| *К защите допустить* |  | |
| Зав. кафедрой |  | Шестопалов М.Ю. |

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

МАГИСТРА

Тема: Разработка алгоритма семантической декомпозиции текста на ключевые элементы для решения задачи категоризации

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студентка |  |  | |  | Терещенко В.Н. |
|  |  | *подпись* | |  |  |
| Руководитель | к.т.н., доцент |  | |  | Кораблев Ю.А. |
|  |  | *подпись* | |  |  |
| Консультанты | к.т.н. |  | |  | Лосева Д.М. |
|  |  | *подпись* | |  |  |
|  | к.т.н. | *По доп разделу* | |  | Заславский М.М. |
|  |  | *подпись* | |  |  |
|  |  | |

Санкт-Петербург

2022**ЗАДАНИЕ**

**на выпускную квалификационную работу**

|  |  |
| --- | --- |
|  | Утверждаю |
|  | Зав. кафедрой АПУ |
|  | \_\_\_\_\_\_\_\_\_\_\_\_ Шестопалов М.Ю. |
|  | «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_2022 г. |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студентка | Терещенко В.Н. | | | | Группа 6371 |
| Тема работы: Разработка алгоритма семантической декомпозиции текста на ключевые элементы для решения задачи категоризации. | | | | | |
| Место выполнения ВКР: Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И. Ульянова (Ленина), кафедра АПУ | | | | | |
| Исходные данные (технические требования): OC Windows 10 64-bit, Python 3.8.0 и выше 64-bit. | | | | | |
| Содержание ВКР:  Введение, Обзор предметной области, Формулировка требований к решению, Описание процесса разработки, Исследование, Безопасность жизнедеятельности, Заключение. | | | | | |
| Перечень отчетных материалов: пояснительная записка, иллюстративный материал. | | | | | |
| Дополнительные разделы: Коммерциализация результатов исследований в рамках НИР. | | | | | |
|  | | | | | |
| Дата выдачи задания | | | Дата представления ВКР к защите | | |
| «\_01\_» \_\_февраля\_\_\_\_\_2022 г. | | | «\_01\_» \_\_июня\_\_\_\_\_2022 г. | | |
| Студентка | |  | | Терещенко В.Н. | |
| Руководитель к.т.н., доцент | |  | | Кораблев Ю.А. | |

**календарный план выполнения**

**выпускной квалификационной работы**

|  |  |
| --- | --- |
|  | Утверждаю |
|  | Зав. кафедрой АПУ |
|  | \_\_\_\_\_\_\_\_\_\_\_\_ Шестопалов М.Ю. |
|  | «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_2022 г. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студентка | Терещенко В.Н. |  | Группа | 6371 |
| Тема работы: Разработка алгоритма семантической декомпозиции текста на ключевые элементы для решения задачи категоризации. | | | | |

|  |  |  |
| --- | --- | --- |
| № п/п | Наименование работ | Срок выполнения |
| 1 | Обзор литературы по теме работы | 20.01 – 21.02 |
| 2 | Разработка алгоритма | 22.02 – 05.04 |
| 3 | Проведение исследования | 06.04 – 09.04 |
| 4 | Внедрение алгоритма в СППР и тестирование | 06.05 – 15.05 |
| 5 | Оформление пояснительной записки | 10.04 – 22.05 |
| 6 | Оформление иллюстративного материала | 23.05 – 26.05 |
| 7 | Предзащита | 26.05 |

|  |  |  |
| --- | --- | --- |
| Студентка |  | Терещенко В.Н. |
| Руководитель к.т.н., доцент |  | Кораблев Ю.А. |

**РЕФЕРАТ**

Пояснительная записка 59 стр., 8 рис., 3 табл., 32 ист., 2 прил.

Ключевые слова: ключевое слово, ключевое словосочетание, ключевая фраза, ключевое предложение, семантическая декомпозиция, классификация, категоризация.

Цель работы – исследование современных методов классификации, создание инструмента, автоматически генерирующего оглавление и указатели терминов для массовых открытых онлайн-курсов платформы Stepik.

Объектом разработки является проектируемое веб-приложение.

В работе описано исследование подходов к решению задачи выделения ключевых слов, описан процесс проектирования веб-приложения для создания оглавления и списка терминов. Приведено практическое сравнение статистического и графового подходов в разработанном веб-приложении. При проектировании приложения были использованы язык программирования Python, веб-фреймворк Flask, СУБД PostgreSQL и REST-интерфейс Stepik API. Полученное приложение исследовано: определены время работы алгоритмов выделения ключевых слов Frequency, TopicRank и TextRank, проверены точность алгоритмов методом сравнения с экспертной оценкой.

**ABSTRACT**

The paper contains a study of approaches to solving the problem of highlighting keywords, describes the design process of a web application to create a table of contents and a list of terms. A practical comparison of statistical and graph approaches in the developed web application is given. When designing the application, the programming language Python, the Flask web framework, the PostgreSQL DBMS, and the Stepik API REST interface were used. The resulting application was investigated: the operating time of the Keyword, Frequency, TopicRank, and TextRank keyword extraction algorithms was determined, the accuracy of the algorithms was verified by comparison with an expert assessment.

**содержание**

[**определения, обозначения и сокращения** 8](#_Toc103106614)

[**введение** 9](#_Toc103106615)

[**1.** **Обзор предметной области** 11](#_Toc103106616)

[**1.1.** **Классификация и категоризация текстов на естественном языке** 11](#_Toc103106617)

[**1.1.1.** **Этапы процесса категоризации текстов** 14](#_Toc103106618)

[**1.1.2.** **Алгоритм декомпозиции и алгоритм выделения ключевых слов. Ключевые слова** 18](#_Toc103106619)

[**1.1.3.** **Ключевые элементы. Понятие расширения ключевого элемента** 19](#_Toc103106620)

[**1.2.** **Семантическая декомпозиция как один из инструментов семантического анализа текстов** 19](#_Toc103106621)

[**1.3.** **Сравнительный анализ существующих алгоритмов семантической декомпозиции и выбор базовых алгоритмов для реализации** 21](#_Toc103106622)

[**1.4.** **Выводы** 42](#_Toc103106623)

[**2.** **ПОСТАНОВКА ЗАДАЧИ** 43](#_Toc103106624)

[**3.** **Описание Теоретических основ разрабатываемого алгоритма** 45](#_Toc103106625)

[**3.1.** **Графическое описание алгоритма семантической декомпозиции** 45](#_Toc103106626)

[**3.2.** **Лингвистические основы алгоритма семантической декомпозиции** 45](#_Toc103106627)

[**3.2.1.** **Основные процедуры алгоритма** 45](#_Toc103106628)

[**3.2.2.** **Вспомогательные процедуры алгоритма** 45](#_Toc103106629)

[**3.3.** **Математические основы алгоритма семантической декомпозиции** 46](#_Toc103106630)

[**3.4.** **Сценарий применения алгоритма для решения задачи категоризации** 46](#_Toc103106631)

[**3.5.** **Выводы** 47](#_Toc103106632)

[**4.** **ПРОГРАММНАЯ РЕАЛИЗАЦИЯ АЛГОРИТМОВ** 48](#_Toc103106633)

[**4.1.** **Технические подробности программной реализации алгоритма** 48](#_Toc103106634)

[**4.1.1.** **Тестирование алгоритма** 48](#_Toc103106635)

[**4.1.2.** **Сценарий использования** 48](#_Toc103106636)

[**4.1.3.** **Применение алгоритма для решения задачи категоризации** 48](#_Toc103106637)

[**4.2.** **Алгоритм семантической декомпозиции на предложения** 48](#_Toc103106638)

[**4.2.1.** **Технические подробности программной реализации алгоритма** 48](#_Toc103106639)

[**4.2.2.** **Тестирование алгоритма** 49](#_Toc103106640)

[**4.2.3.** **Сценарий использования** 49](#_Toc103106641)

[**4.2.4.** **Применение алгоритма для решения задачи категоризации** 49](#_Toc103106642)

[**4.3.** **Выводы** 49](#_Toc103106643)

[**5.** **исследование** 51](#_Toc103106644)

[**5.1.** **Исследование времени работы** 51](#_Toc103106645)

[**5.1.1.** **Сценарий эксперимента** 51](#_Toc103106646)

[**5.1.2.** **Методика измерения** 51](#_Toc103106647)

[**5.1.3.** **Используемые программные средства** 51](#_Toc103106648)

[**5.1.4.** **Результаты эксперимента** 52](#_Toc103106649)

[**5.1.5.** **Сравнение и оценка результатов** 52](#_Toc103106650)

[**5.2.** **Исследование точности результатов** 52](#_Toc103106651)

[**5.2.1.** **Сценарий эксперимента** 53](#_Toc103106652)

[**5.2.2.** **Входные данные** 53](#_Toc103106653)

[**5.2.3.** **Результат работы алгоритмов** 53](#_Toc103106654)

[**5.2.4.** **Экспертная оценка** 54](#_Toc103106655)

[**5.2.5.** **Сравнение с экспертной оценкой** 54](#_Toc103106656)

[**5.3.** **Выводы** 55](#_Toc103106657)

[**6.** **Коммерциализация результатов исследований в рамках НИР** 56](#_Toc103106658)

[**заключение** 57](#_Toc103106659)

[**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ** 59](#_Toc103106660)

# **определения, обозначения и сокращения**

В настоящей пояснительной записке применяют следующие термины с соответствующими определениями:

Веб-сайт – совокупность взаимосвязанных веб-страниц, используемых одно доменное имя (адрес).

Веб-страница – документ, обычно в формате html, содержащий какой-либо контент (текст, фото, видео и другое).

Документ – коллекция текста (твит, статья и др.).

Классификатор – алгоритм, или метод, посредством которого проводится классификация.

Классификация – процедура, в ходе которой объекты (предметы, явления) распределяются по группам, или классам, по каким-либо признакам, удобным для их исследования.

Корпус – набор документов схожей тематики.

Токен – это последовательность символов в документе, имеющая значение для анализа. Обычно это отдельные слова, но не всегда.

Документ – это коллекция текста. (https://nuancesprog.ru/p/5870/)

МООК – массовые открытые онлайн курс (Massive Open Online Course, MOOC).

ОЗУ – оперативное запоминающее устройство.

Реестр – форма систематизации, учета.

СУБД – система управления базами данных.

API – Application Programming Interface (программный интерфейс приложения).

ID – Identifier (идентификатор).

NLP – Natural Language Processing (обработка естественного языка).

REST – Representational State Transfer (передача состояния представления).

SQL – Structured Query Language (язык структурированных запросов).

# **введение**

В настоящее время системы обработки естественного языка используются повсеместно, начиная от переводчика и поисковой системы в браузере, заканчивая умными чат-ботами и голосовыми помощниками. Такие системы внедряются в совершенно различные структуры и упрощают их использование.

Недавно для компании ООО «Газпром переработка» была разработана система поддержки принятия решений для автоматизации работы службы технической поддержки. Данная СППР автоматически обрабатывает запросы, поступающие на 1-ую линию технической поддержки. Обработка главным образом заключается в категоризации запросов и направлении их по различным отделам компании.

Данная тема магистерской диссертации была выбрана с целью исследования влияния различных вариантов разделения текста запроса на результат работы СППР, а именно на результат классификации и, соответственно, категоризации текстовых запросов.

В большинстве случаев наличие среди ключевых слов устойчивых словосочетаний ведет положительно влияет на вероятность правильного категорирования специализированных текстов. **Цель** данной работы – проверить эту гипотезу для текстов обращений, поступающих в техническую поддержку компании ООО Газпром переработка, и предложить вариант доработки проекта с учетов результата исследования.

Для достижения цели были поставлены следующие **задачи**:

* сравнить и проанализировать существующие подходы семантической декомпозиции текста;
* разработать алгоритм и протестировать его (это тоже этап разработки);
* исследовать качество разработанного алгоритма;
* провести внедрение алгоритма в проект (в случае успешного исследования).

**Объектом** исследования являются тексты запросов, поступившие в службу поддержки.

**Предметом** исследования является процесс семантической декомпозиции текста запроса на ключевые элементы.

**Практическая значимость** решения заключается в использовании разработанного алгоритма для улучшения результатов категоризации обращений и, соответственно, для оптимизации работы СППР для службы технической поддержки компании Газпром переработка.

1. **Обзор предметной области**

Разрабатываемый алгоритм является частью системы, предназначенной для категоризации запросов.

* 1. **Классификация и категоризация текстов на естественном языке**

Классификаций или категоризацией называют процесс отнесения входных текстовых документов к одной из групп данных (классу или категории).

Эти две интеллектуальные задачи следует отличать друг от друга. Термин *класс* можно определить, как совокупность предметов или понятий (образов), выделенных и сгруппированных по определенным признакам или правилам. Под *классификацией* понимается отнесение некоторого документа к классу, выполняемое по формальным правилам, исходя из совокупности признаков. Границы классов определены достаточно точно: документ относится к данному классу, если известно, что он обладает необходимым числом признаков, характерных для этого класса.

*Категория* же определяет лишь некоторые общие свойства образов и связи между ними. Задача *категоризации*, т. е. определения отношения данного документа к некоторой категории, гораздо менее определена, чем задача отношения к классу. Границы различных категорий являются нечеткими, расплывчатыми, и обычно сама категория понимается не через формальное определение, а только в сравнении с другими категориями.

Таким образом, задачей систем-классификаторов является установление принадлежности образа к одному из формально определенных классов. Примерами такой задачи является задача классификации растений в ботанике, классификация химических веществ по их свойствам и типам возможных реакций, в которые они вступают, и другие. Формальные признаки могут быть определены посредством правил типа “если-то”, а системы, оперирующие с такими правилами, получили название *экспертных систем*. Традиционной областью применения классификаторов на нейронных сетях является экспериментальная физика высоких энергий, где одной из актуальных задач выступает выделение среди множества зарегистрированных в эксперименте событий с элементарными частицами событий, представляющих интерес для данного эксперимента.

Таким образом, проблема категоризации является более сложной в сравнении с проблемой классификации. В категоризации помимо отнесения документа к какой-либо группе, требуется определить сами эти группы, т. е. сформировать категории.

В случае обучения с учителем формирование категорий происходит методом проб и ошибок на основе примеров с известными ответами, предоставляемыми экспертом. Эксперта обычно называют “супервизором” или учителем. Учитель управляет обучением при помощи изменения параметров связей и, реже, самой топологии сети.

Задачей системы-категоризатора является формирование обобщающих признаков в совокупности примеров. При увеличении числа примеров несущественные, случайные признаки сглаживаются, а часто встречающиеся – усиливаются, при этом происходит постепенное уточнение границ категорий. Хорошо обученная нейросетевая система способна извлекать признаки из новых примеров, ранее неизвестных системе, и принимать на их основе приемлемые решения.

Важно отметить различие в характере неявных “знаний”, запомненных искусственной нейронной сетью, и явных, формальных “знаний”, заложенных в экспертных системах. Некоторые сходства и различия представлены в следующей таблице.

|  |  |  |
| --- | --- | --- |
| Критерий | **Экспертные системы (ЭС)** | **Нейросетевые системы (НС)** |
| Источник знаний | Формализованный опыт эксперта, выраженный в виде логических утверждений (правил и фактов), безусловно принимаемых системой | Совокупный опыт эксперта-учителя, отбирающего примеры для обучения, и индивидуальный опыт обучающейся на этих примерах нейронной сети |
| Характер знаний | Формально-логическое “левополушарное” знание в виде правил | Ассоциативное “правополушарное” знание в виде связей между нейронами сети |
| Развитие знаний | В форме расширения совокупности правил и фактов (базы знаний) | В форме дообучения на дополнительной последовательности примеров, с уточнением границ категорий и формированием новых категорий |
| Роль эксперта | Задает на основе правил полный объем знаний экспертной системы | Отбирает характерные примеры, не формулируя специально обоснование своего выбора |
| Роль искусственной системы | Поиск цепочки фактов и правил для доказательства суждения | Формирование индивидуального опыта в форме категорий, получаемых на основе примеров, и категоризация образов |

Различия в характере экспертных и нейросетевых систем обуславливают и различия в их сферах применения. Экспертные системы применяются в узких предметных областях с хорошо структурированными знаниями, например в классификации неисправностей конкретного типа оборудования, фармакологии, анализе химсостава проб и т. д. Нейронные сети применяются кроме перечисленных областей и в задачах с плохо структурированной информацией, например при распознавании образов, рукописного текста, анализе речи и т. д. (<https://pandia.ru/text/79/574/8277.php#:~:text=Классификация%20и%20категоризация.%20В%20случае%2C,выполняет%20классификацию%20или%20категоризацию%20данных>)

* + 1. **Этапы процесса категоризации текстов**

Решение задач категоризации и классификации состоят из четырех последовательных этапов:

* предобработка и индексация документов;
* уменьшение размерности пространства признаков;
* построение и обучение классификатора с помощью методов машинного обучения;
* оценка качества классификации.

На рисунке 1 представлена общая схема процесса категоризации.



Рисунок 1 – Этапы процесса категоризации текстов (скрин из статьи, можно перерисовать)

Рассмотрим этапы, предшествующие работе классификатора, так как алгоритм семантической декомпозиции будет встроен именно в эту часть системы.

**Предварительная обработка** текста включает нормализацию (перевод всех букв к одному регистру (чаще к нижнему), удаление знаков пунктуации, цифр, пробелов), токенизацию (разбиение текста на более мелкие части – токены), удаление стоп-слов (семантически нейтральных слов, таких как союзы, предлоги, артикли и пр.), морфологический анализ (разметка по частям речи; опционально), лемматизация и стемминг. Это позволяет значительно сократить размерность пространства. В результате в качестве признаков документа выступают все значимые слова, встречающиеся в документе.

Немного о процедурах предварительной обработки.

* **Нормализация и токенизация**. В самом начале обработки текст приводится к единообразному виду (единый регистр слов, отсутствие знаков пунктуации, расшифрованные сокращения, словесное написание чисел и т.д.), что необходимо для применения унифицированных методов обработки текста, и разделяется на более мелкие единицы (на символы, слова, фразы или предложения) для того, чтобы иметь возможность провести последующие процедуры.
* **Лемматизация и стемминг**. При построении классификатора текстов нет смысла различать формы одного и того же слова. Это приводит к неоправданному разрастанию словаря, дроблению статистики, увеличению ресурсоёмкости и снижению качества модели.
  + Лемматизация – приведение каждого слова в документе к его нормальной форме (в русском языке: для существительных – именительный падеж, единственное число; для прилагательных – именительный падеж, единственное число, мужской род; для глаголов, причастий, деепричастий – глагол в инфинитиве). Существуют специальные программы – лемматизаторы, обычно основанные на явном хранении грамматического словаря со всеми формами слов. Недостатком лемматизации является трудоёмкость составления словарей, и, как следствие, их неполнота, особенно по части специальной терминологии и неологизмов, которые как раз и представляют наибольший интерес для тематического моделирования.
  + Стемминг – отбрасывание изменяемых частей слов (главным образом, окончаний). Она не требует хранения словаря всех слов и основана на правилах морфологии языка. Недостатком стемминга является большее число ошибок. Стемминг лучше подходит для английского языка, но хуже для русского.
* **Отбрасывание стоп-слов**. Слова, встречающиеся во многих текстах различной тематики, бесполезны при классификации, и могут быть отброшены. К ним относятся предлоги, союзы, числительные, местоимения, некоторые глаголы, прилагательные и наречия. Число таких слов обычно варьируется в пределах нескольких сотен. Их отбрасывание почти не влияет на длину словаря, но может приводить к заметному сокращению длины некоторых текстов.
* **Отбрасывание редких слов**. Слова, встречающиеся в длинном тексте слишком редко, также можно отбрасывать, полагая, что данное слово не имеет принципиального значения в данном тексте.
* **Выделение ключевых фраз**. При обработке коллекций научных, юридических или других специальных текстов вместо отдельных слов выделяют ключевые фразы – словосочетания, являющиеся устойчивыми оборотами или терминами в данной предметной области. Это отдельная и довольно сложная задача, которая может решаться методами машинного обучения с привлечением экспертов для формирования обучающих выборок и контроля качества автоматического выделения терминов [1].

(http://www.machinelearning.ru/wiki/images/0/02/Romanenko2012bach.pdf)

**Индексация** документов представляет собой построение некоторой числовой модели текста, которая переводит текст в удобное для дальнейшей обработки представление.

Распространенные модели индексации:

* модель «мешка слов» (bag-of-words) позволяет представить документ в виде многомерного вектора слов и их весов в документе [2]. Другими словами, каждый документ – это вектор в многомерном пространстве, координаты которого соответствуют номерам слов, а значения координат – значениям весов;
* модель индексации Word2vec [3] представляет каждое слово в виде вектора, который содержит информацию о контекстных (сопутствующих) словах;
* модель индексации, основанная на учете n-грамм [2], то есть последовательностей из соседних токенов.

(https://cyberleninka.ru/article/n/metody-avtomaticheskoy-klassifikatsii-tekstov/viewer)

Этап **извлечения признаков** из текстовых документов необходим для того, чтобы сопоставить каждому документу некий набор характеристик, которые описывают этот документ [1]. (<https://cyberleninka.ru/article/n/metody-izvlecheniya-priznakov-iz-tekstovyh-dokumentov/viewer>)

Извлечение признаков соответствует выбору элементов признаков и вычислению весов признаков. Основная идея состоит в том, чтобы независимо ранжировать токены в соответствии с определенным индексом оценки, выбирать токены с наивысшими баллами и отфильтровывать оставшиеся. Обычно используемые оценки включают частоту встречаемости токенов, взаимную информацию и др. (https://russianblogs.com/article/52941084692/) Часто для этого применяются алгоритмы выделения ключевых слов, выделяющие наиболее значимые слова в тексте.

Далее на полученных признаках модель обучается и после успешного прохождения тестирования система категоризации готова к работе.

* + 1. **Алгоритм декомпозиции и алгоритм выделения ключевых слов. Ключевые слова**

Исходя из вышеизложенной информации, становится ясно, что разрабатываемый алгоритм семантической декомпозиции на ключевые элементы является элементом этапа предобработки.

На вход алгоритму подается «сырой» текст, а на выходе получается список слов (ключевых элементов) из исходного текста. Данный список используется в качестве исходных данных в следующем алгоритме – алгоритме выделения ключевых слов. Здесь из ключевых элементов выбираются значимые – ключевые слова.

В общем представлении ключевыми называются важные слова или фразы, дающие высокоуровневое описание содержания текстового документа, позволяющие выявить его тематику, имеющие компактное представление в памяти. Выделенный из текста список ключевых слов (КС) может выступать в качестве метаинформации, представляя текстовый документ при решении задач информационного поиска, классификации, кластеризации, аннотирования и реферирования.

Ключевые словосочетания (фразы) представляют собой сочетание двух или более слов, которые как могут следовать друг за другом в тексте, так и быть разделенными другими языковыми единицами. Не все входящие в состав ключевых фраз слова при отдельном рассмотрении являются ключевыми. Но также вполне очевидно, что выделением отдельных ключевых слов затруднительно выразить основной смысл содержимого. Поэтому на практике востребовано выделение именно ключевых фраз. Основное отличие – длина списка на выходе алгоритма. Документу обычно соответствуют единицы ключевых фраз, длина же списка терминов, основой которых также являются КС колеблется от десятков до сотен. Составление перечня ключевых словосочетаний является одной из трудностей в рассматриваемой предметной области. (https://cyberleninka.ru/article/n/metody-i-algoritmy-izvlecheniya-klyuchevyh-slov/viewer)

Таким образом, из данного подраздела следует, что из списка ключевых элементов во время работы алгоритма выделения ключевых слов, следующего после алгоритма декомпозиции, будут сформированы ключевые слова (термины), словосочетания или фразы.

Далее подробнее о ключевых элементах.

* + 1. **Ключевые элементы. Понятие расширения ключевого элемента**

Как было сказано выше, из-за трудности выражения основного смысла текста отдельными ключевыми словами, стала необходимым возможность выделения ключевых словосочетаний или фраз. Для этого в данной работе предусмотрено расширение понятие ключевого элемента от слова (униграммы) до словосочетания (фразы, n-грамма).

Таким образом, под ключевыми элементами в работе понимаются как отдельные слова, так и словосочетания или n-граммы. Также в данной работе будет употребляться понятие токена (опр. См. на стр. 8) в качестве синонима понятию ключевого элемента.

* 1. **Семантическая декомпозиция как один из инструментов семантического анализа текстов**

Итак, понятие ключевого элемента обозначено. Следующим шагом работы будет определение понятия семантической декомпозиции, начиная с более общего понятия – семантического анализа.

Семантический анализ текста оценивает количество слов или фраз, которые определяют смысл текста (то есть его семантическое ядро, и статистические показатели). <https://cropas.by/seo-slovar/semanticheskij-analiz/>

Так, всю совокупность представленных на сегодняшний день методов анализа текста можно разделить на две группы:

* лингвистический анализ − основан на извлечении смысла текста по его семантической структуре;
  + статистический анализ − основан на извлечении смысла текста по частотному распределению слов в тексте.

Деление на группы условное, так как в реальных задачах и при решении проблем всегда используется сочетание методов для достижения определенного результата. (там есть еще подробное описание -х самых востребованных семантических методов с результатами <http://repo.ssau.ru/bitstream/Informacionnye-tehnologii-i-nanotehnologii/Razrabotka-avtomatizirovannoi-sistemy-semanticheskogo-analiza-tekstovoi-informacii-64154/1/paper%20324_1800-1804.pdf>)

Проблема аналитического анализа семантики и извлечения знаний из текстовых документов является достаточно сложной и требует разработки и применения специализированных интеллектуальных информационных систем (ИИС) [2, 3]. На практике путём введения ряда допущений можно разделить проблему на несколько групп относительно независимых типовых задач.

К типовым задачам исследования семантики одиночных текстовых документов следует отнести:

а) выделение в тексте ключевых слов;

б) выделение в тексте цепочки ключевых слов;

в) выявление контекстных слов (контекста);

г) аннотирование текста.

Для случая обработки набора тестового документа приоритетными являются следующие задачи:

а) классификация текстов по совокупности признаков;

б) каталогизация текстов;

в) поиск заданных информационных фрагментов в потоке ТД;

г) поиск (подбор) текстов, обладающих заданными свойствами. (<https://fundamental-research.ru/ru/article/view?id=41321>)

ТУТ МОЖНО ПРОКОММЕНТИРОВАТЬ ПО ПУНКТАМ, ЧТО ПОДХОДИТ НАМ

Под понятием семантической декомпозиции, в свою очередь, понимают алгоритм, разбивающий значения предложений, фраз или понятий на менее сложные понятия. (<https://translated.turbopages.org/proxy_u/en-ru.ru.65356b33-623e167d-0feb0112-74722d776562/https/en.wikipedia.org/wiki/Semantic_decomposition_(natural_language_processing)>)

В лингвистических экспертизах присутствует огромное множество вариантов компонент, на которые может разбиваться текст. (<https://www.tipse.ru/jour/article/viewFile/220/220>) Однако в рамках данной работы будет вестись исследование семантической декомпозиции на слова и фразы (словосочетания). Поэтому здесь будут рассмотрены три вида алгоритмов, исходя из типа единиц декомпозиции текста:

* алгоритмы семантической декомпозиции текста на слова;
* алгоритмы семантической декомпозиции текста на словосочетания;
* алгоритмы семантической декомпозиции текста на слова и словосочетания.
  1. **Сравнительный анализ существующих алгоритмов семантической декомпозиции и выбор базовых алгоритмов для реализации**

По своей сути алгоритм семантической декомпозиции приближен к алгоритмам токенизации, поэтому для анализа уже существующих решений были подобраны следующие инструменты:

1. **Модуль re**.

Используется для применения регулярных выражений для различных задач поиска. Регулярные выражения – это последовательность символов, которая может использоваться для определения искомого шаблона для поиска текста, изменения или разделения строк. Это самый простой способ токенизации, без каких-либо сопутствующих процедур (лемматизации, распознавания частей речи и так далее). (https://rukovodstvo.net/posts/id\_1304/)

1. **Библиотека Nltk**.

Это самая большая библиотека для NLP на python. Здесь помимо токенизации есть также лемматизация и стемминг, маркировка частей речи, подбор синонимов и антонимов и другое.

В модуле tokenize библиотеки nltk хранится 19 токенизаторов. Ниже приведены наиболее подходящие для поставленной задачи (<https://newtechaudit.ru/obzor-tokenizatorov-vhodyashhih-v-sostav-biblioteki-nltk/>):

* 1. **RegexpTokenizer**

Данный токенизатор разбивает исходный текст на подстроки, используя регулярное выражение, передаваемое ему в качестве параметра.

Библиотека NLTK содержит также несколько подклассов RegexpTokenizer, использующих заранее определенные регулярные выражения: WhitespaceTokenizer, BlanklineTokenizer, WordPunctTokenizer.

* + 1. **WhitespaceTokenizer**

При помощи данного токенизатора исходный текст разделяется на группы символов, ограниченных пробелами (обычно, это отдельные слова, либо слова со следующими за ними знаками препинания).

Вместо WhitespaceTokenizer можно использовать строковый метод split().

* + 1. **BlanklineTokenizer**

Токенизатор разбивает текст, рассматривая в качестве разделителя любую последовательность пустых строк (содержащих только символы пробела или табуляции). Для этого используется регулярное выражение ‘\s\*\n\s\*\n\s\*’.

* + 1. **WordPunctTokenizer**

Токенизатор преобразует текст в последовательность буквенных и неалфавитных символов с помощью регулярного выражения ‘\w+|[^\w\s]+’. Отличается от WhitespaceTokenizer тем, что отделяет знаки препинания и иные символы пунктуации от слова, за или перед которым они идут, и возвращает их в качестве отдельных токенов.

* 1. Следующая группа токенизаторов, входящих в состав библиотеки NLTK, носит название «простых» (simple) токенизаторов. Она включает в себя 3 токенизатора: SpaceTokenizer, TabTokenizer и LineTokenizer.
     1. **SpaceTokenizer**

Разбивает строку на токены, используя в качестве разделителя пробел. Результат работы этого метода полностью совпадает с результатом метода s.split(‘ ‘).

* + 1. **TabTokenizer**

Аналогичен SpaceTokenizer, только для разделения строки использует символ табуляции, что совпадает с применением метода s.split(‘\t’).

* + 1. **LineTokenizer**

Аналогичен SpaceTokenizer, за исключением того, что для разделения строки использует символ новой строки (при необходимости может отбрасывать пустые строки). Его применение похоже на применение метода s.split(‘\n’).

* 1. **MWETokenizer (Multi-Word Expression Tokenizer)**

Токенизатор берет строку, которая уже была разделена на токены, и повторно токенизирует ее, объединяя выражения из нескольких слов в отдельные токены, используя словарь MWE. Это может быть полезно в том случае, когда в тексте встречаются устойчивые выражения или именованные сущности, состоящие из нескольких слов.

Из недостатков: токенизатору необходим предварительно заданный список выражений из нескольких слов для определения, существует ли какое-либо из них в тексте. Кроме того, он чувствителен к регистру слов, а также не определяет различные словоформы и написания одного и того же слова.

* 1. **ToktokTokenizer**

Данный токенизатор позиционируется как простой универсальный токенизатор. На вход ему подается по одному предложению в строке. Работа основана на последовательном применении к исходному тексту списка регулярных выражений. С их помощью происходят, например, замена неразрывных пробелов обычными, замена знака многоточия единичной точкой, расстановка пробелов после открывающих и перед закрывающими знаками препинания. В итоговой строке удаляются начальные и конечные пробелы, а сама строка преобразуется в кодировку Unicode.

Согласно документации, он был протестирован и дает достаточно хорошие результаты для английского, персидского, русского, чешского, французского, немецкого, вьетнамского, таджикского и некоторых других языков. Входные данные должны быть представлены в кодировке UTF-8.

* 1. **TweetTokenizer**

Это токенизатор с поддержкой особенностей коротких сообщений Twitter, разработанный для гибкой и простой адаптации к новым предметным областям и задачам. Основная логика его работы заключается в следующем:

* определяется список регулярных выражений, для поиска в исходном тексте гиперссылок, смайликов, телефонных номеров, имен пользователей Twitter, хештегов и другого;
* регулярные выражения помещаются по порядку в скомпилированный объект регулярного выражения, называемый word\_re;
* выполняется токенизация с помощью метода word\_re.findall(s), где s – строка, исходного текста.
  1. **TreebankWordTokenizer**

Токенизатор Treebank использует регулярные выражения для разделения текста на слова так, как это реализовано в Penn Treebank. Для этого последовательно выполняются следующие шаги:

1. разделяются стандартные сокращения, например, «don’t» **->** «do n’t» или «they’ll» **->** «they ‘ll» для английского языка;
2. большинство знаков препинания, разделяются на отдельные токены;
3. разделяются запятые и одинарные кавычки, если за ними следует пробел;
4. отделяются точки, которые проставлены в конце строк.
   1. **NLTKWordTokenize**r

Это улучшенный вариант TreebankWordTokenizer, включающий несколько адаптированных списков сокращений.

Стоит отметить, что, по сравнению с TreebankWordTokenizer, NLTKWordTokenizer является «деструктивным» токенизатором. То есть, применяемые регулярные выражения приводят исходный текст в состояние, не поддающееся восстановлению.

* 1. **StanfordSegmenter**

Некоторые языки требуют более обширной предварительной обработки токенов, которая обычно называется сегментацией. Например, StanfordSegmenter предназначен для «токенизации» или «сегментации» слов китайского или арабского текста (также можно использовать для английского, французского и испанского языков).

Исходя из описания токенизаторов библиотеки nltk, в данной работе могут быть использованы для сравнения следующие токенизаторы: WhitespaceTokenizer, WordPunctTokenizer, SpaceTokenizer, ToktokTokenizer, **TweetTokenizer,** NLTKWordTokenizer**.**

1. **Библиотека SpaCy.**

Одна из самых популярных библиотек NLP наряду с NLTK. Основное различие между двумя библиотеками заключается в том, что NLTK содержит широкий спектр алгоритмов для решения одной проблемы, тогда как spaCy содержит только один, но лучший алгоритм для решения проблемы.

При создании текстового документа SpaCy автоматически разбивает документ на токены, распознает части речи токенов и их зависимости в предложении. Так же с помощью spaCy можно обнаруживать именованные сущности (имена, названия компаний, мест, зданий, учреждений, валют и другие). (<https://rukovodstvo.net/posts/id_1131/>) Поиск n-грамм с помощью texstacy

1. **Библиотека StanfordCoreNLP.**

Здесь тоже присутствует токенизация по словам и предложениям, лемматизация, тегирование и распознавание именных сущностей. (<https://rukovodstvo.net/posts/id_1092/>) Написана для java

1. **Библиотека Pattern.**

Это многоцелевая библиотека, способная решать следующие задачи:

* Обработка естественного языка: выполнение таких задач, как токенизация, стемминг, теги POS, анализ настроений и т. Д.
* Data Mining: он содержит API-интерфейсы для сбора данных с таких сайтов, как Twitter, Facebook, Wikipedia и т. Д.
* Машинное обучение: содержит модели машинного обучения, такие как SVM, KNN и персептрон, которые можно использовать для задач классификации, регрессии и кластеризации.

Касаемо решаемой задачи, в данной библиотеке нет отдельной функции для токенизации, но есть функция, которая включает токенизацию, маркировку частей речи, ролей и лемматизацию. Также есть возможность поиска n-грамм, преобразования чисел в текстовое представление и некоторые другие функции. (<https://rukovodstvo.net/posts/id_1099/>) https://stackabuse.com/python-for-nlp-introduction-to-the-pattern-library/

1. **Библиотека Gensim.**

Gensim – это библиотека, которая используется для тематического моделирования без учителя и обработки естественного языка (NLP). Она предназначена для извлечения семантических тем из документов, где под темой понимается повторяющаяся группа слов, часто встречающихся вместе.

Gensim может работать с большими текстовыми коллекциями. Этим она отличается от других программных библиотек машинного обучения, ориентированных на обработку в памяти. GenSim также предоставляет эффективные многоядерные реализации различных алгоритмов для увеличения скорости обработки. В нее добавлены более удобные средства для обработки текста, чем у конкурентов, таких как Scikit-learn, R и т. д.

В библиотеке реализована токенизация на слова (униграммы) и словосочетания (биграммы и триграммы), а также другие процедуры обработки токенов. (https://pythonru.com/biblioteki/gensim)

1. **Библиотека Scikit-learn.**

Scikit-learn – это библиотека машинного обучения с открытым исходным кодом, которая поддерживает контролируемое и неконтролируемое обучение. Здесь также представлены различные инструменты для подгонки модели, предварительной обработки данных, выбора модели, оценки модели и многих других утилит.

Касаемо токенизации, в Scikit-learn имеется возможность токенизации как на униграммы, так и на n-граммы. (https://scikit-learn.ru/6-2-feature-extraction/)

1. **Библиотека Keras.**

Класс Tokenizer Keras используется для токенизации текста на отдельные слова, но не работает для n-грамм. Конечной целью использования класса является векторизация текстового корпуса. (https://machinelearningknowledge.ai/keras-tokenizer-tutorial-with-examples-for-fit\_on\_texts-texts\_to\_sequences-texts\_to\_matrix-sequences\_to\_matrix/)

1. **Rutokenizer.**

Пакет содержит набор классов, которые умеют выделять предложения из текста (сегментация) и разбивать предложения на слова (токенизация) с учетом многословных единиц текста типа "из-за" или "какой-то". Однако здесь не поддерживается распознавание n-грамм. (<https://github.com/Koziev/rutokenizer>)

1. **TextBlob**

Здесь предложено множество встроенных методов для обычных задач обработки естественного языка, среди которых: извлечение именных фраз, разметка части речи, анализ настроений, классификация текста, токенизация (в том числе n-граммы), частота слов и фраз, словофлексия, исправление орфографии. (https://biconsult.ru/products/textblob-v-python-rukovodstvo)

В качестве резюмирования вышеперечисленных характеристик инструментов были составлены таблицы (см. табл. 1-5). В них наглядно продемонстрированы их сходства и различия.

Таблица 1 – Характеристика инструментов токенизации библиотеки NLTK

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Инструменты | Критерии | | | |
| Поиск униграмм | Поиск n-грамм (где n>1) | Поддержка автоматической обработки | Поддержка обработки русского языка |
| **RegexpTokenizer** | Есть | Нет | Есть | Есть |
| WhitespaceTokenizer | Есть | Нет | Есть | Есть |
| BlanklineTokenizer | Нет | Нет | Есть | Есть |
| WordPunctTokenizer | Нет | Нет | Есть | Есть |
| SpaceTokenizer | Есть | Нет | Есть | Есть |
| TabTokenizer | Нет | Нет | Есть | Есть |
| LineTokenizer | Нет | Нет | Есть | Есть |
| MWETokenizer | Нет | Есть | Нет | Есть |
| ToktokTokenizer | Есть | Нет | Есть | Есть |
| **TweetTokenizer** | Есть | Нет | Есть | Есть |
| **TreebankWordTokenizer** | Есть | Нет | Есть | Есть |
| NLTKWordTokenizer | Есть | Нет | Есть | Есть |
| StanfordSegmenter | Есть | Нет | Есть | Нет |

В первой таблице (см. табл. 1) представлены токенизаторы библиотеки NLTK и их характеристики по критериям возможности поиска n-грамм, применения инструмента к русскоязычному тексту и наличие автоматизации работы инструмента.

Из всех токенизаторов из таблицы 1 только один не работает с русским языком – StanfordSegmenter.

Инструменты BlanklineTokenizer, WordPunctTokenizer, TabTokenizer и LineTokenizer предназначены для токенизации на более крупные единицы, такие как предложения, абзацы и так далее.

MWETokenizer не поддерживает автоматизированную работу: **для работы** MWETokenizer требуется предварительное составление словаря устойчивые выражения или именованные сущности для их дальнейшего поиска в тексте.

При рассмотрении **TreebankWordTokenizer** и NLTKWordTokenizer необходимо отметить, что они очень похожи, но второй представляет собой улучшенный вариант первого. Поэтому было решено из двух инструментов выбрать один **–** NLTKWordTokenizer.

По итогу осталось 7 токенизаторов, применимых для разделения на униграммы (**RegexpTokenizer,** WhitespaceTokenizer, SpaceTokenizer, ToktokTokenizer, TweetTokenizer, NLTKWordTokenizer) и ни одного для n-грамм.

Таблица 2 – Обобщенная характеристика инструментов токенизации

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Инструменты | Критерии | | | |
| Поиск униграмм | Поиск n-грамм (где n>1) | Поддержка обработки русского языка | Возможность реализации дополнительных процедур (кроме токенизации) |
| Re | Есть | Нет | Есть | Нет |
| Nltk | Есть | Есть | Есть | Есть |
| SpaCy | Есть | Есть | Есть | Есть |
| StanfordCoreNLP | Есть | Нет | Есть | Есть |
| Pattern | Есть | Есть | Есть | Есть |
| Gensim | Есть | Есть | Есть | Есть |
| Scikit-learn | Есть | Есть | Есть | Есть |
| Keras | Есть | Нет | Есть | Есть |
| Rutokenizer | Есть | Нет | Есть | Нет |
| TextBlob | Есть | Есть | Есть | Есть |

В таблице 2 рассмотрены характеристики инструментов токенизации по следующим критериям: поиск 1- и n-грамм, поддержка обработки русскоязычных текстов и наличие возможностей реализации дополнительных процедур обработки токенов.

Модули библиотеки NLTK объединены в данной таблице, так как по многим свойствам, важных для данной таблицы, они схожи. Однако в NLTK есть модули (ngrams, bigrams, trigrams), с помощью которых можно распознавать n-граммы. Есть также модуль everygrams, однако в нем нельзя регулировать значение n, n всегда равно количеству слов в предложении.

По итогам сравнения, все перечисленные в таблице инструменты поддерживают работу с русским языком. Однако два из них (Re и Rutokenizer) проигрывают на фоне остальных из-за невозможности реализации дополнительных процедур обработки токенов.

Также было принято решение исключить StanfordCoreNLP из дальнейшего исследования, потому что библиотека написана на java, что затрудняет ее использование.

Оставшиеся инструменты можно разделить на подходящие для поиска униграмм (Nltk, SpaCy, StanfordCoreNLP, Pattern, Gensim, Scikit-learn, Keras, TextBlob) и n-грамм (Nltk, SpaCy, Pattern, Gensim, Scikit-learn, TextBlob).

Далее будут отдельно рассмотрены кандидаты двух этих групп.

Первая группа будет сравниваться по времени работы и по чистоте токенов (см. табл. 3). Время работы будет измерено с помощью библиотеки time, числом в таблице будет среднее между 5-ю измерениями (см. Приложение А). Под чистотой токенов понимаются списки токенов, состоящих из слов и не содержащих знаков препинания, которые некоторые алгоритмы определяют как токены.

В таблице 3 будут сравниваться все инструменты из таблицы 2, так как кандидаты из таблицы 1 также не имеют возможности реализации дополнительных процедур для токенов, как и Re и Rutokenizer.

Таблица 3 – Итоговая таблица сравнения для токенизации по словам

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Инструменты | Критерии | | | |
| Время работы, с | Чистота токенов | | |
| 1 | 2 | 3 |
| **RegexpTokenizer** | 0.00064 | + | - | - |
| WhitespaceTokenizer | 0.00025 | - | + | + |
| SpaceTokenizer | 0.00002 | - | + | + |
| ToktokTokenizer | 0.00072 | + | + | + |
| TweetTokenizer | 0.09419 | + | - | + |
| NLTKWordTokenizer | 0.00925 | + | + | + |
| Re | 0.00045 | + | - | - |
| SpaCy | 0.00033 | + | + | + |
| Pattern | 2.42913 | + | + | + |
| Gensim | 0.00021 | + | - | - |
| Scikit-learn | 0.00116 | + | - | - |
| Keras | 0.00009 | + | - | - |
| Rutokenizer | 0.01642 | + | - | + |
| TextBlob | 0.00116 | + | + | + |

Исследование времени проводилось на предложении «Пушкин неоднократно писал о своей родословной в стихах и прозе; он видел в своих предках образец истинной «аристократии», древнего рода, честно служившего отечеству, но не снискавшего благосклонности правителей и «гонимого».».

Исследование по критерию «Частота токенов» проводилась по четырем предложениям: 1 – «Не ветер, а какой-то ураган!» (проверка распознавания знаков препинания), 2 – «Я - к.т.н., живу в США.» (проверка на распознавание аббревиатур), 3 – «В Нью-Йорке (США) хорошие маффины стоят $3.88.» (проверка на распознавание цифр, чисел с плавающей точкой).

Расшифровка «Частоты токенов»:

* по итогу первого эксперимента «+» получают инструменты, распознающие знаки препинания как отдельные токены или исключающие их из списка токенов сразу. Остальные инструменты получают «-»;
* по итогу второго эксперимента «+» получают инструменты, распознающие обе аббревиатуры («к.т.н.» и «США») как токены. Остальные инструменты получают «-»;
* по итогу третьего эксперимента «+» получают инструменты, распознающие число с плавающей точкой как один токен. Остальные инструменты получают «-».

По количеству плюсов в графе «Чистота токенов»:

* 1 «+» у **RegexpTokenizer, Re,** Gensim, Scikit-learn, Keras;
* 2 «+» у WhitespaceTokenizer, SpaceTokenizer, TweetTokenizer, Rutokenizer;
* 3 «+» у ToktokTokenizer, NLTKWordTokenizer, SpaCy, Pattern и TextBlob.

Pattern исключается из сравнения по причине сравнительно большого времени работы, превышающее среднее время работы других инструментов в несколько раз.

WhitespaceTokenizer и SpaceTokenizer исключаются из сравнения по причине того, что в результатах их работы к токенам «приклеиваются» знаки препинания. В будущем это помешает правильному распределению признаков и может привести к неверно выполненной задаче основного алгоритма.

Инструменты **RegexpTokenizer, Re,** Gensim, Scikit-learn, Keras исключаются из сравнения из-за того, что у них из четырех возможных всего один плюс.

Оставшиеся инструменты были ранжированы по времени работы (см. табл. 4):

Таблица 4 – Оставшиеся инструменты токенизации на униграммы

|  |  |  |
| --- | --- | --- |
| Инструменты | Время работы, с | Количество «+» |
| ToktokTokenizer | 0.00072 | 3 |
| TweetTokenizer | 0.09419 | 2 |
| NLTKWordTokenizer | 0.00925 | 3 |
| SpaCy | 0.00033 | 3 |
| Rutokenizer | 0.01642 | 2 |
| TextBlob | 0.00116 | 3 |

Из оставшихся инструментов у TweetTokenizer и Rutokenizer самое большое время работы при самом малом количестве плюсов, поэтому они исключаются из сравнения.

Все четыре оставшихся инструмента (ToktokTokenizer, NLTKWordTokenizer, SpaCy, TextBlob) являются достойными и имеют одинаковое (максимальное) количество плюсов. Было решено распределить «призовые» места в соответствии с возрастанием времени работы:

* + - 1. SpaCy
      2. ToktokTokenizer
      3. TextBlob
      4. NLTKWordTokenizer

Далее (см. табл. 6) инструменты токенизации на 2 и 3-граммы сравниваются по наличию возможностей дополнительной обработки токенов и по времени токенизации.

Однако в библиотеке NLTK содержатся несколько подходящих модулей. Их свойства исследуются в таблице ниже (см. табл. 5). Полная таблица с проведением проверки времени предложена в Приложении Б.

Таблица 5 – Модули NLTK для токенизации на 2 и 3-граммы

|  |  |  |
| --- | --- | --- |
| Инструменты | n | Среднее время работы инструмента, с |
| Bigram | 2 | 0.00041 |
| Trigram | 3 | 0.00052 |
| Ngrams | 2 | 0.00060 |
| 3 | 0.00055 |

Для проверки инструментов NLTK внутри них использовался токенизатор для униграмм ToktokTokenizer, являющийся лучшим по итогам предыдущего исследования.

Из таблицы 5 видно, что для n = 2 более оптимальным будет использование инструмента Bigram, а для n = 3 – Trigram. При этом результат декомпозиции текста сравниваемых инструментов одинаков. Исходя из данного исследования, в таблицу 6 для модуля NLTK введены данные Bigram и Trigram.

Таблица 6 – Итоговая таблица сравнения для токенизации на 2 и 3-граммы

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Инструменты | Критерии | | | | | | |
| Время работы, с | | Дополнительные шаги внутри токенизатора | Лемматизация | Распознавание частей речи | Распознавание именованных сущностей | Распознавание зависимостей слов |
| n = 2 | n = 3 |
| Nltk | 0.00041 | 0.00052 | - | + | + | + | + |
| SpaCy | 0.00060 | 0.00043 | + | + | + | + | + |
| Pattern | 2.33229 | 2.46947 | + | + | + | - | - |
| Gensim | --- | --- | + | + | - | - | - |
| Scikit-learn | 0.00101 | 0.00177 | + | + | - | - | - |
| TextBlob | 0.00966 | 0.00895 | + | + | + | + | + |

Инструмент Gensim имеет инструмент распознавания n-грамм, но в его документации под n-граммами понимаются значимые словосочетания, такие как «искусственный интеллект», «машинное обучение» и так далее. Для русского языка данный инструмент пока не обучен до уровня, чтобы его можно было применять в прикладной разработке, в отличие от исследовательской. Данный инструмент выбывает из сравнения.

Необходимый шаг для работы алгоритма поиска ключевых слов, следующего за алгоритмом декомпозиции, – лемматизация, есть у всех кандидатов. Остальные распознаватели (частей речи, именных сущностей, зависимостей слов) понадобятся для оптимизации предобработки текста, поэтому в дальнейшее сравнение были взяты те инструменты, в которых присутствуют данные распознаватели. На данном шаге выбыли Scikit-learn и Pattern.

При рассмотрении колонки со временем, инструмент Pattern снова выделяется, и это вторая причина его исключения из сравнения. Напротив Pattern можно увидеть сильно отличающиеся от остальных в худшую сторону значения времени.

При сравнении времени работы оставшихся инструментов получается следующее ранжирование при n = 2: 1 место – NLTK, 2 место – SpaCy, 3 место – TextBlob; при n = 3: 1 место – SpaCy, 2 место – NLTK, 3 место – TextBlob.

Что касается дополнительных шагов внутри токенизатора, то в NLTK они отсутствуют (их нужно подключать и настраивать отдельно), в SpaCy они подключены сразу (сюда входит фильтрация стоп-слов, лишних знаков препинания), в TextBlob есть фильтрация второстепенных знаков препинания.

Таким образом, по итогам анализа таблицы 6 в SpaCy самая хорошая обработка (в сравнении с аналогами) в совокупности с самым малым временем работы.

Далее (см. табл. 8) инструменты токенизации на 1-2-3-граммы сравниваются по наличию возможностей дополнительной обработки токенов и по времени токенизации.

В библиотеке NLTK содержатся несколько подходящих модулей. Их свойства исследуются в таблице ниже (см. табл. 7). Полная таблица с проведением проверки времени предложена в Приложении Г.

Таблица 7 – Модули NLTK для токенизации на 1-2-3-граммы

|  |  |
| --- | --- |
| Инструменты | Среднее время работы инструмента, с |
| Unigram + Bigram + Trigram | 0.00080 |
| Ngrams | 0.00093 |
| Everygrams | 0.00079 |

Для проверки инструментов NLTK внутри них использовался токенизатор для униграмм ToktokTokenizer, являющийся лучшим по итогам предыдущего исследования.

Из таблицы 7 видно, что более оптимальным будет использование инструмента Everygrams. При этом результат декомпозиции текста сравниваемых инструментов одинаков. Исходя из данного исследования, в таблицу 8 для модуля NLTK введены данные Everygrams.

Таблица 8 – Итоговая таблица сравнения для токенизации на 1-2-3-граммы

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Инструменты | Критерии | | | | | |
| Время работы, с | Дополнительные шаги внутри токенизатора | Лемматизация | Распознавание частей речи | Распознавание именованных сущностей | Распознавание зависимостей слов |
| Nltk | 0.00079 | - | + | + | + | + |
| SpaCy | 0.00081 | + | + | + | + | + |
| Pattern | 2.45547 | + | + | + | - | - |
| Gensim | --- | + | + | - | - | - |
| Scikit-learn | 0.00178 | + | + | - | - | - |
| TextBlob | 0.01079 | + | + | + | + | + |

В данном случае инструмент Gensim выбывает из сравнения по тем же причинам, что и в предыдущий раз (пока не готов к применению в прикладной разработке).

Наличие и отсутствие различных распознавателей и лемматизатора в сравниваемых инструментах также описано внутри анализа таблицы 6, поэтому можно сразу сказать, что из сравнения исключаются инструменты Scikit-learn и Pattern.

При сравнении времени работы оставшихся инструментов получается следующее ранжирование при n = 2: 1 место – NLTK, 2 место – SpaCy, 3 место – TextBlob. При рассмотрении дополнительных шагов внутри токенизатора, влияющих на качество результата, видно, что в NLTK они отсутствуют, однако при практически таком же времени работы SpaCy фильтрует стоп-слова и лишние знаки препинания. TextBlob при значительно большем времени работы, чем SpaCy, имеет меньше дополнительной фильтрации (здесь она направлена только на знаки пунктуации).

Таким образом, по итогам анализа таблицы 8 в SpaCy самая хорошая обработка текста (в сравнении с аналогами) в совокупности с самым малым временем работы.

В результате обзора и сравнения инструментов декомпозиции текста на каждом этапе был выбран лучший инструмент в качестве базового для дальнейшей реализации разрабатываемых алгоритмов. Так вышло, что и для токенизации на униграммы, и на би- и триграммы, и для смешанной токенизации наилучшим инструментом оказался в SpaCy. Он отлично прошел тестирование качества результатов декомпозиции и имеет лучшее время работы среди аналогов, также хорошо прошедших тестирование качества. Также SpaCy имеет улучшенную декомпозиции за счет дополнительных процедур обработки текста (фильтрация стоп-слов, лишних знаков препинания).

* 1. **Выводы**

В результате обзора предметной области были определены основные термины, необходимые для понимания рассматриваемой темы. При рассмотрении существующих аналогов для решения задачи декомпозиции были определены: реализуемые процедуры обработки текста, наличие токенизации на слова и словосочетания, наличие поддержки обработки русского языка и автоматизированной обработки, временные затраты на применение и качество результатов относительно поставленных требований. Один инструмент (SpaCy) был выбран в качестве базового для реализации алгоритмов декомпозиции. Таким образом, было проведено сравнение и анализ существующих инструментов декомпозиции текста.

1. **ПОСТАНОВКА ЗАДАЧИ**

Уже не раз отмечалось, что наличие ключевых словосочетаний (фраз) помогает более точно определить тематику текста и, соответственно, улучшить его классификацию/категоризацию. Поэтому в данной работе будут реализованы алгоритмы семантической декомпозиции как на слова, так и на словосочетания, чтобы в последующих исследованиях (не в данной работе) сравнить результаты и сделать выводы о влиянии выделения ключевых словосочетаний относительно отдельных ключевых слов на результаты задачи категоризации. (думаю сделать один для ключевых слов, второй для словосочетаний и третий гибридный)

Исходя из обзора предметной области и анализа проблемы, описанной во введении, созданы критерии к разрабатываемым алгоритмам.

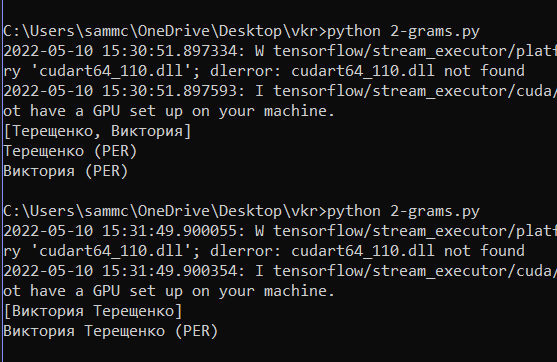
Алгоритм должен решать следующие проблемы:

* основные:

1. декомпозиция текста на ключевые элементы;
2. задача должна решаться за минимально возможное время, не превышающее 5 секунд.
3. Наличие интерфейса для взаимодействия с пользователем
4. Программный вариант встраивания алгоритма в систему СППР

* второстепенные:

1. алгоритм может содержать вспомогательные процедуры обработки токенов:
   1. фильтрация (удаление стоп слов, цифр);
   2. лемматизация;
   3. определение частей речи
   4. поиск именованных сущностей



* 1. поиск n-грамм, состоящих из определенных частей речи
  2. определение роли в предложении
  3. определение зависимостей в предложении (+ схема)
  4. поиск синонимов

1. алгоритм может переключаться между декомпозицией на униграммы, на биграммы, на триграммы или на смешанную декомпозицию.

<https://proglib.io/p/fun-nlp>

https://habr.com/ru/post/531940/

Уточнение понятие токена: токенами являются слова, состоящие из букв или цифр; знаки пунктуации и служебные символы токенами не являются.

1. **Описание Теоретических основ разрабатываемого алгоритма**

Если рассмотреть общую структуру алгоритма категоризации (см. рис. Такой-то), можно понять, что алгоритм семантической декомпозиции будет находиться в самом его начале, среди процедур предобработки текста.

Рисунок такой-то

* 1. **Графическое описание алгоритма семантической декомпозиции**

Схема построения алгоритма декомпозиции представлена на рис. Таком-то

Рисунок такой-то

* 1. **Лингвистические основы алгоритма семантической декомпозиции**
     1. **Основные процедуры алгоритма**

Процедуры основной части алгоритма декомпозиции (см. рис. Такой-то):

* 1. Сегментация предложений: текст делится на предложения. Это вспомогательная процедура для дальнейшего шага. Она нужна для более сложной семантической обработки (учет границ предложений).
  2. Токенизация: тест делится на токены. Это основная процедура. Здесь происходит декомпозиция текста на ключевые элементы (токены).

Перед началом работы алгоритма пользователю будет предложено выбрать вариант нужной ему декомпозиции:

1. Чему равно значение n?
2. Быстрая декомпозиция (по умолчанию – с фильтрацией стоп-слов и/или цифр) или с выбором критериев:

* декомпозиция без фильтрации;
* декомпозиция с фильтрацией стоп-слов и/или цифр;
* декомпозиция с отбором токенов, содержащих определенные части речи.
  + 1. **Дополнительные процедуры алгоритма**

Помимо основной части алгоритм имеет реализованные дополнительные процедуры обработки токенов. Они могут быть полезны в будущем для выполнения дальнейших шагов задачи категоризации:

* Поиск именованных сущностей (если говорить об СППР для службы технической поддержки компании Газпром переработка, в которой может в будущем применяться разрабатываемый алгоритм) поможет фильтровать имена и фамилии клиентов, которые могут идентифицироваться важными словами в тексте заявки, а по своей сути не иметь смысла.
* Лемматизация приводит токены в начальную форму. Это означает, что при необходимости подсчета частоты встречаемости токенов в тексте различные словоформы будут посчитаны вместе, как разные формы одного слова, а не как разные слова. Соответственно, вес данного слова возрастет.
* Определение роли токена и зависимостей в предложении может быть также применимо для определения веса токена при необходимости использования семантических данных.
* Поиск синонимов к заданным словам может быть применим при подготовке размеченных данных в случае, если данных будет мало.
* Фильтрация (удаление стоп-слов и цифр), определение частей речи и поиск n-грамм, состоящих из определенных частей речи, подключены к основной части алгоритма и помогают получать более чистый результат, что сокращает время работы и увеличивает точность следующих шагов алгоритма категоризации.
  1. **Математические основы алгоритма семантической декомпозиции**

Рассмотрим процесс фильтрации подробнее.

Декомпозиция текста на n-граммы основывается на декомпозиции на униграммы. Данный процесс изображен на рисунке ниже:

Рисунок такой-то (как происходит поиск униграмм, биграмм, триграмм)

На рисунке Таком-то изображена логическая схема формирования n-грамм. Глядя на него, становится понятна важность фильтрации стоп-слов (и других ненужных символов, которые могут быть распознаны как токены) для процесса декомпозиции на n-граммы, так как любое стоп-слово при выделении биграмм появится в двух токенах, при выделении триграмм – в трех и так далее. Так, фильтрация контролирует и размерность выходных данных, и сохраняет пользу процесса предобработки.

* 1. **Выводы**

В данной главе были подробно описаны основные моменты касаемо теоретической части построения алгоритма семантической декомпозиции текста: определены основные и вспомогательные процедуры алгоритма, их назначение, необходимость. Основными процедурами алгоритма являются сегментация (вспомогательная) и токенизация (основная). Алгоритм также содержит ряд дополнительных процедур обработки токенов.

1. **ПРОГРАММНАЯ РЕАЛИЗАЦИЯ АЛГОРИТМОВ**
   1. **Технические подробности программной реализации алгоритма**

Для реализации алгоритма был выбран язык Python версии 3.10. Решение использовать Python было принято потому, что данный язык программирования является активно применяется для работы с естественным языком и имеет готовые базовые инструменты для работы с текстом. Второй причиной является то, что на данном языке была написана система СППР для системы поддержки компании Газпром переработка.

Ниже дается описание основных библиотек и фреймворков, задействованных при разработке.

* + 1. **SpaCy**

spaCy — это библиотека для расширенной обработки естественного языка в Python и Cython. Он основан на самых последних исследованиях и с самого начала разрабатывался для использования в реальных продуктах.

spaCy поставляется с предварительно обученными конвейерами и в настоящее время поддерживает токенизацию и обучение для более чем 60 языков. Он включает в себя самые современные модели скорости и нейронной сети для тегов, синтаксического анализа, распознавания именованных объектов, классификации текста и многого другого, многозадачное обучение с предварительно обученными преобразователями, такими как BERT, а также готовую к производству систему обучения и простую модель. упаковка, развертывание и управление рабочим процессом. spaCy — это коммерческое программное обеспечение с открытым исходным кодом, выпущенное под лицензией MIT.

Функции

Поддержка более 60 языков

Обученные пайплайны для разных языков и задач

Многозадачное обучение с предварительно обученными преобразователями, такими как BERT

Поддержка предварительно обученных векторов слов и вложений

Непревзойденная скорость

Готовая к производству система обучения

Лингвистически мотивированная токенизация

Компоненты для распознавания именованных сущностей, тегирования частей речи, анализа зависимостей, сегментации предложений, классификации текста, лемматизации, морфологического анализа, связывания сущностей и т. д.

Легко расширяемый с помощью настраиваемых компонентов и атрибутов

Поддержка пользовательских моделей в PyTorch, TensorFlow и других фреймворках.

Встроенные визуализаторы синтаксиса и NER

Простая упаковка модели, развертывание и управление рабочим процессом

Надежная, тщательно оцененная точность

<https://github.com/explosion/spaCy>

https://spacy.io/

* + - 1. **Модель Russian**
      2. **Модель Ru2**
      3. **Модель Ru\_core\_news\_sm**
    1. **Textacy**

Это библиотека Python для выполнения различных задач обработки естественного языка (NLP), построенная на основе высокопроизводительной библиотеки spaCy. Ее специализация – токенизация, маркировка частей речи, анализ зависимостей и т. д. На данный момент не весь ее функционал применим для русского языка, однако базовые процедуры, необходимые для разработки в данной работе, работают и с ним.

Особенности

Доступ и расширение основных функций spaCy для работы с одним или несколькими документами с помощью удобных методов и пользовательских расширений.

Загружайте подготовленные наборы данных как с текстовым содержимым, так и с метаданными, от речей Конгресса до исторической литературы и комментариев Reddit.

Очищайте, нормализуйте и исследуйте необработанный текст перед его обработкой с помощью spaCy.

Извлечение структурированной информации из обработанных документов, включая n-граммы, сущности, аббревиатуры, ключевые термины и триплеты SVO.

Сравнивайте строки и последовательности, используя различные показатели сходства.

Токенизируйте и векторизируйте документы, затем обучайте, интерпретируйте и визуализируйте тематические модели.

Вычислять статистику удобочитаемости текста, включая уровень успеваемости по Флешу-Кинкейду, индекс SMOG и многоязычную легкость чтения по Флешу.

<https://textacy.readthedocs.io/en/latest/> https://buildmedia.readthedocs.org/media/pdf/textacy/latest/textacy.pdf

* + 1. **DisplaCy**

DisplaCy – визуализатор НЛП. Его можно использовать для визуализации отношений POS (ролей и зависимостей в предложении), обнаруженных с помощью синтаксического анализатора spaCy. <https://github.com/nineinchnick/displacy> https://spacy.io/usage/visualizers

* + 1. **Time**
  1. **Тестирование алгоритма**

Лалалалал

Нет, тестирование здесь это просто проверка программной реализации алгоритма на отсутствие ошибок и проверка его работоспособности, а исследование предполагает определение его отличительных потребительских свойств в предположении, что Вы работаете с полностью работоспособной проверенной Вами в результате тестирования программой.

* 1. **Сценарий использования**

После запуска веб-приложения работа с ним может проводиться по следующему сценарию:

1. На начальной странице пользователь вводит ID курса.
2. Веб-приложение загружает список разделов курса в порядке, в котором они находятся в МООК.
3. Пользователь выбирает интересующий раздел, кликая по нему.
4. На экране появляются три группы по 10 терминов из выбранного раздела в порядке от слова с самой большой вероятностью того, что оно является термином в данном разделе, к слову с наименьшим значением вероятности. Во главе каждой группы терминов выводится название метода, которым было проведено их выделение.
5. Пользователь кликает на термин.

Под выбранным термином появляется его определение.

* 1. **Применение алгоритма для решения задачи категоризации**

Текст поступает на вход в систему, где первым делом он проходит через предобработку, в которую входит алгоритм семантической декомпозиции. Его выходные данные подаются на вход алгоритму выделения ключевых слов, где токенам присваиваются веса и на их основе происходит разметка данных для решения задачи категоризации.

Возможно, будет какое-то программный вариант внедрения

программный пример его привязки к решению задачи категоризации. В этом же наша главная цель...

* 1. **Выводы**

Результатом разработки является готовое веб-приложение для генерации оглавления и списка терминов для МООК, в котором были реализованы алгоритмы выделения ключевых слов Frequency, TextRank и TopicRank. С помощью данных алгоритмов приложение проводит анализ русскоязычных курсов Stepik, в результате которого из содержимого разделов МООК извлекаются термины. К терминам подбираются определения из Википедии. Приложение интегрировано со Stepik API и с PostgreSQL и имеет пользовательский интерфейс.

1. **исследование**
   1. **Исследование времени работы**

Было проведено исследование времени работы алгоритмов Frequency, TextRank и Topic Rank.

* + 1. **Сценарий эксперимента**

Измеряемой величиной является время, затраченное на работу каждого из алгоритмов. Единица измерения – секунды.

Измерение проводилось путем фиксирования времени начала выполнения алгоритма и окончания его выполнения, затем найденная разность данных значений приравнивалась ко искомому значению затраченного времени.

Процедура проводилась 100 раз для каждого эксперимента.

* + 1. **Методика измерения**

Измерение происходит машине со следующими характеристиками:

* операционная система – Windows 10 Pro;
* системные характеристики:
  + процессор – Intel Core i3-7100U CPU 2.4GHz (2 ядра);
  + ОЗУ – 8Гб;
  + тип системы – х64.

Эксперимент проводился для текстового содержимого раздела «Функции. Словари. Интерпретатор. Файлы. Модули» курса «Программирование на Python» (ID 67).

* + 1. **Используемые программные средства**

Эксперимент проводился с использованием средств Python, а именно с помощью библиотеки time и ее метода monotonic() [32].

* + 1. **Результаты эксперимента**

Было рассчитано минимальное, максимальное и среднее время выполнения каждого алгоритма. Полученные данные, приведенные в секундах с точностью до тысячных, представлены в табл. 2.

Таблица 2 – Обработанные данные (для алгоритмов)

|  |  |  |  |
| --- | --- | --- | --- |
| Данные | Алгоритмы | | |
| Frequency | TopicRank | TextRank |
| Минимальное время работы | 8,032 | 5,892 | 7,504 |
| Среднее время работы | 8,761 | 6,556 | 8,165 |
| Максимальное время работы | 13,100 | 11,014 | 12,563 |

Таблица с необработанными данными представлена в прил. Б.

* + 1. **Сравнение и оценка результатов**

Как видно из расчетов, разность между максимальным и средним временем работы гораздо больше, чем разность между средним и минимальным временем, что говорит о том, что итерации, показавшие сравнимое с максимальным время, достигалось малое количество раз. Данную гипотезу можно проверить, ознакомившись с таблицей в приложении Б. Подобное поведение может быть связано с внешними факторами, такими как скорость интернет-соединения, время ответа сервера Stepik.

Сравнивая среднее время работы алгоритмов, видно, что лучшее время работы показывает TopicRank, алгоритм TextRank оказывается на втором месте, а Frequency на третьем. Все три алгоритма реализованы с применением различных готовых решений, имеющихся в библиотеках обработки текстов, что объясняет разную скорость работы алгоритмов. Работа методов таких библиотек не отличается быстродействием, что объясняет невысокую скорость выполнения алгоритмов.

* 1. **Исследование точности результатов**

В рассмотрении задач выделения ключевых слов большую роль играет точность их выделения.

* + 1. **Сценарий эксперимента**

В ходе эксперимента запускается веб-приложение, результатом чего становится получение трех списков с терминами. Далее слова в каждом из списков сравниваются со словами эталонного списка, состоящего из такого же количества ключевых слов, таким образом выделяется точность алгоритма в процентах.

Под эталонным списком понимается список отобранных вручную ключевых слов из того же документа.

В списках терминов и в эталонном списке содержится по 10 слов.

* + 1. **Входные данные**

Эксперимент проводится на базе текстового содержимого раздела «Введение» курса «Как писать научные статьи» (ID 10524).

* + 1. **Результат работы алгоритмов**

Результатом работы алгоритмов стали следующие ключевые слова:

1. Frequency:
   * научный;
   * статья;
   * работа;
   * курс;
   * комментарий;
   * исследование;
   * результат;
   * источник;
   * текст;
   * пример;
2. TopicRank:
   * научный;
   * работа;
   * курс;
   * комментарий;
   * исследование;
   * результат;
   * источник;
   * текст;
   * список;
   * пример;
3. TextRank:
   * метод;
   * явно;
   * вывод;
   * введение;
   * аннотация;
   * слово;
   * state;
   * анализ;
   * название;
   * совет.
     1. **Экспертная оценка**

Эталонный список содержит следующие ключевые слова:

* научный;
* статья;
* исследование;
* обоснованность;
* анализ;
* аннотация;
* курс;
* автор;
* публикация;
* источник.

Словосочетание «научная статья» было разделено ввиду того, что алгоритмы ведут поиск юниграмм – N-грамм, состоящих из одного слова.

* + 1. **Сравнение с экспертной оценкой**

В ходе сравнения было вычислено отношение числа совпавших слов к общему количеству слов в списке. Результаты сравнения представлены в табл. 3.

Таблица 3 – Сравнение с экспертной оценкой

|  |  |  |  |
| --- | --- | --- | --- |
| Данные | Алгоритмы | | |
| Frequency | TopicRank | TextRank |
| Точность, % | 50 | 50 | 20 |

Согласно таблице 3, алгоритм TextRank не применим к задаче выделения ключевых слов, так как его точность составляет всего 20%. Это можно объяснить тем, что данный алгоритм основан на алгоритме PageRank, для построения графа которого используются связи между элементами. В данной же реализации алгоритма на вход подается список уже отделенных друг от друга слов, что мешает работе алгоритма.

В связи с этим же фактом, алгоритм TopicRank, показавший средние результаты (50%), может улучшить свой результат при иной реализации этапа предобработки.

Алгоритм Frequency, показавший также средний результат (50%), тоже можно улучшить с помощью использования N-грамм. Однако, даже в этом случае, он с малой вероятностью покажет стопроцентный результат, так как не предназначен для специализированных текстов.

* 1. **Выводы**

Таким образом, было проведено исследование свойств программного продукта, таких как время работы алгоритмов, выполняющих выделение терминов из текста МООК, и их точность. В ходе исследования было выявлено, что алгоритм TopicRank быстрее всех справляется с задачей (6,556 с.) и имеет среднюю точность (50%); алгоритм TextRank на втором месте по времени (8,165 с.), но на последнем по точности (20%); Frequency на третьем месте по времени (8,761 с.) и имеет такую же точность, как и TopicRank – 50%.

1. **Коммерциализация результатов исследований в рамках НИР**

Здесь о том, что разработанный алгоритм до настоящего времени не внедрялся в программные продукты организации, но в будущем может быть использован в СППР для службы технической поддержки ООО ГП.

Пример: «Разработанное решение до настоящего момента не внедрялось в организации, но было опубликовано на Docker Hub. Docker Hub – это общедоступный реестр, поддерживаемый Docker, хранящий большую коллекцию Docker-образов».

# **заключение**

В результате работы над выпускной квалификационной работой были изучены походы к задаче выделения ключевых слов, выполнено проектирование программного продукта, представляющее собой веб-приложение, автоматически генерирующее оглавление и указатели терминов для массовых открытых онлайн-курсов платформы Stepik.

В процессе обзора предметной области были определены основные термины рассматриваемой области, дано описание подходов выделения ключевых слов, проанализированы области их применения, временные затраты на применение, наличие требований использования специфических средств для работы и особенности подходов. Проведен выбор двух подходов, реализуемых в дальнейшем в веб-приложении. Выбранными подходами стали статистический и графовый.

На основании обзора предметной области и анализа проблемы, описанной во введении, были выдвинуты требования к разрабатываемому решению. Учитывая данные требования, была достигнута основная задача работы – спроектирован программный продукт для автоматического генерирования оглавления и указателей терминов для русскоязычных массовых открытых онлайн-курсов. Разработанное решение представлено веб-приложением на языке программирования Python, с использованием веб-фреймворка Flask и СУБД PostgreSQL. Полученное программное решение имеет пользовательский интерфейс. Приложение интегрируется платформой Stepik с помощью REST-интерфейса Stepik API, а с хранилищем данных с помощью библиотеки **SQLAlchemy**. Посредством анализа алгоритмами Frequency, TopicRank и TextRank из выбранного раздела МООК выделяются термины, для которых имеются определения.

Было проведено исследование свойств программного решения: исследование времени работы алгоритмов выделения ключевых слов и их точности. По результатам исследования быстродействия алгоритмов самым быстрым оказался алгоритм TopicRank (6,556 с.), а самым медленным – Frequency (8,761 с.). Это объясняется использованием готовых решений для обработки текста, что сильно влияет на скорость работы алгоритмов. В результате исследования точности, ни один алгоритм не показал отличных результатов из-за неподходящего для них сценария реализации этапа предобработки (TopicRank, 50%; TextRank, 20%) или из-за неподходящего проверяемому алгоритму типа текста (Frequency, 50%).

В будущем данную проблем, а также проблему малой скорости работы планируется решить для алгоритмов, потенциально подходящих для задачи выделения терминов из содержимого МООК, то есть для графовых алгоритмов TopicRank и TextRank.

Кратко (на одну – две страницы) описать основные результаты работы, проанализировать их соответствие поставленной цели работы, показать рекомендации по конкретному использованию результатов исследования и перспективы дальнейшего развития работы.

Научная ценность данной работы заключается в улучшении качества результатов проекта автоматизации службы технической поддержки компании ООО ГПП.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Что такое МООС (Массовые открытые онлайн курсы) [Электронный ресурс]. URL: https://etu.ru/ru/on-line-obuchenie/mooc (дата обращения: 15.05.2020).
2. Чекалина Т. А., Тумандеева Т. В., Максименко Н. В. / Основные направления и перспективы развития онлайн-обучения // Профессиональное образование в России и за рубежом. – 2018. – (№) 3 (31). – С. 44-52. [Электронный ресурс] URL: https://cyberleninka.ru/article/n/osnovnye-napravleniya-i-perspektivy-razvitiya-onlayn-obucheniya/viewer (дата обращения: 15.05.2020).
3. Ванюшкин А. С., Гращенко Л. А. / Методы и алгоритмы извлечения ключевых слов // Новые информационные технологии в автоматизированных системах. – 2016. – (№) 19. – С. 85-93. [Электронный ресурс]. URL: https://cyberleninka.ru/article/n/metody-i-algoritmy-izvlecheniya-klyuchevyh-slov/viewer (дата обращения: 24.05.2020).
4. Manning C. D., Manning C. D., Schütze H. Foundations of statistical natural language processing. – MIT press, 1999. 680p.
5. Лазарева О. Ю., Боломутова М. С. / Методы выделения ключевых слов в контексте электронных обучающих систем // Молодой ученый. – 2016. – (№) 26 (130). С. 143-146. [Электронный ресурс] URL: https://moluch.ru/archive/130/35952/ (дата обращения: 16.05.2020).
6. Лекция. Задачи Data Mining. Классификация и кластеризация // НОУ ИНТУИТ. [Электронный ресурс] URL: https://www.intuit.ru/studies/courses/6/6/lecture/166 (дата обращения: 16.05.2020).
7. Keyword Extraction: A Guide to Finding Keywords in Text // MonkeyLearn. [Электронный ресурс] URL: https://monkeylearn.com/keyword-extraction/ (дата обращения: 17.05.2020).
8. Girish K.P. Keyword Extraction from a Single Document Using Centrality Measures / Pattern Recognition and Machine Intelligence, Kolkata, India, December 18–22, 2007 / Kolkata, 2007, pp. 503–510.
9. Litvak M. Graph-based Keyword Extraction for Single-Document Summarization / Proceedings of the Workshop on Multi-source Multilingual Information Extraction and Summarization, Manchester, United Kingdom, August 2008 / Manchester, 2008, pp. 17–24.
10. Григорьева Е. Г., Клячин В. А., Помельников Ю. В., Попов В. В. / Алгоритм выделения ключевых слов на основе графовой модели лингвистического корпуса // Вестник ВолГУ. Серия 2, Языкознание. – 2017. – Т. 16 № 2. – С. 58-67. [Электронный ресурс] URL: https://cyberleninka.ru/article/n/algoritm-vydeleniya-klyuchevyh-slov-na-osnove-grafovoy-modeli-lingvisticheskogo-korpusa/viewer (дата обращения: 24.05.2020).
11. Лекция. Принципы работы и структура Web-приложений на основе ASP.NET // НОУ ИНТУИТ [Электронный ресурс] URL: https://www.intuit.ru/studies/courses/1139/250/lecture/6422 (дата обращения: 19.05.2020).
12. Welcome to Python.org // Python [Электронный ресурс] URL: https://www.python.org/ (дата обращения: 20.05.2020).
13. О нас // Stepik [Электронный ресурс] URL: https://welcome.stepik.org/ru/about (дата обращения: 20.05.2020).
14. StepikOrg/Stepik-API: API documentation and Examples // GitHub [Электронный ресурс] URL: https://github.com/StepicOrg/Stepik-API (дата обращения: 20.05.2020).
15. Welcome to Flask // Flask Documentation (1.1.x) [Электронный ресурс] URL: http://flask.pocoo.org/ (дата обращения: 20.05.2020).
16. SQLAlchemy – The Database Toolkit for Python // SQLAlchemy [Электронный ресурс] URL: https://www.sqlalchemy.org/ (дата обращения: 20.05.2020).
17. PostgreSQL: The world’s most advanced open source database // PostgreSQL [Электронный ресурс] URL: https://www.postgresql.org/ (дата обращения: 20.05.2020).
18. Collections – Container datatypes // Python 3.8.3 documentation [Электронный ресурс] URL: https://docs.python.org/3/library/collections.html (дата обращения: 21.05.2020).
19. Boudinfl/pke: Python Keyphrase Extraction module // GitHub [Электронный ресурс] URL: https://github.com/boudinfl/pke (дата обращения: 21.05.2020).
20. Pke documentation // pke 1.8 documentation [Электронный ресурс] URL: https://boudinfl.github.io/pke/build/html/index.html (дата обращения: 23.05.2020).
21. Docker Documentation // Docker Documentation. [Электронный ресурс] URL: https://docs.docker.com/ (дата обращения: 20.05.2020).
22. Заглавная страница // Википедия – свободная энциклопедия. [Электронный ресурс] URL: https://ru.wikipedia.org/wiki/Заглавная\_страница (23.05.2020).
23. Русская Википедия // Википедия. [Электронный ресурс] URL: https://ru.wikipedia.org/wiki/Русская\_Википедия (дата обращения: 23.05.2020).
24. Bougouin A. TopicRank: Graph-Based Topic Ranking for Keyphrase Extraction / International Joint Conference on Natural Language Processing, Nagoya, Japan, 14-18 October 2013 / Nagoya, 2013, pp 543–551.
25. Unsupervised models (TopicRank) // pke 1.8 documentation [Электронный ресурс] URL: https://boudinfl.github.io/pke/build/html/unsupervised.html#topicrank (дата обращения: 23.05.2020).
26. Mihalcea R. TextRank: Bringing Order into Texts / EMNLP, July 2004 / EMNLP, July 2004, № 32.
27. Unsupervised models (TextRank) // pke 1.8 documentation [Электронный ресурс] URL: https://boudinfl.github.io/pke/build/html/unsupervised.html#textrank (дата обращения: 23.05.2020).
28. Алгоритм PageRank // Data Science [Электронный ресурс] URL: http://datascientist.one/pagerank-algorithm/ (дата обращения: 23.05.2020).
29. Docker Hub // Docker Hub [Электронный ресурс] URL: https://hub.docker.com/repository/docker/vika2108991/vkr (дата обращения 24.05.2020).
30. Moevm/bsc\_tereshchenko // GitHub [Электронный ресурс] URL: https://github.com/moevm/bsc\_tereshchenko (дата обращения: 24.05.2020).
31. Разработка модели алгоритма составления оглавления и указателя терминов для массовых открытых онлайн курсов. / Терещенко В. Н., Заславский М. М. // 73-я научно-техническая конференция профессорско-преподавательского состава СПбГЭТУ «ЛЭТИ», СПб, 5 февраля 2020 г. / СПбГЭТУ «ЛЭТИ», СПб, 2020.
32. 15.3. time – Time access and conversions // Python 2.7.18 documentation. [Электронный ресурс] URL: https://docs.python.org/2/library/time.html (дата обращения: 24.05.2020).

Приложение А

Таблица – Время работы токенизаторов униграмм

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Инструменты | Измерения времени, с | | | | | Среднее значение, с |
| 1 | 2 | 3 | 4 | 5 |
| **RegexpTokenizer** | 0.00100 | 0.0 | 0.00118 | 0.00103 | 0.0 | 0.00064 |
| WhitespaceTokenizer | 0.00066 | 0.0 | 0.0 | 0.00058 | 0.0 | 0.00025 |
| SpaceTokenizer | 0.0 | 0.0 | 0.0 | 0.0 | 0.00010 | 0.00002 |
| ToktokTokenizer | 0.00056 | 0.00100 | 0.00107 | 0.0 | 0.00095 | 0.00072 |
| TweetTokenizer | 0.09257 | 0.09634 | 0.09355 | 0.09338 | 0.09460 | 0.09419 |
| NLTKWordTokenizer | 0.00804 | 0.00955 | 0.00900 | 0.00961 | 0.01007 | 0.00925 |
| Re | 0.0 | 0.00100 | 0.00068 | 0.00055 | 0.0 | 0.00045 |
| SpaCy | 0.0 | 0.00096 | 0.00069 | 0.0 | 0.0 | 0.00033 |
| Pattern | 2.42281 | 2.45196 | 2.44199 | 2.38195 | 2.44693 | 2.42913 |
| Gensim | 0.00106 | 0.0 | 0.0 | 0.0 | 0.0 | 0.00021 |
| Scikit-learn | 0.00145 | 0.00104 | 0.00100 | 0.00104 | 0.00127 | 0.00116 |
| Keras | 0.0 | 0.0 | 0.00046 | 0.0 | 0.0 | 0.00009 |
| Rutokenizer | 0.01816 | 0.01580 | 0.01606 | 0.01614 | 0.01595 | 0.01642 |
| TextBlob | 0.00145 | 0.00156 | 0.00091 | 0.00100 | 0.00089 | 0.00116 |

Тестирование времени проводилось на предложении «Пушкин неоднократно писал о своей родословной в стихах и прозе; он видел в своих предках образец истинной «аристократии», древнего рода, честно служившего отечеству, но не снискавшего благосклонности правителей и «гонимого».».

Приложение Б

Таблица Б1 – время работы модулей NLTK для токенизации на 2-граммы (для Таблицы 6 основной части документа)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Инструменты | Измерения времени, с | | | | | Среднее время работы инструмента, с |
| 1 | 2 | 3 | 4 | 5 |
| Bigram | 0.00100 | 0.0 | 0.0 | 0.0 | 0.00104 | 0.00041 |
| Ngrams | 0.00096 | 0.0 | 0.00100 | 0.00105 | 0.0 | 0.00060 |

Таблица Б2 – время работы модулей NLTK для токенизации на 3-граммы (для Таблицы 6 основной части документа)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Инструменты | Измерения времени, с | | | | | Среднее время работы инструмента, с |
| 1 | 2 | 3 | 4 | 5 |
| Trigram | 0.0 | 0.00105 | 0.00053 | 0.00100 | 0.0 | 0.00052 |
| Ngrams | 0.00036 | 0.00080 | 0.0 | 0.00054 | 0.00105 | 0.00055 |

Тестирование времени проводилось на предложении «Пушкин неоднократно писал о своей родословной в стихах и прозе; он видел в своих предках образец истинной «аристократии», древнего рода, честно служившего отечеству, но не снискавшего благосклонности правителей и «гонимого».».

Приложение В

Таблица В1 – время работы инструментов для токенизации на 2-граммы (для Таблицы 5 основной части документа)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Инструменты | Измерения времени, с | | | | | Среднее время работы инструмента, с |
| 1 | 2 | 3 | 4 | 5 |
| SpaCy | 0.00100 | 0.00100 | 0.0 | 0.00100 | 0.0 | 0.00060 |
| Pattern | 2.33088 | 2.27264 | 2.26600 | 2.24720 | 2.54472 | 2.33229 |
| Scikit-learn | 0.00100 | 0.00102 | 0.00101 | 0.00102 | 0.00100 | 0.00101 |
| TextBlob | 0.00925 | 0.01004 | 0.00901 | 0.01004 | 0.00997 | 0.00966 |

Таблица В2 – время работы инструментов для токенизации на 3-граммы (для Таблицы 5 основной части документа)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Инструменты | Измерения времени, с | | | | | Среднее время работы инструмента, с |
| 1 | 2 | 3 | 4 | 5 |
| SpaCy | 0.0 | 0.00100 | 0.0 | 0.00117 | 0.0 | 0.00043 |
| Pattern | 2.44092 | 2.43213 | 2.56806 | 2.42952 | 2.47671 | 2.46947 |
| Scikit-learn | 0.00184 | 0.00165 | 0.00195 | 0.00184 | 0.00159 | 0.00177 |
| TextBlob | 0.00948 | 0.00933 | 0.00917 | 0.00899 | 0.00777 | 0.00895 |

Тестирование времени проводилось на предложении «Пушкин неоднократно писал о своей родословной в стихах и прозе; он видел в своих предках образец истинной «аристократии», древнего рода, честно служившего отечеству, но не снискавшего благосклонности правителей и «гонимого».».

Приложение Г

Таблица Г1 – время работы модулей NLTK для токенизации на 1-2-3-граммы (для Таблицы 7 основной части документа)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Инструменты | Измерения времени, с | | | | | Среднее время работы инструмента, с |
| 1 | 2 | 3 | 4 | 5 |
| Unigram + Bigram + Trigram | 0.00100 | 0.00093 | 0.00105 | 0.0 | 0.00104 | 0.00080 |
| Ngrams | 0.00066 | 0.00200 | 0.00028 | 0.00078 | 0.00095 | 0.00093 |
| Everygrams | 0.00100 | 0.00101 | 0.00096 | 0.00100 | 0.0 | 0.00079 |

Таблица Г2 – время работы инструментов для токенизации на 1-2-3-граммы (для Таблицы 8 основной части документа)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Инструменты | Измерения времени, с | | | | | Среднее время работы инструмента, с |
| 1 | 2 | 3 | 4 | 5 |
| SpaCy | 0.0 | 0.00105 | 0.00100 | 0.00100 | 0.00100 | 0.00081 |
| Pattern | 2.45792 | 2.42852 | 2.48038 | 2.46185 | 2.44867 | 2.455468 |
| Scikit-learn | 0.00300 | 0.00199 | 0.00100 | 0.00186 | 0.00105 | 0.00178 |
| TextBlob | 0.01099 | 0.01097 | 0.01204 | 0.00996 | 0.00998 | 0.010788 |

Тестирование времени проводилось на предложении «Пушкин неоднократно писал о своей родословной в стихах и прозе; он видел в своих предках образец истинной «аристократии», древнего рода, честно служившего отечеству, но не снискавшего благосклонности правителей и «гонимого».».