**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., доцент |  |  | Белаш О.Ю. |
|  |  | *подпись* |  |  |
|  | ассистент |  |  | Ряскова Е.Б. |
|  |  | *подпись* |  |  |
|  | к.т.н., доцент |  |  | Ичкитидзе Ю.Р. |
|  |  | *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, Составление бизнес-плана по коммерциализации результатов НИР магистранта, Заключение. | | | | | |
| Перечень отчетных материалов: пояснительная записка, иллюстративный материал. | | | | | |
| Дополнительные разделы: Составление бизнес-плана по коммерциализации результатов НИР магистранта. | | | | | |
|  | | | | | |
| 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](#_Toc103433134)

[**введение** 9](#_Toc103433135)

[**1.** **Обзор предметной области** 11](#_Toc103433136)

[**1.1.** **Классификация и категоризация текстов на естественном языке** 11](#_Toc103433137)

[**1.1.1.** **Этапы процесса категоризации текстов** 14](#_Toc103433138)

[**1.1.2.** **Алгоритм декомпозиции и алгоритм выделения ключевых слов. Ключевые слова** 18](#_Toc103433139)

[**1.1.3.** **Ключевые элементы. Понятие расширения ключевого элемента** 19](#_Toc103433140)

[**1.2.** **Семантическая декомпозиция как один из инструментов семантического анализа текстов** 19](#_Toc103433141)

[**1.3.** **Сравнительный анализ существующих алгоритмов семантической декомпозиции и выбор базовых алгоритмов для реализации** 21](#_Toc103433142)

[**1.4.** **Выводы** 42](#_Toc103433143)

[**2.** **ПОСТАНОВКА ЗАДАЧИ** 43](#_Toc103433144)

[**3.** **Описание Теоретических основ разрабатываемого алгоритма** 45](#_Toc103433145)

[**3.1.** **Графическое описание алгоритма семантической декомпозиции** 45](#_Toc103433146)

[**3.2.** **Лингвистические основы алгоритма семантической декомпозиции** 46](#_Toc103433147)

[**3.2.1.** **Основные процедуры алгоритма** 46](#_Toc103433148)

[**3.2.2.** **Дополнительные процедуры алгоритма** 46](#_Toc103433149)

[**3.3.** **Математические основы алгоритма семантической декомпозиции** 47](#_Toc103433150)

[**3.4.** **Выводы** 48](#_Toc103433151)

[**4.** **ПРОГРАММНАЯ РЕАЛИЗАЦИЯ АЛГОРИТМОВ** 49](#_Toc103433152)

[**4.1.** **Технические подробности программной реализации алгоритма** 49](#_Toc103433153)

[**4.1.1.** **SpaCy** 49](#_Toc103433154)

[**4.1.2.** **Textacy** 50](#_Toc103433155)

[**4.1.3.** **DisplaCy** 51](#_Toc103433156)

[**4.1.4.** **Tkinter** 51](#_Toc103433157)

[**4.1.5.** **Time** 51](#_Toc103433158)

[**4.2.** **Сценарий использования** 52](#_Toc103433159)

[**4.3.** **Тестирование алгоритма** 53](#_Toc103433160)

[**4.4.** **Применение алгоритма для решения задачи категоризации** 70](#_Toc103433161)

[**4.4.1.** **Общий вид внедрения алгоритма декомпозиции в систему категоризации** 70](#_Toc103433162)

[**4.4.2.** **Частный вид внедрения алгоритма декомпозиции в систему категоризации** 70](#_Toc103433163)

[**4.5.** **Выводы** 70](#_Toc103433164)

[**5.** **исследование** 71](#_Toc103433165)

[**5.1.** **Исследование времени работы** 71](#_Toc103433166)

[**5.1.1.** **Сценарий эксперимента** 71](#_Toc103433167)

[**5.1.2.** **Методика измерения** 71](#_Toc103433168)

[**5.1.3.** **Используемые программные средства** 71](#_Toc103433169)

[**5.1.4.** **Результаты эксперимента** 72](#_Toc103433170)

[**5.1.5.** **Сравнение и оценка результатов** 72](#_Toc103433171)

[**5.2.** **Исследование точности результатов** 73](#_Toc103433172)

[**5.2.1.** **Сценарий эксперимента** 73](#_Toc103433173)

[**5.2.2.** **Входные данные** 73](#_Toc103433174)

[**5.2.3.** **Результат работы алгоритмов** 73](#_Toc103433175)

[**5.2.4.** **Экспертная оценка** 74](#_Toc103433176)

[**5.2.5.** **Сравнение с экспертной оценкой** 74](#_Toc103433177)

[**5.3.** **Выводы** 75](#_Toc103433178)

[**6.** **Технико-экономическое обоснование научно-технического проекта разработки алгоритма семантической декомпозиции текста для задачи категоризации** 76](#_Toc103433179)

[**6.1.** **Описание проекта** 76](#_Toc103433180)

[**6.1.1.** **Резюме** 76](#_Toc103433181)

[**6.1.2.** **Описание продукции** 76](#_Toc103433182)

[**6.1.3.** **Анализ рынка сбыта** 76](#_Toc103433183)

[**6.1.4.** **Анализ конкурентов** 76](#_Toc103433184)

[**6.2.** **План маркетинга** 76](#_Toc103433185)

[**6.2.1.** **План продаж** 76](#_Toc103433186)

[**6.2.2.** **товарная политика** 76](#_Toc103433187)

[**6.2.3.** **ценовая политика** 76](#_Toc103433188)

[**6.2.4.** **сбытовая политика и мероприятия** 76](#_Toc103433189)

[**6.3.** **План производства** 76](#_Toc103433190)

[**6.4.** **финансовый план** 76](#_Toc103433191)

[**заключение** 78](#_Toc103433192)

[**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ** 80](#_Toc103433193)

# **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.

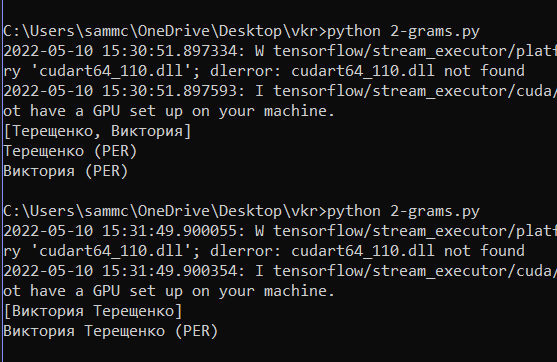
Алгоритм должен решать следующие проблемы:

* 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. 2), можно понять, что алгоритм семантической декомпозиции будет находиться в самом его начале, среди процедур предобработки текста.

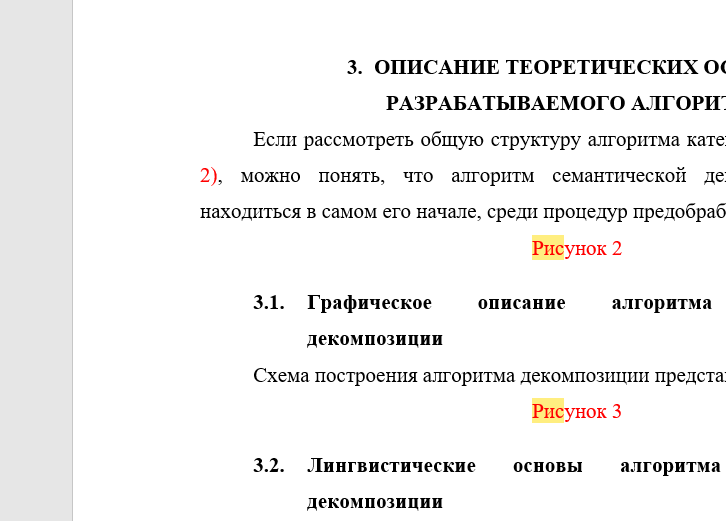


Рисунок 2 – Общая структура алгоритма категоризации

(рисунок нарисовать, сейчас стоит заглушка!)

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

Схема построения алгоритма декомпозиции представлена на рис. 3

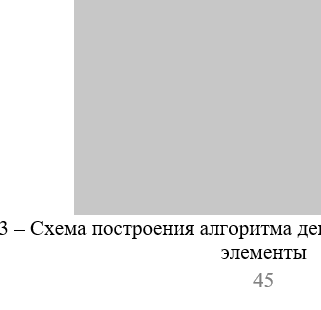


Рисунок 3 – Схема построения алгоритма декомпозиции текста на ключевые элементы

(рисунок нарисовать, сейчас стоит заглушка!)

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

Процедуры основной части алгоритма декомпозиции (см. рис. 3):

* 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 цифр) или с выбором критериев:

* декомпозиция без фильтрации;
* декомпозиция с фильтрацией стоп-слов и 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:

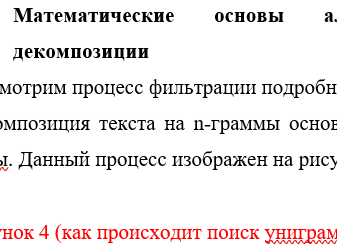


Рисунок 4 – Схематическое описание логики поиска униграмм, биграмм, триграмм

Рисунок 4 (как происходит поиск униграмм, биграмм, триграмм)

На рисунке 4 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**

Данный инструмент, SpaCy, выбранный в главе 1, был использован в качестве основы для разработки алгоритма декомпозиции текста на ключевые элементы.

SpaCy — это библиотека для расширенной обработки естественного языка в Python и Cython. Она основана на самых последних исследованиях и с самого начала разрабатывалась для использования в реальных продуктах.

Поставляется с предварительно обученными языковыми моделями и в настоящее время поддерживает токенизацию и обучение для более чем 60 языков. Она включает в себя самые современные модели для тегирования частей речи, сегментации предложений, синтаксического анализа, распознавания именованных объектов, классификации текста, многозадачного обучения с предварительно обученными инструментами, а также готовую к использованию на производстве систему обучения. Легко расширяется с помощью настраиваемых компонентов и атрибутов. Поддерживает пользовательские модели в PyTorch, TensorFlow и других фреймворках.

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

* + - 1. **Модель Russian**

Наиболее подходящая модель SpaCy для русского языка по умолчанию. С помощью нее проходило тестирование SpaCy в первой главе. Однако ее функционала недостаточно для применения ее в разработке данного алгоритма, а именно, для его дополнений. https://translated.turbopages.org/proxy\_u/en-ru.ru.16175f5e-627e9b2f-7ab37deb-74722d776562/https/www.tutorialspoint.com/spacy/spacy\_models\_and\_languages.htm

* + - 1. **Модель Ru2**

Модель для русского языка SpaCy, которая разрабатывалась тогда, когда официальных расширенных SpaCy моделей еще не было в общем доступе. Однако она давно не обновляется и не поддерживает последние версии Python и некоторых других сопутствующих пакетов. Также сейчас уже можно найти официальные модели для обработки русского языка широкого спектра действий. https://github.com/buriy/spacy-ru

* + - 1. **Модель Ru\_core\_news\_sm**

Официальная модель для русского языка SpaCy. Поддерживает широкий функционал, имеет инструменты для переобучения и для создания своих методов обработки текста. Содержит все необходимые компоненты для реализации алгоритма декомпозиции текста на ключевые элементы. В связи с этим, именно она была выбрана для использования в разработке. <https://spacy.io/models/ru>

* + 1. **Textacy**

Библиотека Textacy была использована непосредственно для задач токенизации и поиска n-грамм в тексте.

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.

Textacy работает с наборами данных как с текстовым содержимым, так и с метаданными. С ее помощью исследуемый текст очищается и нормализуется перед его обработкой с помощью spaCy, извлекается структурированная информация из обработанных документов, включая n-граммы, сущности, аббревиатуры, ключевые термины и триплеты SVO, сравниваются строки и последовательности, используя различные показатели сходства, токенизируются и векторизируются документы для дальнейшего обучения, интерпретации и визуализации тематических моделей, вычисляется статистика удобочитаемости текста по различным метрикам.

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

* + 1. **DisplaCy**

DisplaCy – визуализатор НЛП. В разработке он был использован 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. **Tkinter**

Tkinter – кроссплатформенный GUI фреймворк Python. С его помощью был написан пользовательский интерфейс для более удобного использования и тестирования алгоритма. https://docs.python.org/3/library/tkinter.html

* + 1. **Time**

Данный модуль предоставляет различные функции для работы со временем. С его помощью проводились тестирования времени работы инструментов токенизации в 1-ой главе, а также проводилось тестирование времени работы алгоритма в главе 5. https://docs.python.org/3/library/time.html

* 1. **Сценарий использования**

После запуска файла с пользовательским интерфейсом алгоритма работа с ним может проводиться по следующему сценарию (см. рис. 5):

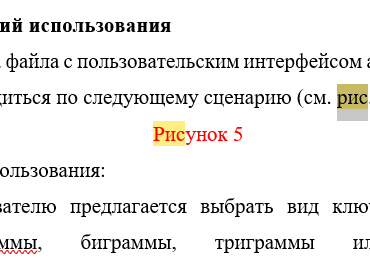


Рисунок 5 – Блок схема "Сценарий использования"

Сценарий использования:

1. Пользователю предлагается выбрать вид ключевого элемента (униграммы, биграммы, триграммы или смешанная декомпозиция). Пользователь выбирает нужный ему вариант и нажимает кнопку «Выбрать».
2. Далее Пользователю предлагается выполнить «быструю» декомпозицию по умолчанию или декомпозицию с выбором параметров. Декомпозиция по умолчанию – с фильтрацией стоп-слов, цифр и именованных сущностей. Пользователь выбирает нужный ему вариант и нажимает кнопку «Выбрать».
   1. При выборе варианта быстрой декомпозиции Пользователь получает выходные данные алгоритма.
   2. Иначе продолжается работа с пользовательским интерфейсом по выбору параметров декомпозиции.
3. Следующий вопрос Пользователю: нужен ли ему отбор частей речи. Под отбором частей речи по умолчанию понимается отбор существительных и прилагательных (значения частей речи можно поменять в исходном коде в файле tok.py в функциях tokenize\_checkPOS() и tokenize\_filter\_checkPOS()). Пользователь выбирает нужный ему вариант и нажимает кнопку «Выбрать».
4. Далее Пользователь отвечает на вопрос, нужна ли ему фильтрация стоп-слов и цифр, как элементов, не несущих смысла. Пользователь выбирает нужный ему вариант и нажимает кнопку «Выбрать».
   1. В случае положительно ответа в п.3 Пользователь получает выходные данные алгоритма.
   2. В случает отрицательного ответа в п.3 продолжается работа с пользовательским интерфейсом по выбору оставшихся параметров декомпозиции.
5. Последний вопрос, который поступает Пользователю: нужна ли ему фильтрация именных сущностей. Именные сущности – имена собственные (имена, фамилии, названия городов, стран, предприятий и так далее). Пользователь выбирает нужный ему вариант и нажимает кнопку «Выбрать».
6. Пользователь получает выходные данные алгоритма.

На любом этапе взаимодействия с пользовательским интерфейсом можно изменить выбор в уже отмеченных вопросах и алгоритм подстроится под изменения.

* 1. **Тестирование алгоритма**

Ниже проводится тестирование работоспособности алгоритма и проводятся действия по шагам из сценария использования (см п. 4.2).

Тестирование проводится на тексте: «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». (1999)».

Шаги тестирования:

* + - 1. Запуск пользовательского интерфейса (см. рис. 6)



Рисунок 6 – Запуск пользовательского интерфейса

Первоначальный вид пользовательского интерфейса (см. рис. 7)

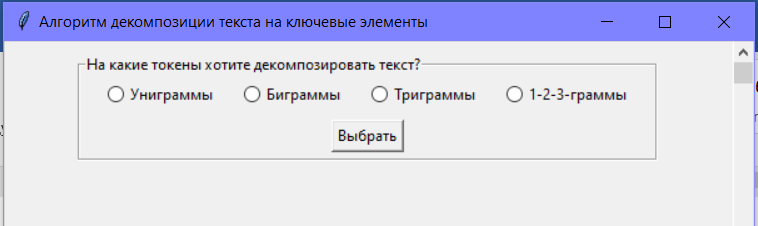


Рисунок 7 – Первоначальный вид пользовательского интерфейса

1. Выбор вида ключевого элемента (см. рис. 8)

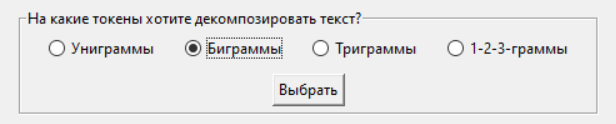


Рисунок 8 – Выбор вида ключевого элемента

1. После нажатия кнопки «Выбрать» на рис. 8 на экране появляется следующий вопрос. (см. рис. 9) Выбор сделан в пользу «Декомпозиции по умолчанию».

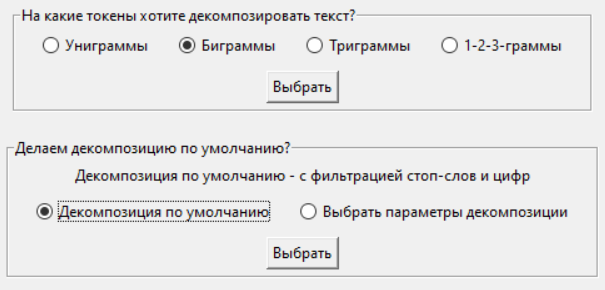


Рисунок 9 – Выбор быстрой или выборочной декомпозиции

1. После нажатия кнопки «Выбрать» на рис. 9 на экране появляется окно с выходными данными (см. рис. 10)

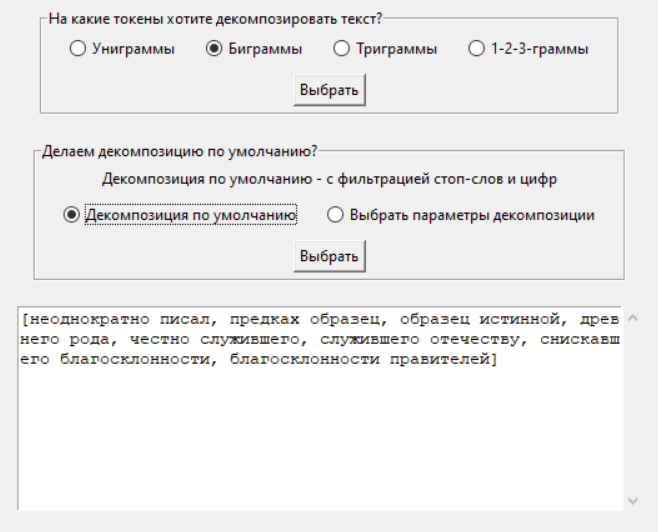


Рисунок 10 – Выходные данные при выборе "Декомпозиции по умолчанию"

1. При выборе смешанной декомпозиции на данном этапе, то есть при изменении ответа на первый вопрос «На какие токены хотите декомпозировать текст» на «1-2-3-граммы» (см. рис. 8), и нажатии кнопки «Выбрать» для декомпозиции по умолчанию текст вывода алгоритма сменится в соответствии с изменением выбора в первом окне (см. рис. 11). Обратите внимание, что в смешанной декомпозиции результатом является список, состоящий из униграмм, биграмм и триграмм одновременно.

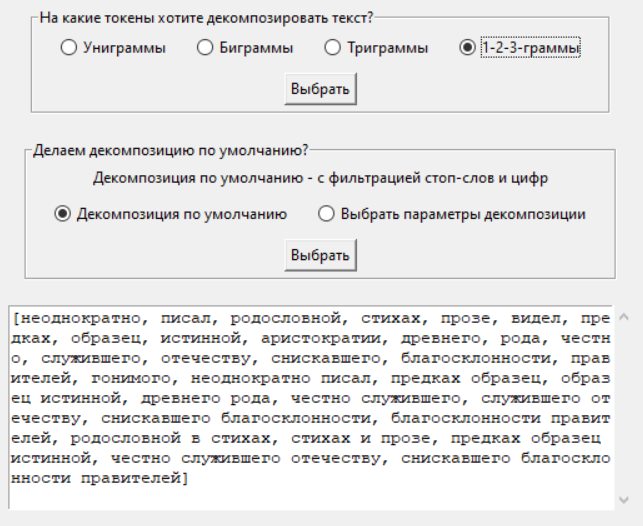


Рисунок 11 – Выходные данные алгоритма при декомпозиции по умолчанию для 1-2-3-граммов

1. Далее вид токенов (см. рис. 8) сменился на «униграммы», а ответ на вопрос «Делаем декомпозицию по умолчанию?» (см. рис. 9) был изменен на «Выбрать параметры декомпозиции», во втором окне нажата кнопка «Выбрать» (см. рис. 12)

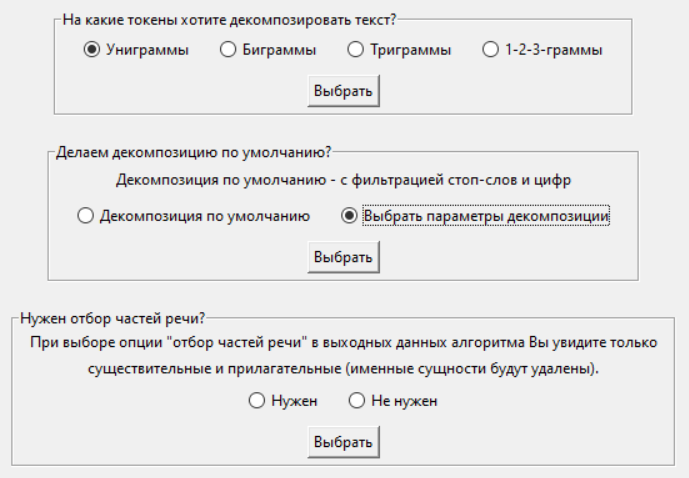


Рисунок 12 – Выбор в пользу "Выбрать параметры декомпозиции"

Появилось третье окно с выбором необходимости отбора частей речи. При выборе отбора частей речи на выходе алгоритма будут присутствовать только «нужные» части речи. В данном случае по умолчанию стоят существительные и прилагательные.

Выбор в третьем окне был сделан в пользу «Нужен» (см. рис. 13)

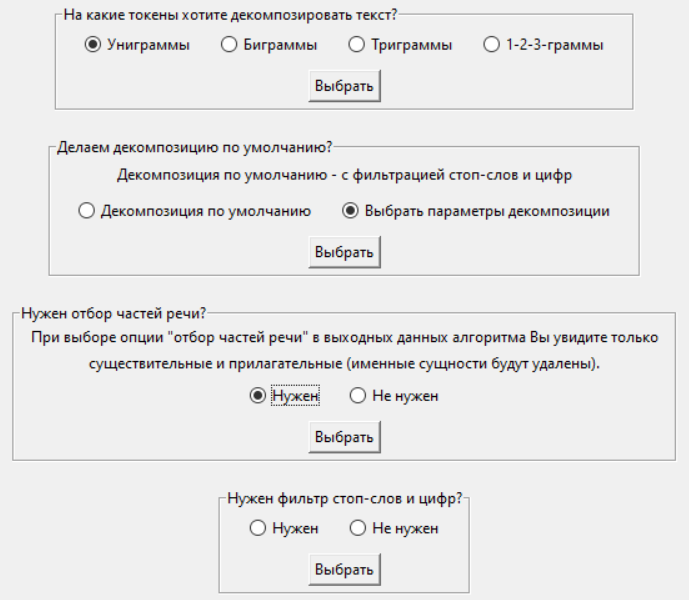


Рисунок 13 – Выбор в пользу "Нужен" в контексте отбора частей речи

1. Появилось четвертое окно выбора «Нужен фильтр стоп-слов и цифр?». Данный фильтр убирает из выходных данных слова, не несущие особого смысла. Необходимость данной фильтрации рассмотрена в гл.3.3.

Выбор в четвертом окне сделан в пользу «Нужен» (см. рис. 14)

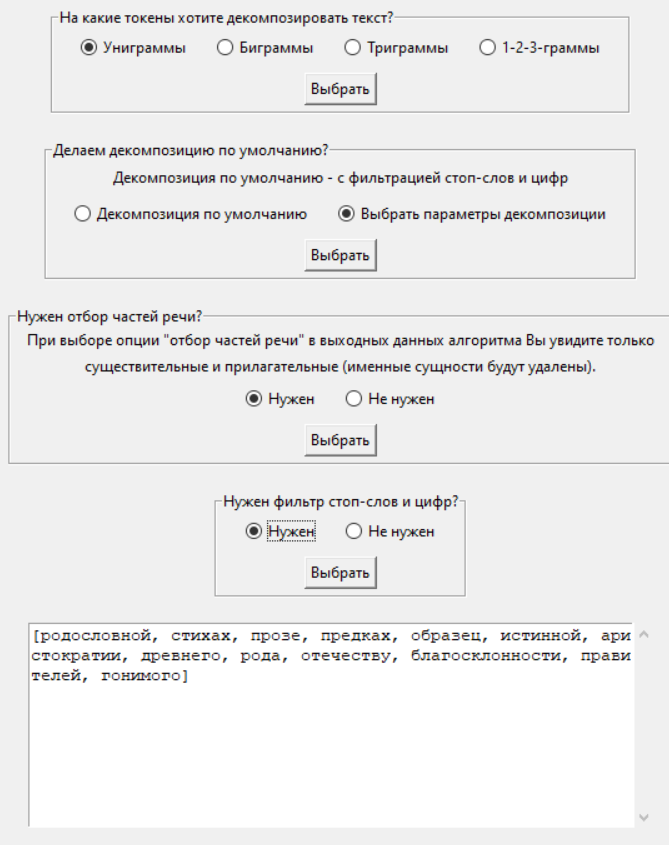


Рисунок 14 – Выбор в пользу "Нужен" в контексте фильтрации стоп-слов и цифр

На рис. 14 изображен вывод программы декомпозиции на униграммы с фильтрацией стоп-слов и цифр, с отбором существительных и прилагательных.

1. При выборе в четвертом окне «Не нужен» программа выдает следующий результат (см. рис. 15)

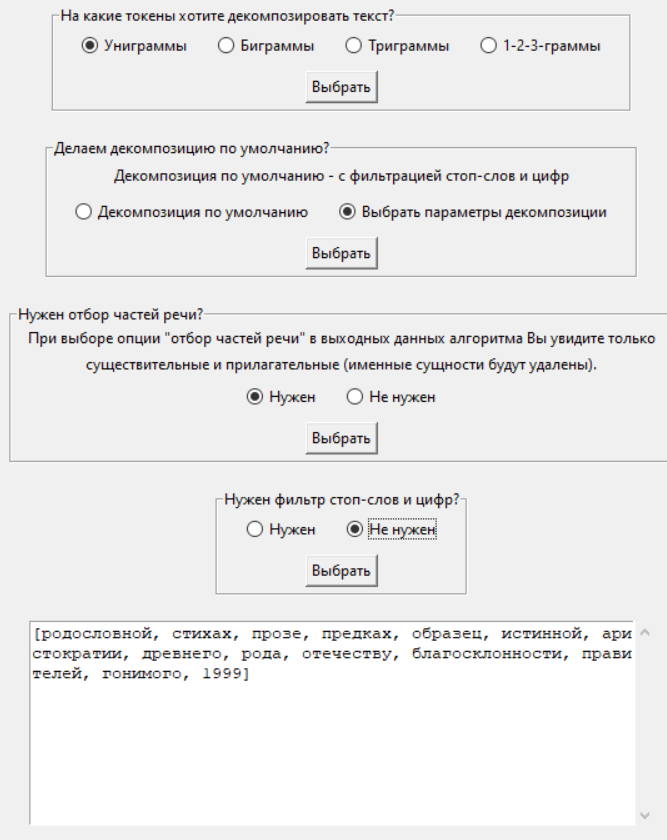


Рисунок 15 – Выбор в пользу "Не нужен" в контексте фильтрации стоп-слов и цифр

На рис. 15 изображен вывод результата декомпозиции на униграммы без фильтрации стоп-слов и цифр, но с отбором существительных и прилагательных.

1. Если вернуться в п.5 (см. рис. 13) и в третьем окне отказаться в необходимости отбора частей речи, то также появится окно с вопросов про фильтрацию стоп-слов и цифр (см. рис. 16)

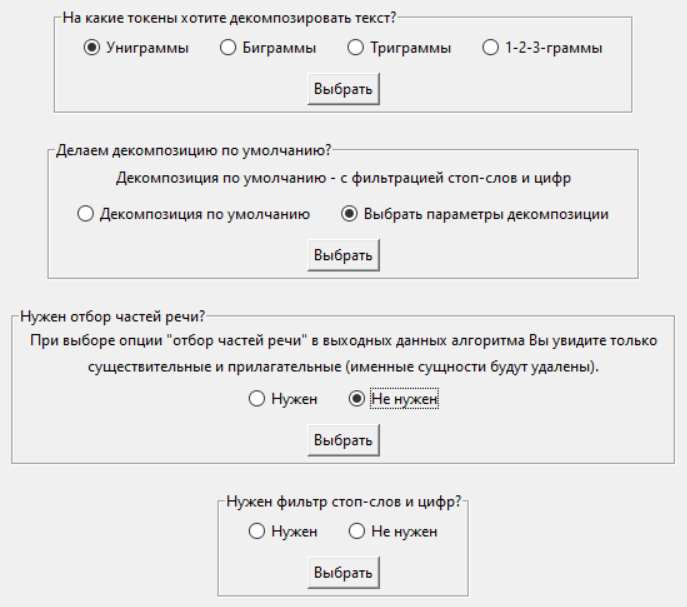


Рисунок 16 – Выбор в пользу "Не нужен" в контексте отбора частей речи

1. При выборе «Нужен» в отношении фильтрации стоп-слов и цифр, появится следующее окно выбора (см. рис. 17)

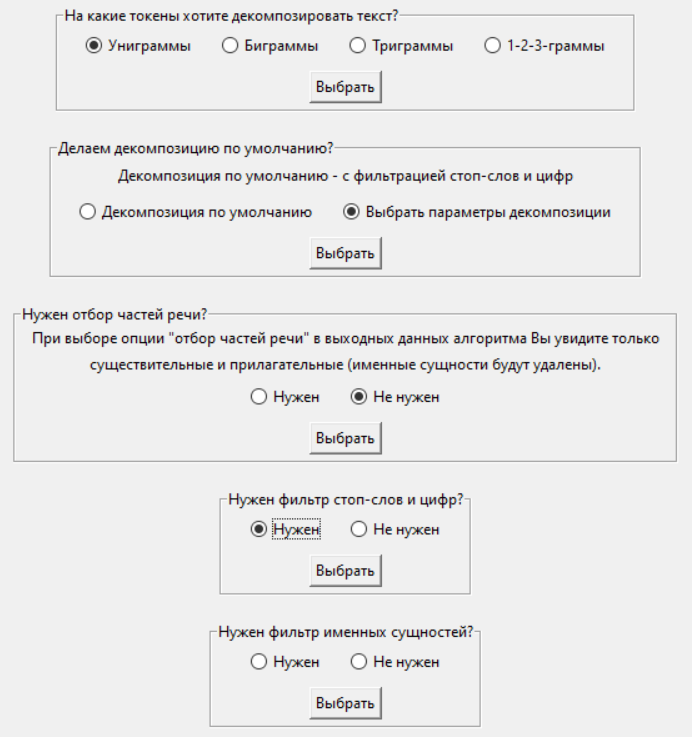


Рисунок 17 – Выбор в пользу "Нужен" в контексте фильтрации стоп-слов и цифр при отказе от отбора частей речи

1. Фильтр именных сущностей очищает выходные данные от имен собственных, фамилий, названий компаний, которые могут часто появляться в заявках службы поддержки. Выбор в пятом окне в пользу «Не нужен» покажет список токенов, содержащий именованные сущности (см. рис. 18)

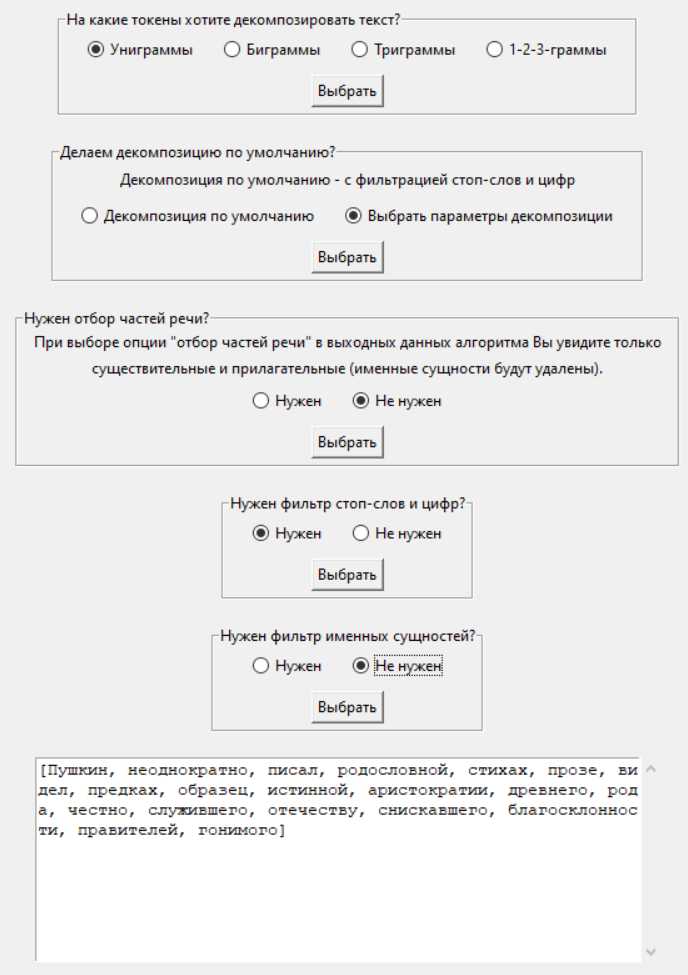


Рисунок 18 – Выбор в пользу "Не нужен" в контексте фильтрации именованных сущностей при наличии фильтра стоп-слов и цифр

На рис. 18 изображен вывод результата декомпозиции на униграммы с фильтрацией стоп-слов и цифр, но без фильтрации именованных сущностей.

В данном случае именованной сущностью является фамилия писателя «Пушкин».

1. Проверкой этого будет выполнение фильтрации именованных сущностей. Выбор в пятом окне в пользу «Нужен» (см. рис. 19)

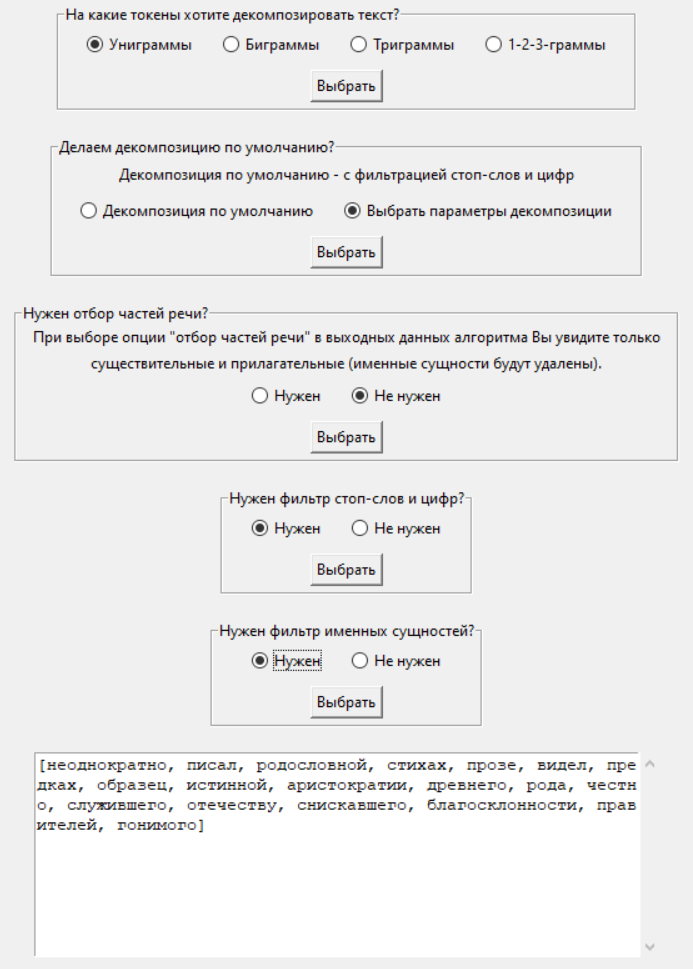


Рисунок 19 – Выбор в пользу "Нужен" в контексте фильтрации именованных сущностей при наличии фильтра стоп-слов и цифр

На рис. 19 изображен вывод результата декомпозиции на униграммы с фильтрацией стоп-слов и цифр, с фильтрацией именованных сущностей. Как видно, «Пушкин» уже не присутствует в выходных данных, что означает успешную фильтрацию именованных сущностей.

1. Если вернуться в п.9 и в отношении фильтрации стоп-слов и цифр выбрать «Не нужен», далее последует тот же вопрос о фильтрации именованных сущностей (см. рис. 20).

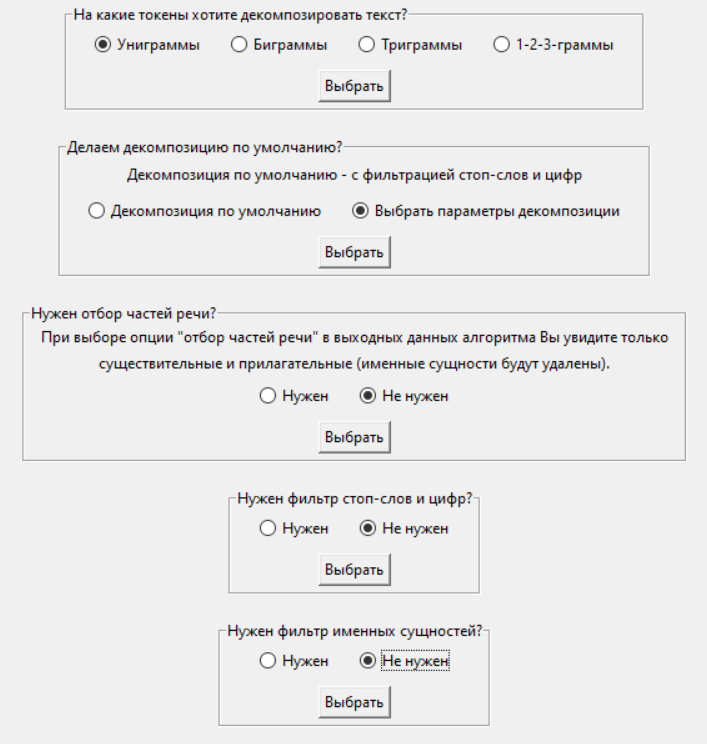


Рисунок 20 – Выбор в пользу "Не нужен" в контексте фильтрации стоп-слов и цифр при отказе от отбора частей речи

1. Выбор в пользу «Не нужен» в контексте фильтрации именных сущностей (см. рис. 20) в данной ситуации означает отказ от любой фильтрации и вывод полного списка токенов (см. рис. 21)

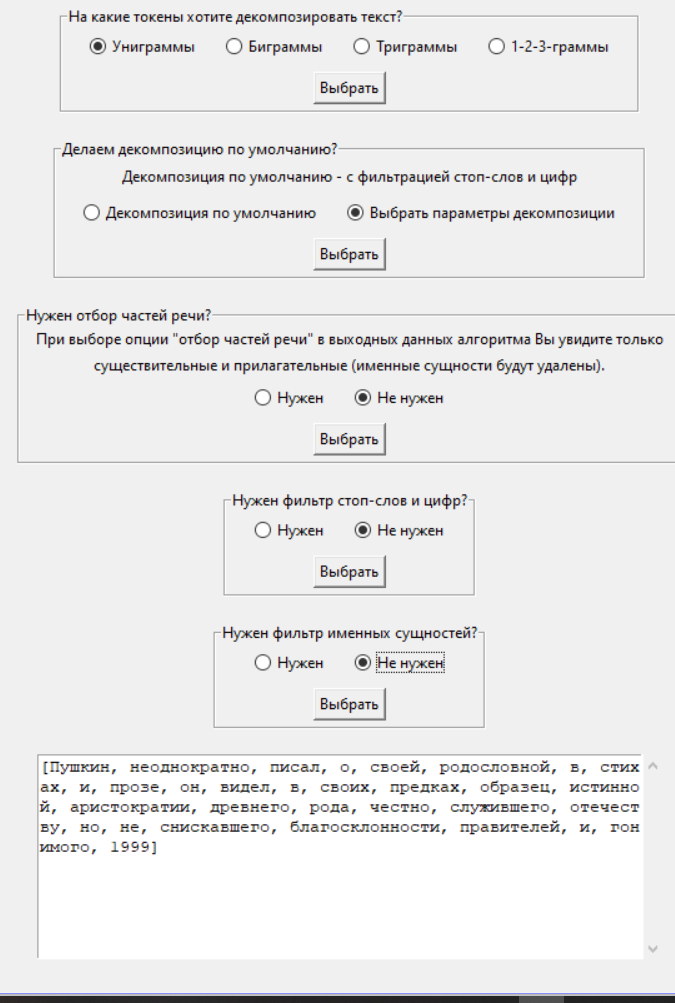


Рисунок 21 – Выбор в пользу "Не нужен" в контексте фильтрации именованных сущностей при отказе от фильтрации стоп-слов и цифр и от отбора частей речи

На рис. 21 изображен вывод результата декомпозиции на униграммы без фильтрации стоп-слов и цифр, без фильтрации именованных сущностей, без отбора частей речи. В данном случае можно прочитать целое предложение, поступившее на вход алгоритму, что говорит о том, что ни одно слово не было изъято их входных данных.

1. Оставшийся непроверенный выбор – в пользу «Нужен» в контексте фильтрации именованных сущностей (см. рис. 22)

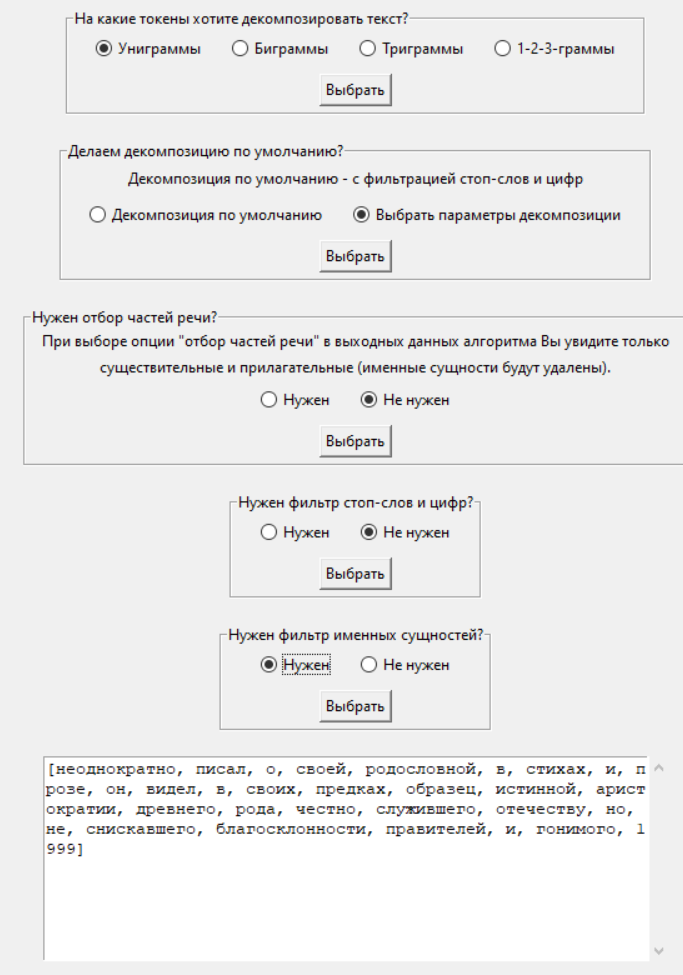
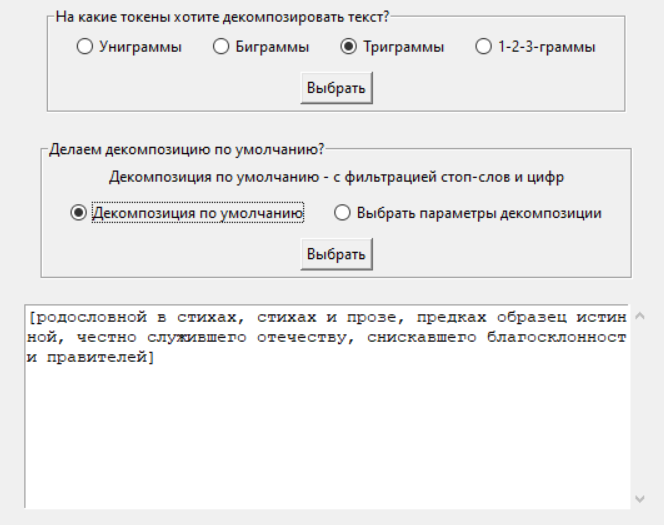


Рисунок 22 – Выбор в пользу "Нужен" в контексте фильтрации именованных сущностей при отказе от фильтрации стоп-слов и цифр и от отбора частей речи

На рис. 22 изображен вывод результата декомпозиции на униграммы без фильтрации стоп-слов и цифр, с фильтрацией именованных сущностей, без отбора частей речи. Здесь, как и в п.11, было отфильтровано слово «Пушкин».

1. Из неопробованных остался только вариант триграмм в первом окне выбора. Ниже выполнено тестирование для декомпозиции на триграммы по умолчанию (см. рис. 23)

****

На рис. 23 изображен вывод результата декомпозиции на триграммы по умолчанию (с фильтрации стоп-слов, цифр и именованных сущностей).

Таким образом, были протестированы все варианты комбинаций фильтраций для декомпозиции текста на ключевые элементы, приведен пример выходных данных для каждого вида декомпозиции, доказано, что программа может на любом этапе менять значения фильтрации для тех окон выбора, где ранее уже был дан ответ, и проводить фильтрацию снова.

Сеанс работы с программой заканчивается при нажатии на «крестик» в правом верхнем углу окна пользовательского интерфейса.

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

Внедрение алгоритма в общую систему категоризации может происходить путем подключения кода алгоритма к коду системы.

* + 1. **Общий вид внедрения алгоритма декомпозиции в систему категоризации**

Текст, поступающий на вход системе, проходит через алгоритм декомпозиции в самом начале работы, так как он должен пройти этап предобработки, в которую входит алгоритм семантической декомпозиции. Далее выходные данные алгоритма декомпозиции подаются на вход алгоритму выделения ключевых слов (или аналогичному алгоритму), где токенам присваиваются веса и на их основе происходит разметка данных для решения задачи категоризации.

* + 1. **Частный вид внедрения алгоритма декомпозиции в систему категоризации**

Программный вариант внедрения алгоритма в код системы категоризации (СППР для службы технической поддержки компании Газпром переработка) может выглядеть следующим образом. В файле text\_processing.py код замещается на представленный ниже:

import tok

import functional

import spacy

class TextProcessing:

def preprocess(self, raw\_text):

nlp = spacy.load('ru\_core\_news\_sm')

doc = nlp(raw\_text)

n = 4

result\_tok = tok.tokenize\_filter\_ents(doc, n)

result\_tok = functional.get\_lemma(result\_tok, n)

result\_tok = functional.remove\_spec\_stop\_words(result\_tok, n)

return result\_tok

В данных строках кода подключается модель для обработки русскоязычного текста, формируется документ Spacy из поступившего на вход текста, выставляется значение n, которое характеризует тип ключевого элемента (униграммы, биграммы, триграммы или смешанная токенизация). Далее запускается функция токенизации из модуля tok, в данном случае запущена функция tokenize\_filter\_ents() по умолчанию, ее можно менять на усмотрение разработчика. Затем запускается процесс лемматизации и удаления специализированных стоп-слов. На этом предобработка заканчивается.

Разработчик может подключать и другие предложенные, но незадействованные функции в работу системы. Более подробная информация о них и о работе с ними содержится по ссылке https://github.com/vika210899/master-s-dissertation.

Для работы кода выше в проект системы необходимо добавить файлы functional.py и tok.py (см. Приложения Д и Е), в которых содержатся дополнительный и основной функционал алгоритма (более подробное описание по ссылке https://github.com/vika210899/master-s-dissertation).

* 1. **Выводы**

Результатом разработки является готовая программа, имеющая удобный графический пользовательский интерфейс и выполняющая главную задачу диссертации – семантическую декомпозицию текста на ключевые элементы для решения задачи категоризации.

В данной главе приведено техническое описание разработки алгоритма, приведен сценарий использования программы и дан пример ее использования. Также проведено тестирование алгоритма и программы в целом, приведены варианты использования алгоритма для задачи категоризации.

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

Проведено исследование времени работы алгоритма семантической декомпозиции. Исследование проводилось над каждым токенизатором, входящим в алгоритм:

* токенизатор без фильтрации (tokenize\_only);
* токенизатор с фильтрацией стоп-слов и цифр (tokenize\_filter);
* токенизатор с фильтрацией именных сущностей (tokenize\_ents);
* токенизатор с фильтрацией стоп-слов, цифр и именных сущностей (tokenize\_filter\_ents);
* токенизатор с отбором частей речи (tokenize\_checkPOS);
* токенизатор с фильтрацией стоп-слов и цифр, с отбором частей речи (tokenize\_filter\_checkPOS).

А также над работой токенизатора в совокупности с лемматизатором (lemmatize), а также с функцией удаления специализированных стоп-слов (remove\_spec\_stop\_words).

* + 1. **Сценарий эксперимента**

Измеряемая величина – время (в секундах), затраченное на работу каждого из токенизаторов.

Измерение проводилось на наборе из 35 328 текстов заявок путем фиксирования времени начала и окончания выполнения алгоритма и последующего деления времени работы (разности значений начала и окончания выполнения декомпозиции) на количество заявок для получения времени обработки одной заявки.

Процедура проводилась 50 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 на тестовом наборе данных системы категоризации, находящемся в репозитории проекта https://gitlab.com/Merlinkoss/neurogp/-/tree/master.

* + 1. **Используемые программные средства**

Эксперимент проводился с использованием средств Python, а именно с помощью библиотеки time и ее метода time() [https://docs-python.ru/standart-library/modul-time-python/funktsija-time-modulja-time/].

* + 1. **Результаты эксперимента**

Отдельно было рассчитано время загрузки модели для обработки руского языка, так как она будет загружаться в систему единожды, следовательно, незачем включать время ее загрузки в общее время работы алгоритма. Среднее время загрузки модели для обработки русского языка составляет 1.881943 с. Для расчета среднего значения измерения проводились 100 раз.

Так, общим временем работы алгоритма будет считаться время формирование doc совместно со временем проведения необходимых процедур (токенизации, лемматизации и удаления специализированных стоп-слов) в разных комбинациях.

Также отдельно было рассчитано среднее время формирования doc (центральной структуры данных в SpaCy). Время формирования doc зависит от длины текста, подаваемое на вход. Среднее время формирования doc для первых 100 заявок из предоставленного файла для тестирования составляет 0.018825 с. Данный расчет сделан для понимания времени работы оставшейся части алгоритма (токенизации, лемматизации и удаления специализированных стоп-слов), так как формирование doc занимает большую часть всего времени работы алгоритма. Данные, использовавшиеся для расчета частично представлены в Приложение Ж, (время формирования doc посчитано для tokenize\_only). Там же можно сравнить общее время работы алгоритма и время для формирования doc.

Было рассчитано минимальное, максимальное и среднее время работы каждого токенизатора, а также совокупной работы токенизатора с лемматизатором и с функцией удаления специализированных стоп-слов. Измерения проводились для первых 100 заявок из предоставленного файла для тестирования. Полученные данные, приведенные в секундах с точностью до тысячных, представлены в табл. 2.

Таблица 2 – Время работы алгоритма семантической декомпозиции

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № | Алгоритм | Тип ключевого элемента | | | | | | | | | | | |
| Униграммы | | | Биграммы | | | Триграммы | | | 1-2-3-граммы | | |
| Время работы, с | | | | | | | | | | | |
| min | avg | max | min | avg | max | min | avg | max | min | avg | max |
| 1. | tokenize\_only | 0.001 | 0.014 | 0.068 | 0.001 | 0.022 | 0.104 | 0.001 | 0.021 | 0.084 | 0.003 | 0.016 | 0.072 |
| tokenize\_only + lemmatize | 0.001 | 0.014 | 0.067 | 0.001 | 0.014 | 0.066 | 0.001 | 0.021 | 0.090 | 0.007 | 0.026 | 0.108 |
| tokenize\_only + lemmatize + remove\_spec\_stop\_words | 0.001 | 0.024 | 0.108 | 0.001 | 0.015 | 0.066 | 0.003 | 0.017 | 0.074 | 0.005 | 0.018 | 0.068 |
| 2. | tokenize\_filter | 0.001 | 0.014 | 0.063 | 0.008 | 0.020 | 0.104 | 0.005 | 0.021 | 0.089 | 0.001 | 0.014 | 0.070 |
| tokenize\_filter + lemmatize | 0.001 | 0.014 | 0.050 | 0.001 | 0.014 | 0.064 | 0.007 | 0.022 | 0.131 | 0.007 | 0.024 | 0.095 |
| tokenize\_filter + lemmatize + remove\_spec\_stop\_words | 0.001 | 0.021 | 0.099 | 0.001 | 0.014 | 0.068 | 0.003 | 0.016 | 0.059 | 0.003 | 0.021 | 0.104 |
| 3. | tokenize\_ents | 0.001 | 0.013 | 0.051 | 0.007 | 0.023 | 0.120 | 0.001 | 0.021 | 0.099 | 0.004 | 0.018 | 0.122 |
| tokenize\_ents + lemmatize | 0.001 | 0.014 | 0.082 | 0.001 | 0.014 | 0.083 | 0.006 | 0.023 | 0.112 | 0.006 | 0.022 | 0.151 |
| tokenize\_ents + lemmatize + remove\_spec\_stop\_words | 0.007 | 0.022 | 0.102 | 0.001 | 0.014 | 0.068 | 0.003 | 0.016 | 0.067 | 0.006 | 0.022 | 0.132 |
| 4. | tokenize\_filter\_ents | 0.001 | 0.014 | 0.068 | 0.001 | 0.021 | 0.099 | 0.002 | 0.023 | 0.098 | 0.003 | 0.015 | 0.069 |
| tokenize\_filter\_ents + lemmatize | 0.001 | 0.013 | 0.067 | 0.001 | 0.014 | 0.084 | 0.001 | 0.014 | 0.069 | 0.001 | 0.018 | 0.081 |
| tokenize\_filter\_ents + lemmatize + remove\_spec\_stop\_words | 0.008 | 0.022 | 0.107 | 0.001 | 0.014 | 0.066 | 0.003 | 0.015 | 0.127 | 0.007 | 0.022 | 0.118 |
| 5. | tokenize\_checkPOS | 0.001 | 0.013 | 0.067 | 0.004 | 0.020 | 0.092 | 0.001 | 0.020 | 0.083 | 0.005 | 0.015 | 0.068 |
| tokenize\_checkPOS + lemmatize | 0.001 | 0.014 | 0.074 | 0.001 | 0.014 | 0.064 | 0.001 | 0.014 | 0.050 | 0.001 | 0.014 | 0.082 |
| tokenize\_checkPOS + lemmatize + remove\_spec\_stop\_words | 0.002 | 0.021 | 0.113 | 0.001 | 0.014 | 0.065 | 0.004 | 0.018 | 0.097 | 0.007 | 0.021 | 0.100 |
| 6. | tokenize\_filter\_checkPOS | 0.001 | 0.014 | 0.066 | 0.007 | 0.020 | 0.095 | 0.002 | 0.017 | 0.068 | 0.006 | 0.015 | 0.059 |
| tokenize\_filter\_checkPOS + lemmatize | 0.003 | 0.016 | 0.170 | 0.001 | 0.014 | 0.066 | 0.001 | 0.013 | 0.065 | 0.001 | 0.014 | 0.082 |
| tokenize\_filter\_checkPOS + lemmatize + remove\_spec\_stop\_words | 0.001 | 0.021 | 0.094 | 0.001 | 0.014 | 0.068 | 0.003 | 0.016 | 0.066 | 0.007 | 0.022 | 0.106 |

Таблицы с полными данными можно найти по ссылке https://github.com/vika210899/master-s-dissertation.

* + 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 качества результатов алгоритма**

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. **СОСТАВЛЕНИЕ БИЗНЕС-ПЛАНА ПО КОММЕРЦИАЛИЗАЦИИ РЕЗУЛЬТАТОВ НИР МАГИСТРАНТА**
   1. **Описание проекта**
      1. **Резюме**

Проект представляет собой программный модуль для осуществления семантической декомпозиции текста для задачи категоризации, состоящий из алгоритма семантической декомпозиции и пользовательского интерфейса. Разработка велась командой из двух разработчиков компании ООО «Газпром переработка» по внутреннему заказу под требования системы категоризации запросов, поступающих в службу технической поддержки. Проект находится на стадии тестирования.

* + 1. **Описание продукции**

Продукт содержит в себе разработанный алгоритм и пользовательский интерфейс, предназначен для встраивания в систему СППР службы технической поддержки компании ООО Газпром переработка для категоризации поступающих заявок.

Ценностью продукта является возможность исключить человеческий фактор из процесса предобработки текста, из процесса формирования корпусов для обучения/переобучения категоризаторов. Для компании Газпром переработка это еще и возможность сокращения времени и трудозатрат на обработку поступающих заявок в службу технической поддержки, а также увеличение точности системы категоризации.

Конкурентные преимущества программного продукта:

* наличие удобного графического пользовательского интерфейса;
* наличие возможности фильтрации текста перед его декомпозицией;
* наличие возможности выбора параметров семантической декомпозиции текста;
* наличие возможности выбора типа ключевого элемента;
* высокая скорость работы алгоритма;
* открытый исходный код.

Целевая аудитория: разработчики систем категоризации текстовых данных, работники службы технической поддержки компании Газпром переработка.

* + 1. **Анализ рынка сбыта**

Программный продукт был разработан для внедрения в СППР для службы технической поддержки компании Газпром переработка с целью увеличения точности категорирования заявок, поступающих в поддержку, а также для исключения человеческого фактора из процесса предобработки текста входящих заявок разработчиками СППР.

Заказчиком является компания ООО “Газпром переработка”. Программный продукт проектировался под особенности уже реализованной СППР для службы технической поддержки.

Целевой аудиторией продукта являются сотрудники службы технической поддержки и разработчики СППР.

* + 1. **Анализ конкурентов**

В качестве конкурентов рассматриваются существующие инструменты декомпозиции, подробный анализ которых представленные в гл. 1.3. Основные конкуренты вынесены в таблицу ниже:

Таблица 9 – Анализ инструментов декомпозиции

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Название | Время работы, с | Лемматизация | Распознавание частей речи | Распознавание именованных сущностей | Распознавание зависимостей слов | Стоимость |
| Nltk | 0.00179 | + | + | + | + | Бесплатно |
| SpaCy | 0.00081 | + | + | + | + | Бесплатно |
| Pattern | 2.45547 | + | + | - | - | Бесплатно |
| Gensim | --- | + | - | - | - | Бесплатно |
| Scikit-learn | 0.00178 | + | - | - | - | Бесплатно |
| TextBlob | 0.01079 | + | + | + | + | Бесплатно |

Для проекта было крайне важно время работы, чтобы система СППР сохранила свою производительность после внедрения алгоритма, и не мало важны остальные аспекты сравнения для реализации различных вариантов фильтрации текста (подробнее см. гл.1.3. и 3).

Из таблицы 9 видно, что в сравнении с аналогами больше преимуществ имеет инструмент SpaCy, который и был взят в основу проекта. SpaCy имеет лучшее время работы, а так же возможность проводить лемматизацию, распознавание частей речи, именных сущностей, ролей и зависимостей слов в тексте.

Алгоритм семантической декомпозиции из SpaCy, взятый за основу проекта, был дополнен сопутствующим функционалом, описанным в гл. 6.1.2, а также для более удобного использования в проекте появился графический пользовательский интерфейс. Программный продукт был усовершенствован путем проведения ряда тестирований на специализированных текстах, используемых в системе категоризации службы технической поддержки ООО «Газпром переработка». Все перечисленные шаги делают продукт более конкурентноспособным, наряду со стандартными инструментами семантической декомпозиции.

* 1. **План маркетинга**

Продукт был разработан специально под требования заказчика и в будущем предполагается его использование только внутри компании, поэтому никаких маркетинговых стратегий для него не предусматривается.

* 1. **План производства**

При разработке программного продукта были использованы только продукты с бесплатной лицензией, находящиеся в открытом доступе. Закупка сырья для разработки не требуется.

Производство полностью реализуется разработчиками ООО «Газпром переработка».

Все задачи выполняются на личном оборудовании разработчика, поэтому затраты на аппаратное обеспечение равны нулю. Затраты на лицензионное программное обеспечение так же равны нулю, так как вся разработка выполнялась с использованием инструментов с открытым кодом (безоплатные решения).

Таким образом, амортизационные отчисления равны нулю, а затраты на производство сводятся к затратам на оплату труда.

Для реализации программного продукта была задействована команда из двух разработчиков. Первый разработчик осуществлял сбор и анализ требований для разрабатываемого алгоритма и его внедрение в СППР, второй, изучив предметную область и конкурентов, протестировав и проанализировав их, вел разработку алгоритма, пользовательского интерфейса, тестирование программного продукта, исследование потребительских свойств и написание технической документации. Отдельная поддержка программного продукта не предусмотрена, так как по итогу алгоритм будет внедрен в СППР, и поддержка будет вестись для всей системы одновременно.

Этапы и сроки реализации представлены в таблице 10.

Таблица 10 – Этапы и сроки реализации проекта

|  |  |
| --- | --- |
| Наименование этапа | Длительность этапа, чел.-дни |
| Изучение предметной области | 20 |
| Анализ и тестирование конкурентов | 25 |
| Изучение требований заказчика | 5 |
| Проектирование логики алгоритма | 15 |
| Техническая реализация алгоритма | 25 |
| Проектирование пользовательского интерфейса | 7 |
| Тестирование программного продукта | 10 |
| Исследование потребительских свойств алгоритма | 10 |
| Написание технической документации | 3 |
| Проектирование сценария внедрения решения в СППР ООО Газпром переработка | 5 |
| Внедрение решения | 3 |

Исходя из обязанностей разработчиков и информации из таблицы 9, общее количество затраченных на работу дней каждым работником составляет:

* для 1-го работника – 13 отработанных дней;
* для 2-го работника – 115 отработанных дней.

Заработная плата 1-го работника в месяц составляет 15 000 руб., а 2-го – 25 000 руб. Учитывая, что в месяце 21 рабочий день, рассчитана дневная ставка работников:

* для 1-го работника: руб./день;
* для 2-го работника: руб./день.

Расходы на заработную плату работников с учетом отработанных дней:

* для 1-го работника: руб.;
* для 2-го работника: руб.

Отчисления на обязательное социальное, пенсионное и медицинское страхование (общая ставка норматива отчислений на 2022 год составляет 30% http://www.consultant.ru/document/cons\_doc\_LAW\_93256/4b298e40529770eabeae8e0d02c92102992f6acc/#dst102585):

* для 1-го работника: руб.;
* для 2-го работника: руб.

Затраты на оплату труда приведены в таблице 11.

Таблица 11 – Затраты на оплату труда

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Работник | Средняя з/п, руб./мес. | Кол-во рабочих дней | Дневная ставка, руб./день | Затраты на оплату труда, руб. | Отчисления на соц. взносы, руб./мес. |
| Разработчик 1 | 15 000 | 13 | 714,29 | 9 285,77 | 2 785,73 |
| Разработчик 2 | 25 000 | 115 | 1190,48 | 136 905,20 | 41 071,56 |
| Итого, руб. | | | | 146 190,97 | 43 857,29 |

* 1. **финансовый план**

Выручка от реализации проекта рассчитывается исходя из экономии времени сотрудников службы технической поддержки.

За рабочий день один сотрудник технической поддержки обрабатывает в среднем 30 запросов. На обработку (категоризацию) одного запроса работник в среднем тратит 1 минуту. Заявки приходят достаточно часто, и работник ожидает, пока наберется несколько, чтобы обработать их все сразу. Допустим, работник обрабатывает 5 заявок за раз. На накопление 5-ти заявок уходит приблизительно 30 минут. То есть, на один подход для обработки 5 заявок понадобится 30 + 5 = 35 минут. Таких подходов: 30/5 = 6. Значит, всего на обработку заявок в день у одного работника уходит 35\*6 = 210 минут = 3,5 часов. Средняя стоимость заработной платы работника технической поддержки составляет 15 000 руб./мес. (дневная ставка: 714,29 руб./день). Таким образом, стоимость длительной обработки заявок одним работником поддержки в день (рабочий день составляет 8 часов) составляет 714,29 \* 3,5/8 = 312,5 руб.

Из технической документации СППР известно, что округленное среднее значение времени для обработки одной заявкой системой составляет 1 сек. Так, 30 заявок система обработает за 30 секунд. В перерасчете на заработную плату работника поддержки стоимость обработки 30-ти заявок составляет 714,29 \* 30/(60\*60\*8) = 0,74 руб.

Таким образом, выгода использования СППР вместо работника поддержки в день составит: 312,5 – 0,74 = 311,76 руб.

Если за выручку от реализации проекта считать выгоду использования СППР, то выручка за год (в 2022 году их 247 <https://www.garant.ru/calendar/buhpravo/>) для команды технической поддержки (4 человека) составит 247 \* 4 \* 311,76 = 308 018,88 руб. – начиная с 2023 года.

Выручка за 2022 год равна 308 018,88/2 = 154 009,44 руб., так как внедрение будет окончено в июне.

Начальный бюджет на реализацию проекта не выделялся.

Все полученные данные сведены в таблицу 12, где рассчитан план прибыли и убытков.

Таблица 12 – План прибыли и убытков

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Показатели/Период** | **Интервал планирования (годы)** | | | **Всего на 2022** |
| **2022** | **2023** | **2024** |
| 1. Выручка от реализации (Затраты на ведение отчетности вручную) | 154 009,44 | 308 018,88 | 308 018,88 | 770 047,20 |
| 2. Производственные затраты | 146 190,97 | 0 | 0 | 146 190,97 |
| 2.1 Аренда помещения | 0 | 0 | 0 | 0 |
| 2.2 Программное обеспечение | 0 | 0 | 0 | 0 |
| 2.3 Аппаратное обеспечение | 0 | 0 | 0 | 0 |
| 2.4 Офисное оборудование | 0 | 0 | 0 | 0 |
| 2.5 Затраты на оплату труда | 146 190,97 | 0 | 0 | 146 190,97 |
| 3. Валовая прибыль | 7 818,47 | 308 018,88 | 308 018,88 | 623 856,23 |

Ставка дисконтирования (R) = 20%, тогда коэффициент дисконтирования: , где i – номер временного периода.

Таблица 13 – План движения дисконтированных денежных потоков

|  |  |  |  |
| --- | --- | --- | --- |
| **Показатели** | **Год** | | |
| **2022** | **2023** | **2024** |
| 1. Остаток денежных средств на начало периода | 0 | -36 038,82 | 271 980,06 |
| 2. Выручка от реализации (Затраты на ведение отчетности вручную) | 154 009,44 | 308 018,88 | 308 018,88 |
| 3. Выбытие денежных средств (затраты) | 146 190,97 | 0 | 0 |
| 3.1 Аренда помещения | 0 | 0 | 0 |
| 3.2 Программное обеспечение | 0 | 0 | 0 |
| 3.3 Аппаратное обеспечение | 0 | 0 | 0 |
| 3.4 Офисное оборудование | 0 | 0 | 0 |
| 3.5 Затраты на оплату труда | 146 190,97 | 0 | 0 |
| 4. Уплата налогов | 43 857,29 | 0 | 0 |
| 4.1. НДС | 0 | 0 | 0 |
| 4.2. Отчисления на соц. нужды | 43 857,29 | 0 | 0 |
| 5. Всего оттоки (3+4) | 190 048,26 | 0 | 0 |
| 6. Чистый денежный поток (1+2-5) | -36 038,82 | 271 980,06 | 579 998,94 |
| 7. Коэффициент дисконтирования | 0,83 | 0,69 | 0,58 |
| 8. Дисконтированный денежный поток | -30 032,35 | 188 875,04 | 335 647,53 |
| 9. Дисконтированный денежный поток накопленным итогом | -30 032,35 | 158 842,69 | 494 490,22 |

Проанализировав данные таблицы 13, можно сделать вывод, что проект является экономически выгодным, начинает приносить прибыль в 2023 году.

* 1. **Анализ и оценка рисков**

В таблице 14 проанализированы основные риски реализации и эксплуатации проекта и предложены меры по предотвращению и снижению возможных потерь.

Таблица 14 – Основные риски и пути снижения и предотвращения потерь

|  |  |  |
| --- | --- | --- |
| Риск | Последствия | Решение |
| Нарушение совместимости программных инструментов основной системы и внедряемого алгоритма | Сбои при внедрении алгоритма | Перевести реализацию алгоритма на программные инструменты, совместимые с системой |
| Выявление недочетов функции фильтрации алгоритма для специализированных текстов | Увеличение вероятности ухудшения качества работы системы категоризации | Обучить алгоритм семантической декомпозиции текста на корпусе специализированных текстов |
| Выявление недочетов производительности | Увеличение времени обработки заявок системой категоризации может привести к сокращению финансовой выгоды проекта, а также, в худшем случае, к временному возвращению к категоризации заявок работниками поддержки вручную (до момента решения проблемы) | Проанализировать возможные причины сбоя производительности, устранить их |
| Появление новых, не оговоренных ранее, требований к функциональности алгоритма со стороны заказчика | – | Доработать алгоритм согласно обновленным требованиям заказчика |
| Нарушение календарного плана реализации проекта | Нарушение сроков сдачи проекта отразится на увеличении затрат на проект, изменит финансовый план проекта, а также срок окупаемости проекта | Контролировать исполнения задач в срок и своевременное увеличение профессиональных компетенций сотрудников компании, в частности, отдела разработки |

# **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».».

Приложение Д

Код functional.py

Приложение Е

Код tok.py

Приложение Ж

Таблица 9 – Сравнение времени формирования doc с полным временем работы алгоритма

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Формирование doc | Doc + tokenize + lemmatize + remove\_spec\_stop\_words | | | | | |
| tokenize\_only | tokenize\_filter | tokenize\_ents | tokenize\_filter\_ents | tokenize\_checkPOS | tokenize\_filter\_checkPOS |
| 0,022734 | 0,018711 | 0,03336 | 0,015568 | 0,029413 | 0,0312 | 0,028958 |
| 0,016937 | 0,057843 | 0,033271 | 0,024493 | 0,035017 | 0,016052 | 0,015589 |
| 0,023141 | 0,015751 | 0,013332 | 0,019688 | 0,016886 | 0,015581 | 0,015081 |
| 0,008111 | 0,016282 | 0,015617 | 0,017802 | 0,01572 | 0,018128 | 0 |
| 0,016744 | 0,018263 | 0,016873 | 0,014611 | 0,015225 | 0,00833 | 0,013705 |
| … | … | … | … | … | … | … |
| 0,015635 | 0,01631 | 0,017135 | 0,021976 | 0,014581 | 0,015861 | 0,007357 |
| 0,02294 | 0,027336 | 0,031398 | 0,029377 | 0,032921 | 0,026942 | 0,016833 |
| 0,035534 | 0,021858 | 0,016709 | 0,015666 | 0,025808 | 0,01722 | 0,016844 |
| 0,01473 | 0,023019 | 0,011858 | 0,001024 | 0,020453 | 0 | 0,015668 |
| 0,047969 | 0,066114 | 0,062445 | 0,07111 | 0,112896 | 0,06667 | 0,046885 |