Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н. Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

НА ТЕМУ:

«Автоматическое извлечение ключевых слов и словосочетаний из единичного электронного документа русском языке»

Студент	ИУ7-86Б (Группа)	(Подпись, дата)	Барсуков Н. М. (И. О. Фамилия)
Руководитель ВКР		(Подпись, дата)	Барышникова М. Ю. (И. О. Фамилия)
Консультант		(Подпись, дата)	Строганов Ю. В. (И. О. Фамилия)
Нормоконтролер		(Подпись, дата)	Мальцева Д. Ю. (И. О. Фамилия)

2022 г.

Задание стр 1

Задание стр 2

Календарный план

РЕФЕРАТ

Расчетно-пояснительная записка 58 с., 17 рис., 1 табл., 17 источн., 1 прил.

Тема: "Автоматическое извлечение ключевых слов и словосочетаний из единичного электронного документа на русском языке"

Ключевые слова: КС, извлечение ключевых слов, извлечение ключевых словосочетаний, Yake, N-граммы, русский язык, pdf, ключевое слово.

Задачи решаемые в работе:

1. анализ существующих методов извлечения ключевых слов;
2. выбор основного алгоритма и определение направлений его модификации;
3. проектирование и разработка программного обеспечения для реализации метода;
4. экспериментальное исследование характеристик разработанного метода.

Область применения разрабатываемого метода - программные решений в области обработки текстов на естественных языках: поисковые и информационные системы, системы построения авторефератов.

В первой части работы проводится классификация методов извлечения ключевых слов, а так же их обзор. Во второй части проводится проектирование алгоритма извлечения КС на основе Yake. В третий части описываются детали реализации разработанного метода. Четвертая часть посвящена исследованию характеристик разработанного метода.

Поставленная в работе цель было достигнута: был разработ и реализован метод извлечения ключевых слов и словосочетаний из документов на русском языке, проведено исследования характеристик метода и предложены направления дальнейшего развития.

СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ

В настоящей расчетно-пояснительной записке применяют следующие термины с соответствующими определениями.

Ключевое слово — термин из текста описывающий тему документа

Извлечение ключевых слов — задача по извлечение терминов наилучшем образом описывающих тему документа

PDF — формат переносимых документов (Portable Document Format)

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

В настоящей расчетно-пояснительной записке применяют следующие сокращения и обозначения.

КС — ключевое слова

ПО — программное обеспечение

ИР — информационная революция

ЯП — язык программирования

TF — частота термина (term frequency)

ВВЕДЕНИЕ

В истории развития цивилизации выделяют несколько **информационных революций**. Первая была связана с изобретением письменности, благодаря чему появилась передача знаний между людьми без непосредственного общения. Вторая (17-18 вв.) была вызвана благодаря изобретению книгопечатания и реформами школьной системы, сделавшими возможным массовое образование и популяризацию знаний и сыгравшими важную роль в становлении индустриального общества. Третья и.р. (конец 19 - начало 20 вв.) началась с появлением телеграфа, телефона, радио, позволивших оперативно передавать информацию на любые расстояния и вступила в новую фазу с изобретением телевидения и компьютеров, а затем (кон. 1979-х гг.) с широким распространением информационных технологий. [1]

Количество знаний, накопленных цивилизацией, поражает и, благодаря продолжающейся информационной революции, стремительно растет. Традиционные медиа, такие как газеты и телевидение, мигрируют в интернет. Газеты и другие СМИ обновляют новостые ленты в реальном времени, что позволяет интересующимся получать свежую информацию. Ежедневно милионы пользователей интернета пораждают колоссальное количество контента для различных целей, по разным вопросам, в разных странах, на всевозможных языках и в многообразных онлайн средах. Это пользовательский контент в блогах и на сайтах социальных сетей, электронная почта, новости, научные работы и т.д. По всему миру государства, институты, библиотеки, музеи цифвровизируют свои материалы и выкладывают их во всемирную сеть, что бы информацию для бизнеса, науки, исследований, развлечений можно было получить через любое доступное нам "умное"устройство: телефон, планшет, компьютер и т.д. [2]

Таким образом, интернет и стал наиболее эффективными ресурсом для исследования современной экономики, культуры, политики, человеческого общения и взаимодействия людей. [2]

Объемы циркулирующей в мировых телекоммуникационных сетях и хранящейся на серверах информации демонстрируют динамику взрывного роста. На сегодняшний день, количество опубликованных документов достигает 1 биллио-

на веб-страниц. Все указанное обуславливает увеличение состава и сложности программных решений в области обработки текстов на естественных языках в основе которых лежит ряд базовых алгоритмов, в том числе выделения или извлечения ключевых слов (КС).

Извлечение ключевых слов (Keyword extraction) - это задача по автоматическому определению набора терминов, которые наилучшем образом описывают "основную мысль"документа. При изучении терминов, представляющих наиболее релевантную информацию, содержащуюся в документе, используется различная терминология: ключевые фразы, ключевые сегменты, ключевые термины, или просто ключевые слова. Однако, несмотря на достаточно большое количество исследований в данной направлении, автоматическое извлечение ключевых слов представляет собой проблему, которая до сих пор не решена [3]. Большинство существующих методов, успешно справляющихся с языками с бедной морфологией, такими как английский язык, не пригодны для естественных языков с богатой, в частности для русского языка, где каждая лексема характеризуется большим количеством словоформ с низкой частотностью в каждом конкретном тексте.

Методы способные на извлечения КС из русского языка, представляют в основном своем большинстве объемное программное обеспечение, требующее предварительного сбора и обработки корпуса текстов, относящихся к одной предметной области, что влечет за собой узконаправленность методов, что ограничивает область применения.

Целью данной работы является разработка метода для автоматического извлечения ключевых слов из электронных документов на русском языке, способного работать с одиночным документом и не зависимого от его предметной области.

Для достижения поставленной цели необходимо выполнить следующие задачи:

1. проанализировать современные методы извлечения ключевых слов;
2. отобрать метод, который будет использоваться в качетсве базового в данной работе и предложить направления его модификации;

3. описать основные шаги разработанного метода;

4. разработать программное обеспечение для реализации предложенного метода;

5. провести эксперементальное сравнение оригинального и модифицированного метода.

1 Аналитический раздел

В рамках данного раздела представлено описание предметной области задачи по извлечению ключевых слов. Приведена систематизация методов извлечения КС и текстов. Произведен отбор и сравнение методов, удовлетворяющих выставленным ограничениям. Отобран метод на базе которого будуте производиться модификация. Определено направление модификации метода.

1.1 Задача автоматического извлечения ключевых слов из текста на естественном языке

Первые попытки теоретического решения проблемы выделения ключевых ("опорных"или "обобщающих") слов была предпринята в работе А.Н. Соколова "Внутренняя речь и мышление"[4]. Основы современного понимания ключевых слов, можно сформулировать следующим образом [5]:

1. ключевые слова отображают тему текста;
2. их упорядоченность в наборе ключевых слов может трактоваться как эксплицитно невыраженная тема текста;
3. набор ключевых слов рассматривается как один из минимальных вариантов "текста";
4. такого типа "текст"характеризуется "ядерной"цельностью и минимальной связностью.

Ключевые слова - это одно или многокомпонентные лексические группы, отражающие содержание документа [6].

Извлечение ключевых слов (Keyword extraction) - это задача по автоматическому определению набора терминов, которые наилучшим образом описывают объект документа. При изучении терминов, представляющих наиболее релевантную информацию, содержащуюся в документе, используется различная терминология: ключевые фразы, ключевые сегменты, ключевые термины, или просто ключевые слова. Все выше перечисленные синонимы имеют одну и туже функцию - охарактеризовать обсуждаемую тему в документе [7]. Извлечение

маленького подмножества элементов, включающих от одного до нескольких терминов из одного документа, является важной проблемой в "Информационном поиске"(Information Retrieval, IR), "Интеллектуальном анализе текста"(Text mining, TM) и в "Обработке естественного языка"(Natural Language Processing, NLP).

Ключевые слова нашли широкое применение в запросах к системам информационного поиска, по сколько их легко определить, пересмотреть, запомнить и поделиться. По сравнению с математическими сигнатурами, они независимы от любого корпуса и могут применяться в нескольких корпусах и системах ИП [8]. Так же ключевые слова используются для улучшения функциональности информационно-поисковых систем. Другими словами они могут быть использованы для создания автоматического индекса для коллекции документов или, в качестве альтернативы, могут использоваться для представления документов в задачах категоризации или классификации [9].

Общая схема извлечения ключевых слов из текста практически одинакова для всех используемых методов и состоит из следующих шагов:

1. предварительная обработка текста:
2. (a) исключение элементов маркировки;
   (b) приведение слова к словарной форме;
   (c) удаление стоп слов, не несущих смысловой нагрузки (предлоги, союзы, частицы, местоимения, междометия и т.д.).
3. отбор кандидатов в ключевые слова;
4. фильтрация кандидатов в ключевые слова (анализ значимых признаков для каждого кандидата).

1.2 Систематизация методов

Доступные публикации описывают классификация методов автоматического извлечения КС разной степени полноты и детализации. В самом простом случае исследователи выделяют статистические методы и методы, основанные

на машином обучении. Схожая классификация приводится в работе отечественных авторов, которые рассматривают статистические и гибридные модели КС, на основе которых рассматриваются конкретные методы. Более развернутая классификация подразумевает выделение четырех стратегий:

1. не требующих обучения;
2. простых статистических методов;
3. лингвистических методов;
4. метод, основанных на машинном обучении, и их комбинаций.

В последних из доступных отечественных обзоров предметной области извлечения ключевых слов проводится классификация на основе типа системы распознавания которая подразумевает выделение [5]:

1. лингвистических;
2. статистических;
3. гибридных лингвостатистических метов.

Однако и эта, и прочие классификации не отражают весь спектр и специфику существующих решений.

Так как любой алгоритм извлечения ключевых слов по сути реализует одну или несколько систем распознавания образов, разбивающих входное множество слов на два класса:

1. ключевые;
2. прочие.

то предлагается использовать не иерархическую, а фасетную классификацию соответствующих методов и выбрать следующую совокупность признаков:

1. наличие элемента обучения и подходы к его реализации;

2. тип математического аппарата системы распознавания, обусловленного формой предствления признаков ключевых слов;

3. тип используемых для реализации метода лингвистических ресурсов.

По наличию обучения выделяют:

1. необучаемые;

2. обучаемые;

3. самообучаемые.

Более простые необучаемые методы подразумевают контекстно независимое выделение КС из отдельного текста на основе априорно составленных моделей и правил. Они подходят для гомогенных по функциональному стилю корпусов текстов, увеличивающихся со временем в объемах, на пример научных работ или нормативных актов. Обучаемые методы предполагают использование разнообразных лингвистических ресурсов для настройки критериев принятия решений при распознавании ключевых слов. Здесь большое значение имеет корректное выделение КС в выборке, используемой для обучения. Среди методов с обучением можно выделить подкласс самообучаемых, если обучение ведется без учителя, или с подкрепление на основе пассивной адаптации [10].

По второму признаку классификации прежде всего стоит выделить статистические и структурные методы извлечения КС. Статистические методы учитывают относительные частоты встречаемости морфологических, лексических, синтаксических единиц и их комбинации. Это делает создаваемые на их основе алгоритмы довольно простыми, но недостаточно точными, так как признак частотности ключевых слов не является превалирующим. Одним из классических методов в данном классе является расчет для каждого слова меры, отражающей его важность в тексте, рассматриваемого как элемент коллекции документов.

В основе структурных методов лежит представление о тексте как о системе семантически и грамматически взаимосвязанных элементов слов которые в свою очередь характеризуются набором лингвистических признаков. Поэтому многие

Рисунок 1.1 – Классификация методов извлечения ключевых слов

исследователи называют этот класс методов лингвистическим. Здесь в первом приближении могут быть выделены два подкласса: графовые и синтаксически шаблонные методы.

Графовые или граф-ориентированные методы представляют текс в виде множества слов-вершин или вершин-словосочетаний и ребер отношений между ними. Эти отношения могут выражать для каждой пары слов такие факты как последовательное появления в тексте, наличие слова в окне заданного размера и семантическую близость. Для вершин полученного графа вычисляются меры центральности и по пороговому критерию отбираются ключевые слова. Различия между данными методами состоят в особеностях учета значимости каждой вершины и вычисления отношений между ними. В основе синтаксических шаблонных методов лежит представление о регулярных синтаксических конструкциях, содержащих на определенных позициях ключевые слова. В чи-

стом виде такие методы слабо применимы к рассматриваемой задаче, могут использоваться в сочетании с другими.

Нейросетевые методы к задаче извлечения КС стали применяться сравнительно недавно. Они основаны на свойстве искуственных нейронных сетей к обобщению и выделению скрытых зависимостей между входными и выходными данными. Однако для формирования наборов данных для обучения и функционирования нейронных сетей требуется выделение структурных и статистических признаков, поэтому на практике методы выделения КС являются гибридными, т.е. сочетающими в себе элементы основных рассмотренных классов.

Наконец, алгоритмы извлечения КС, реализующие означенные методы, могут не использовать какие либо лингвистические ресурсы или использовать своего рода словари, онтологии, тезаурусы, а так же корпуса текстов с разметкой [10].

В данной работе не будут рассматриваться методы, требующие обучения или наличия корпуса текстов для обучения.

1.3 TF-IDF

TF-IDF представляет собой классическую числовую метрику, которая показывает релевантность ключевого слова в рамках выбранного документа. TF (Term Frequency) частота термина равняется количеству повторений кандидата в общем количестве слов в документе (1.1). IDF (Inevrsed Document Frequency) инвертированная частота документа, используется для балансировки излишне повторяющихся слов (1.2). Вычисляется как натуральный логарифм от частного - результата деления количества документов, в которых найден термин-кандидат, на общее количество документов.

$$TF = \frac{N_t}{N_{all}} \tag{1.1}$$

где N_t - количество термина в документе; N_{all} - колчество всех слов в документе.

$$IDF = \log(N_d/D) \tag{1.2}$$

где N_d - количество документов в которых было найден кандидат; D - общее количество документов.

1.4 Rake

Rapid Authomatic Keyword Extraction (Rake) - это графовый метод использующий лингвистический подход "совместного появления"(co-occurence) для извлечения ключевых слов из едининичного документа. Не требует обучения, не зависит от предметой области и языка документа.

Rake базируется на том что ключевые слова чаще всего представляют собой несколько слов и редко содержат пунктуацию и стоп слова такие как предлоги, речевые обороты и другие не значимые слова с минимальным лексическим значением. [11]

1.5 Yake

Алгоритм Yake относится категории комбинированных методов, который использует подходы как статистических методов, так и графовых. Состоит из 4 главных шагов:

1. предварительная обработка текста и определение термина-кандидата;
2. извлечение свойств;
3. вычисление счета термина;
4. вычисление дублей и ранжирование.

На первом шаге обрабатывается документ в машино-читаемый формат для определения потенциальных терминов-кандидатов. Это важный и ключевой шаг, от которого зависит качество определения кандидатов, что на прямую влияет на эффективность самого алгоритма. Следующей этап принимает на вход массив индивидуальных терминов и репрезентует их в виде набора статистических свойств. На третьем этапе эти свойства эвристически объединяются в единую оценку которая отражает важность значения термина. На финальном этапе сравниваются вероятно похожие термины на основе измерений схожести, полученных от специальных алгоритмов.

Алгоритм на вход получает текст и необходимые параметры: размера окна w (используется для вычисления одного из статистических свойств), порог повторения θ и язык текста, последний строчный параметр используется для получения списка стоп слов. Полученный текст метод начинает разделять на предложения. Каждое предложение в результате предобработки превращается в некоторое количество терминов, что завершает первый этап. На следующем шаге для каждого термина высчитываются его статистические свойства и подсчитывается оценка. На завершающем шаге происходит отсеивание кандидатов на ключевое слово, где свойство схожести больше θ. Дальше список ключевых слов и оценки сортируются по релевантности и возвращаются данным методом, что завершает работу алгоритма. Ниже приведено более детальное описание каждого шага.

1.5.1 Предварительная обработка текста и определение списка кандидатов

Предварительная обработка текста - это первый этап, после которого идет репрезентация текста и его анализ. На данном этапе идет очищение текста и трансформация его в машино-читаемый формат: выделены важные части и удалены шумовые слова. Классическая предобработка включает в себя: очистку текста, разбиение на предложения, аннотирование текста, токенизацию и определение стоп слов. Так же могут использоваться техники обработки текстов на естественном языке (natural language processing, NLP):

1. пометка частей речи (part-of-speech tagging, PoS);
2. распознавание именованных объектов (named-entity recognition, NER);
3. нормализация;
4. лингвистический разбор или стемминг;

однако, для этого требуются специальные инструменты, каждый из которых работает с тем языком для которого он был реализован. В рамках этапа данного алгоритма они не используются.

Поступивший текст алгоритм начинает делить на предложения. Это достигается путем применения сегментера текстов основанного на правилах, который разделяет индоевропейские языки на высказывания следую предопределенному шаблону. На пример: "Python is awesome! But C++ is also very good"будет разделено на 2 выражения ("Python is awesome! "But is awesome") в то время как "Mr. Smith"будет одиночным выражением. Затем каждое высказывание делится на группы (по найденой пунктуации) и затем проходит процесс токенизации. Это очень важный этап который не только позволяет выделить слова, но и отсеять шум в виде пунктуации, адресов электронных почт, и ссылок, которые мешают при работе с огромными текстами. После этого каждый токен приводится к нижнему регистру и помечается специальной меткой разделителем.

Результат предобработки - это список предложений, разделенных на группы, сформированные из помеченных терминов. В следующем разделе приводится описание статистических свойств, использующихся в данном методе.

1.5.2 Вычисление свойств

После предварительной обработки и получения кандидатов применяется статистический анализ, который уделяет внимание структуре, частоте терминов и их сочетаемости. В начале создается пустая структура, которая будет хранить в себе одиночные термины, найденные в тексте, и дополнительную информацию, такую как результаты статистических метрик и оценку (вычисляется позже). Затем перебираем список выражений и чанки. Каждый чанк разбивается на ранее размеченные токены и для них вычисляются:

1. частота термина (TF, term frequency);
2. индекс выражений, где встречается данный токен (offsets_sentences);
3. частота акронима термина (TF_a, term frequency of acronym);
4. частота слова в старшем регистре (TF_U, term frequency of uppercase);

в дополнении к ним так же вычисляется матрица сочетаемости для сохранения связи между термином и его предшественником или подтермины, которые найдены в окне размером w. После того, как статистика каждого термина вычислена,

можно проводить процесс извлечения признаков. Дли одиночного термина извлекаются следующие признаки: TCase, TPos, TFNorm, Trel и Tsent, детальное описание каждого признака идет ниже.

1.5.3 Регистр (T_{Case})

Аспект термина связанный с регистром является важной характеристикой при рассмотрении извлечения ключевых слов. Основная идея состоит в том, что термины в верхнем регистре, как правило, более релевантны, чем слова в нижнем регистре. В представленом подходе уделяется особое внимание любому термину, начинающемуся с заглавной буквы (исключая начало предложений), тем более аббревиатуры где все буквы слова заглавные. Однако вместо того, чтобы считать это двойным весом, мы будем рассматривать только максимальное вхождение в пределах двух из них. Уравнение (1.3) отражает внешний вид термина-кандидата:

$$T_{Case} = \frac{max(TF(U(t))), TF(A(t))}{ln(TF(t))} \tag{1.3}$$

где $TF(U(t))$ количество термина-кандидата t начинающегося с заглавной буквы, $TF(A(t))$ сколько раз кандидат t был отмечен как акроним и $TF(t)$ - это частота t. Таким образом чем чаще кандидат пишется с заглавной буквы, тем более важным он считается. Это означает, что термин-кандидат, встречающийся с заглавной буквой 10 из 10 случаев будет иметь большее значение, чем коллега, встретившийся 5 раз из 5 случаев.

1.5.4 Позиция термина ($T_{position}$)

Другим индикатором важности кандидата является позиция. Причина в том, что релевантные слова, как правило, появляются в самом начале документа, тогда как слова, встречающиеся в середине или в конце документа, имеют тенденцию быть менее важными. Особенно это очевидно как для новостных статей, так и для научных текстов, двух видов публикаций, которые склонны концентрировать большое количество важных ключевых слов в верхней части текста. Например, во ведении или в аннотации. Предположение авторов состоит в том, что термины, встречающиеся в первых предложениях текста должны быть

более высоко оценены, чем термины, которые появляются позже. Таким образом, вместо того, чтобы рассматривать равномерное распределение терминов, их модель присваивает более высокие баллы терминам, встречающимся в первых предложениях. Вес рассчитывается, используя следующее уравнение (1.4):

$$T_{position} = ln(ln(3 + Median(Sen_t))) \tag{1.4}$$

где Sen_t множество позиций в выражениях, где кандидат t появляется и медиана от Sen_t. Зная что медианая функция $Median(Sen_t)$ может возвращать значение 0 (когда термн-кандидат появляется только в первом предложении), к уравнению добавляются константа $C > e$, в случае данной реализации 3 (как первое число после e), что бы гарантировать что $T_{position} > 0$. Так же для сглаживания разницы между терминами с большой средней разницей используется двойной логарифм.

Так для термина-кандидата t_1 который появляется в 2, 35, 70, 74, 4000 выражениях в документе будет иметь $T_{position} = ln(ln(3 + Median(2, 35, 70, 74, 4000))) = 1.45$ в то время как t_2, появляющийся в 1, 4 и 7 выражении получит $T_position = ln(ln(3 + Median(1, 4, 7)) = 0.66$. Результатом является возрастающая функция, значения которой имеют тенденцию плавно возрастать по мере того, как термины-кандидаты располагаются ближе к концу документа, а это значит, что чем больше кандидатов появляются в начале документа, тем ниже его значение $T_{position}$ и наоборот терминам (менее актуальные), расположенным ближе к концу документа, будет присвоено более высокое значение $T_{position}$.

1.5.5 Нормализация частоты термина (TF_{norm})

Эта функция указывает частоту термина-кандидата t в документе на основе работе Луна [12], которая утверждает что "частота появления слова в тексте обеспечивает полезное измерение значения слова". Данное высказывание отображает убеждение, что чем выше частота кандидата, тем выше его важность. Тем не менее, это не означает что важность пропорциональна тому сколько раз встречается термин. Таким образом для предотвращения смещения в сторону высоких частот в длинных документах значение TF кандидата t делится на сред-

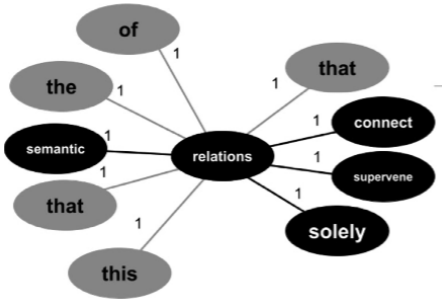
нее значение частот (MeanTF) плюс 1, умноженное на стандартное отклонение (1.5).

$$TF_{norm} = \frac{TF(t)}{MeanTF + 1 * \sigma} \tag{1.5}$$

Цель заключается в том, что бы оценить все термины-кандидаты, частоты которых выше среднего, сбалансированного определенной степенью дисперсии, заданной стандартным отклонением. Чтобы вычислить это авторы метода решили учитывать только MeanTF и стандартное отклонение не стоп-слов, что гарантирует, что на вычисления этих двух компонентов не влияют высокие частоты, регистрируемые по стоп-словам.

1.5.6 Связь термина с контекстом

Хотя стоп-слова представляют собой бесспорно полезный источник знаний о том, что явно не относится к делу, только полагаясь на статическом списке,информации может оказаться недостаточно для отбрасывания нерелевантных слов. В данном разделе описывается статистическая функция, которая нацелена на определение дисперсии (D) термина-кандидата t относительно его конкретного контекста, опирающаяся на работу Machado [13], утверждающая что чем выше число различных терминов, которые встречаются вместе с терминов-кандидатом t с обеих сторон, тем менее значимым будет термин t



w=2	w=1	t	w=1	w=2
of	the	**relations**	that	connect
	semantic	**relations**		
that	this	**relations**	supervene	solely

Рисунок 1.2 – Совместное появление при w = 2

$$DL[DR] = \frac{|A_t, w|}{\sum_{k \in A_{t,w}} CoOccur_{t,k}} \tag{1.6}$$

где $|A_t, w|$ представляет собой количество различных терминов, где термин представляет собой анализируемое содержимое, аббревиатуру, или слово в верхнем регистре, которое появляется слева(справа) от термина-кандидата t в заданном окне размера w (1.2) в отношении k терминов с которыми он встречается.

Затем DL и DR умножаются на частоту термина-кандидата, деленную на максимальную частоту термина среди всех кандидатов, которые встречаются в документе. Окончательное уравнение имеет вид (1.7):

$$T_{Rel} = 1 + (DL + DR) * \frac{TF_t}{MaxTF} \tag{1.7}$$

где крайняя левая часть уравнения измеряет значимость термина-кандидата по отношению к его левой части, а самая правая часть измеряет свою значимость по отношению к правой стороне. С практической точки зрения, чем менее релевантен кандидат t, тем выше будет оценка этой функции. Таким образом, стоп-слова и, аналогичным образом, недискриминационные термины, как правило, получают более высокий балл. На рис. (1.2) показаны различные термины, которые встречаются вместе с термином-кандидатом «relations» в пределах w, равного 2, где «relations» — неважный термин (не часть истины, т. е. не золотое ключевое слово) в контексте нашего текущего примера. В правой части рисунка показаны отрывки из текста, в которых встречается термин «отношения», а в левой — графическое представление. Число на ребрах — это частота, с которой термин-кандидат «relations» встречается вместе в соответствующем овале. Серым цветом обозначены стоп-слова.

1.5.7 Другое предложение термина ($T_{sentence}$)

Это свойство подсчитывает как часто термин-кандидат встречается в других предложениях. Оно отображает, что кандидат, встречающийся во многих других предложениях, можеть быть более важным. Оценка подчитывается по следующей формуле:

$$T_{sentence} = \frac{SF(t)}{Sentences} \tag{1.8}$$

где $SF(t)$ - количество предложений в которых появляется t. $Sentences$ - это максимальное количесвто предолжений в тексте. Результат лежит в диапозоне от $[0, 1]$

1.5.8 Подсчет оценки термина

После того как все оценки свойств получены, можно подсчитать финальный результат оценки термина. Все полученные ранее оценки подставляются в выражение S(t) (1.9).

$$S(t) = \frac{T_{Rel} * T_{Postion}}{T_{Case} + \frac{TF_{Norm}}{T_{Rel}} + \frac{T_{sentences}}{T_{Rel}}} \tag{1.9}$$

Стоит обратить внимание на то что TF_{Norm} и $TF_{sentensec}$ делятся на T_{Rel}. Это делается для того чтобы присвоить большое значение терминам, которые появляются часто во многих предложениях, до тех пор пока они релевантны. Поскольку, некоторые кандидаты могут встречаться много раз в многих предложениях и при этом быть бесполезными и должны быть "оштрафованы". Таким образом достижение высокой оценки по TF_{Norm} и $TF_{sentensec}$ является индикацией важности термина при низком значении T_{Rel}. Так же важным свойством является позиция появления термина, она добавляется в выражение путем умножения $T_{Rel} * T_{Postion}$

1.6 Сравнение методов

1. TF-IDF
2. Плюсы
   (a) простой.
3. Минусы
   (a) требует наличия большого корпуса текстов;
   (b) не работает со словосочетаниями
   (c) оценка происходит только по частотной характеристеке термина

4. Rake
5. Плюсы
    (a) простой
    (b) не требует наличия корпуса текстов;
    (c) учитывает взаимосвязи слов в тексте.
6. Минусы
    (a) оценка происходит только по частотной характеристике термина
    (b) не выдает словосочетания
7. Yake
8. Плюсы
    (a) учитывает взаимосвязи слов;
    (b) учитывает позицию кандидата;
    (c) учитывает его написание (абревиатуры, регистры);
    (d) учитывается связь с контекстом
9. Минусы
    (a) Не выдает словосочетания

Учитывая все ранее выставленные требования к алгоритмам было принято решение использовать метод Yake. До этого данный метод не использовался для вычисления ключевых слов из текста на русском языке и для этого необходимо провести следующие модификации:

1. добавление сборника стоп-слов;
2. добавление работы с N-gram;

1.7 Модификация

Шумовые слова (или стоп-слова) — термин из теории поиска информации по ключевым словам. Это такие слова, знаки, символы, которые самостоятельно не несут никакой смысловой нагрузки, но которые, тем не менее, совершенно необходимы для нормального восприятия текста, его целостности. К ним относятся предлоги, суффиксы, причастия, междометия, цифры, частицы и т. п. Так как данные слова имеют самые большие частоты встречи в документах, то для более корректного результата небходимо данные слова исключать из текста, по этому есть необходимость в добавлении в метод списка стоп-слов для русского языка.

Как было описано ранее, результатом работы метода Yake являются одиночные ключевые слова с оценкой. Современные технологии непрерывно развиваются. Это значительно влияет на научно-технический функциональный стиль, в частности, на его терминосистемы. Появляются новые термины и усложняются старые [14]. В результате растет количество многокомпонентных терминов. По этому есть необходимость в модификации выбранного метода для возможности извлечения многокомпонентных ключевых слов. Данную модификацию проведем с помощью добавления n-грамм. Пусть задан некоторый конечный алфавит $V = w_i$, где w_i - символ. Языком $L(V)$ называют множество цепочек конечной длины из символов w_i. Высказыванием называют цепочку из языка. N-граммой на алфавите V называют произвольную цепочку длинной N, например последовательность из N букв русского языка, одного слова, одной фразы, одного текста [15].

Конечный результат будет высчитываться по формуле (1.10):

$$S(kw) = \frac{\prod_{t \in kw} S(t)}{KF(kw) * (1 + \sum_{t \in kw} S(t))} \tag{1.10}$$

где $S(kw)$ - финальный результат оценки, чем ниже тем реливантнее, $S(t)$ - оценка одиночного термина, $KF(kw)$ - частота ключевого слова(выражения)

1.8 Заключение

В рамках данного раздела представлено описание предметной области задачи по извлечению ключевых слов. Приведена систематизация методов извлечения КС и текстов. Произведен отбор и сравнение методов удовлетворяющих выставленным ограничениям. Отобран метод на базе которого будет проводиться модификация Определены направления модификации выбранного метода Yake.

2 Конструкторский раздел

В данном разделе приведено описание структуры алгоритма и отдельных его этапов с помощью диаграммы IDEF0. Продемонстрирована схема этапа предварительной обработки текста. Представлена архитектура и выставлены требования к пользовательскому интерфейсу разрабатываемого программного обеспечения. Продемонстрирована диаграмма классов, использующихся в работе.

2.1 Yake

Общая структура метода извлечения ключевых слов представлена на рисунках 2.1 - 2.2 Входными параметрами данного алгоритма являются тест

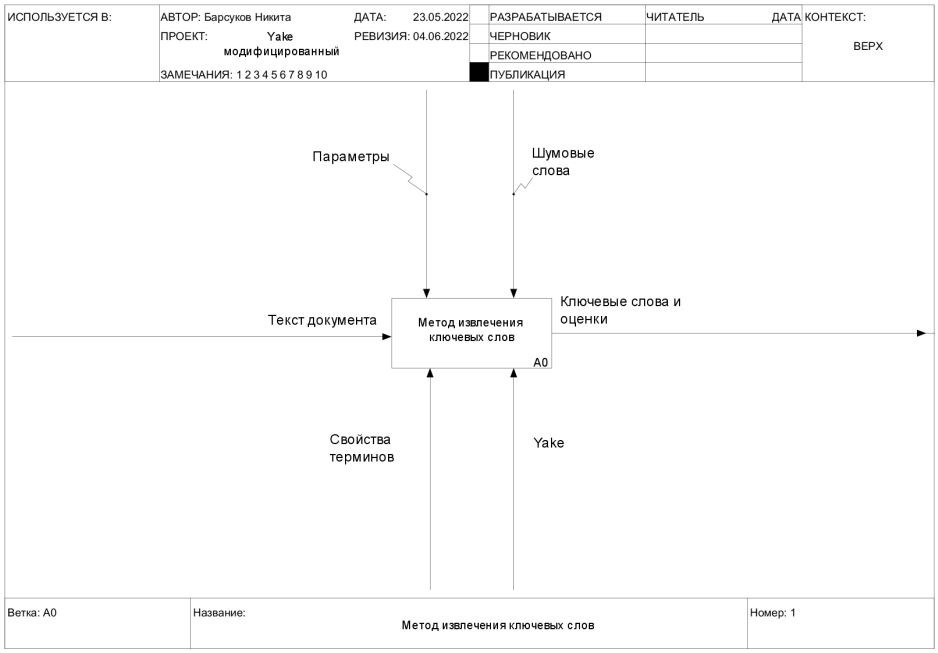


Рисунок 2.1 – IDEF0 диаграмма разрабатываемого метода

извлеченный из электронного документа и язык работы. Результатом исполнения данного метода является список, состоящий из кортежей, содержащих в себе термин и его оценку.

Данный метод можно разделить на несколько основных этапов, которые представлены на IDEF0-диаграмме, изображенной на рисунке 2.2

1. предварительная обработка текста и выделение кандидатов (рисунок 2.2 блок A1)

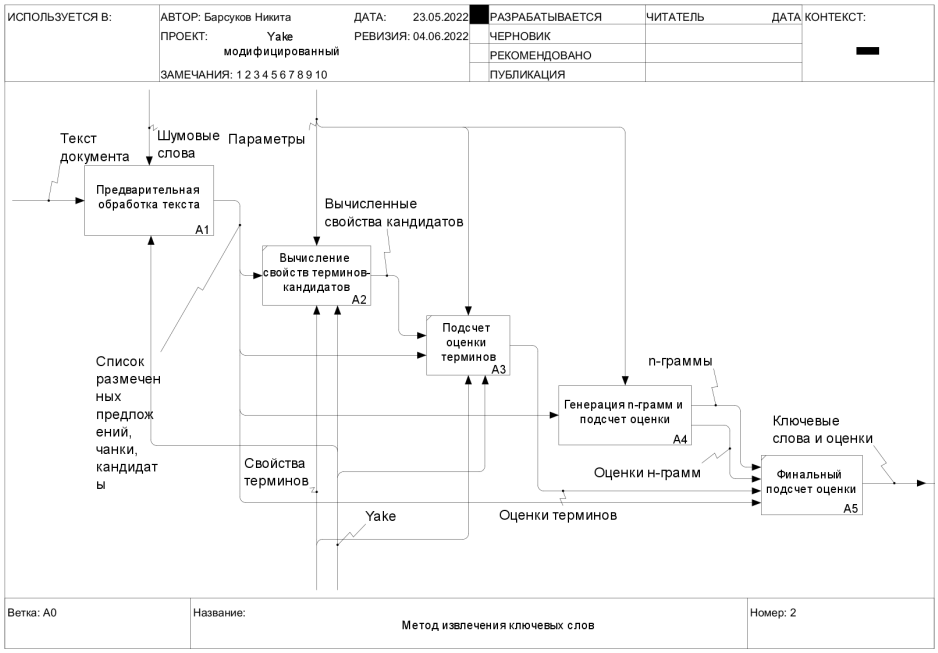


Рисунок 2.2 – IDEF0 диаграмма модуля извлечения ключевых слов из текста

2. вычисление свойств термин-кандидатов (рисунок 2.2 блок A2)
3. подсчет оценки кадидатов (рисунок 2.2 блок A3)
4. генерация н-грамм и подсчет оценки (рисунок 2.2 блок A4)
5. подсчет финальной оценки терминов (рисунок 2.2 блок A5)

Формально весь процесс работы алгоритма отображен на изображениях 2.3 - 2.5

В начале происходит предварительное форматирование текста, путем замены спецсимволов, таких как табуляция, перенос на новую строку и т.п. на пробелы. Затем текст разбивается на предложения, а сами предложения на сегменты, используются на этапе оценки свойств кандидатов и построении н-грамм. После этого из полученных данных извлекаются уникальные термин-кандидаты с контекстной информацией.

Для оценки валидности терминов необходимо вычислить их статистические свойства и характеристики. На рисунках **??**, **??** и **??** отображена схема вычисления характеристик, где TF - частота термина, TF_U - частота термина в верхнем регистре, TF_a - частота термина в виде аббревиатуры. Теперь

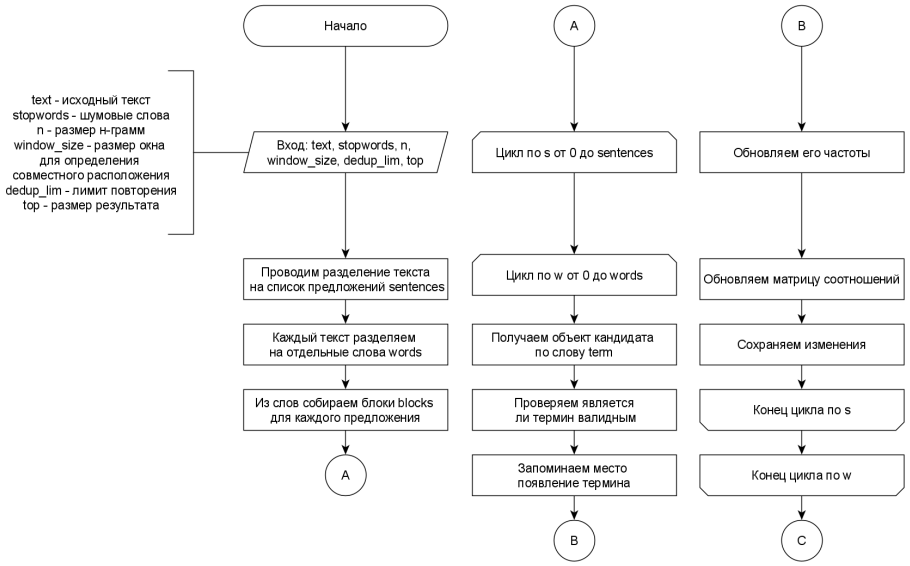


Рисунок 2.3 – Предварительная обработка текста

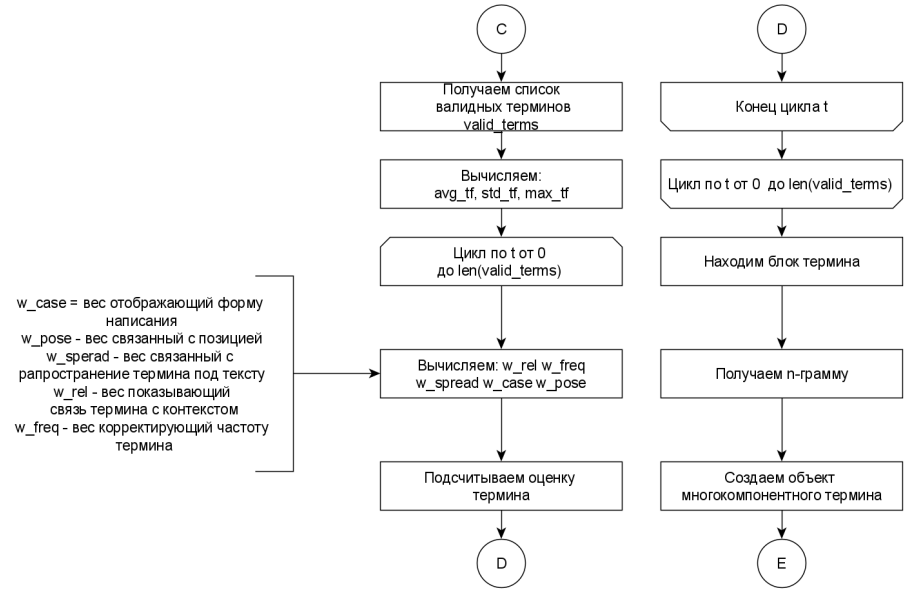


Рисунок 2.4 – Вычисление висов методов

необходимо вычислить свойства терминов, процесс вычисления отображен на рисунках 2.6

После полученияx всех свойств для каждого термина идет процесс выбора н-грамм и подсчет оценки. Так оценка н-грамм строится на основе оценок одиночных терминов, то остается только объединить результаты и отсортировать полученный список по релевантности.

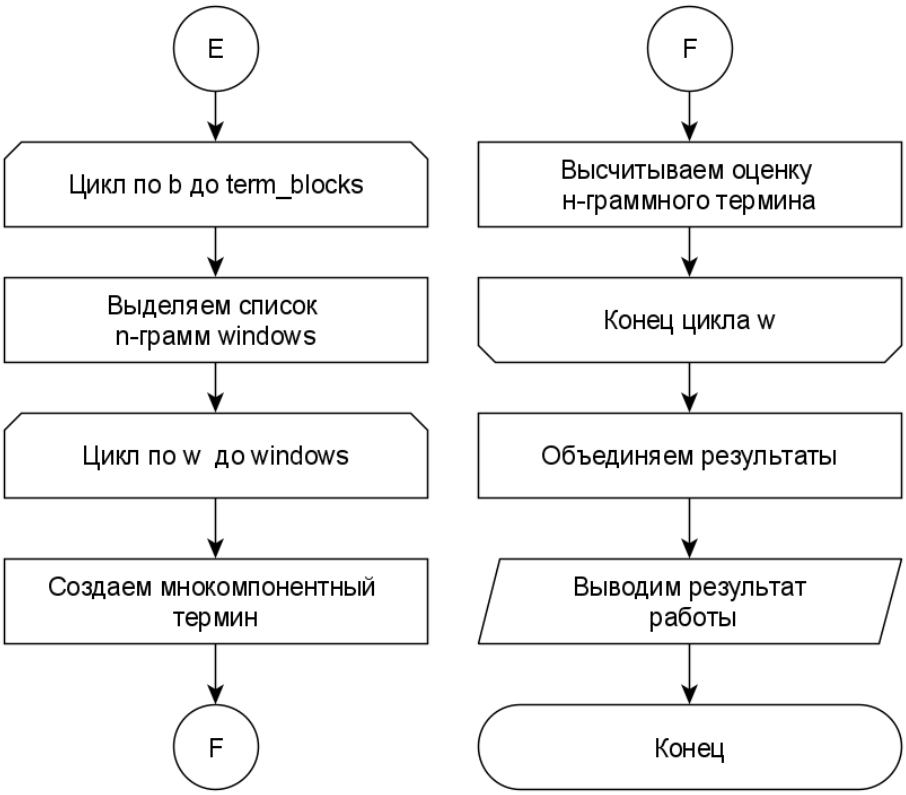


Рисунок 2.5 – Вычисление N-грамм

2.2 Архитектура ПО

Для реализации программного обеспечения была выбрана MVC (Model-View-Controller) архитектура, разбивающая программу на три отдельных компонента:

1. представление - это отображение состояния внутренней системы;
2. модель - это компонента, отвечающая за предоставление данных конкретным элементам системы;
3. контроллер - это связующее звено между представлением и моделью, обрабатывает действия пользователя, полученные от представления и отдает команды модели.

Благодаря использованию MVC подхода к организации архитектуры ПО, воздействие в случаи модификации или замены модуля на другие компоненты сводится к минимуму или полностью отсутствует, что добавляет системе гибкости.

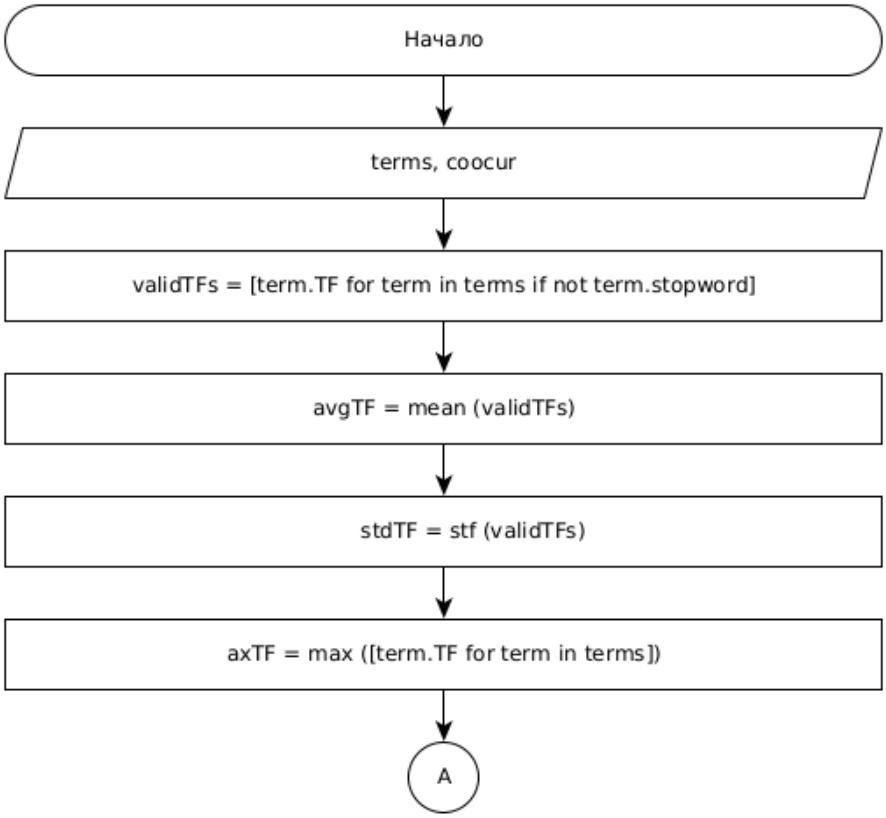


Рисунок 2.6 – Вычисление общих свойств

Это достигается путем разделения: работы с данными, логики взаимодействия пользователя с интерфейсом. Отличительной чертой данного решения является принцип чистой архитектуры, основанный на разделении данных. Каждый компонент системы должен легко и просто заменяться другим и на оборот. Это же касается используемых инструментов, библиотек, фрейворков, то есть система не привязана к конкретным технологиям и их в любой момент можно заменить.

Через графический интерфейс у пользователю должна быть возможность взаимодействия с программный ПО. Под взаимодействием подразумевается запуск ПО и получения результата. Предоставляющийся функционал:

1. выбор из списка методов извлечения КС от одного до нескольких алгоритмов;

2. выбор одного или нескольких файлов через специальное окно;

3. установка параметров для методов в отдельном окне;

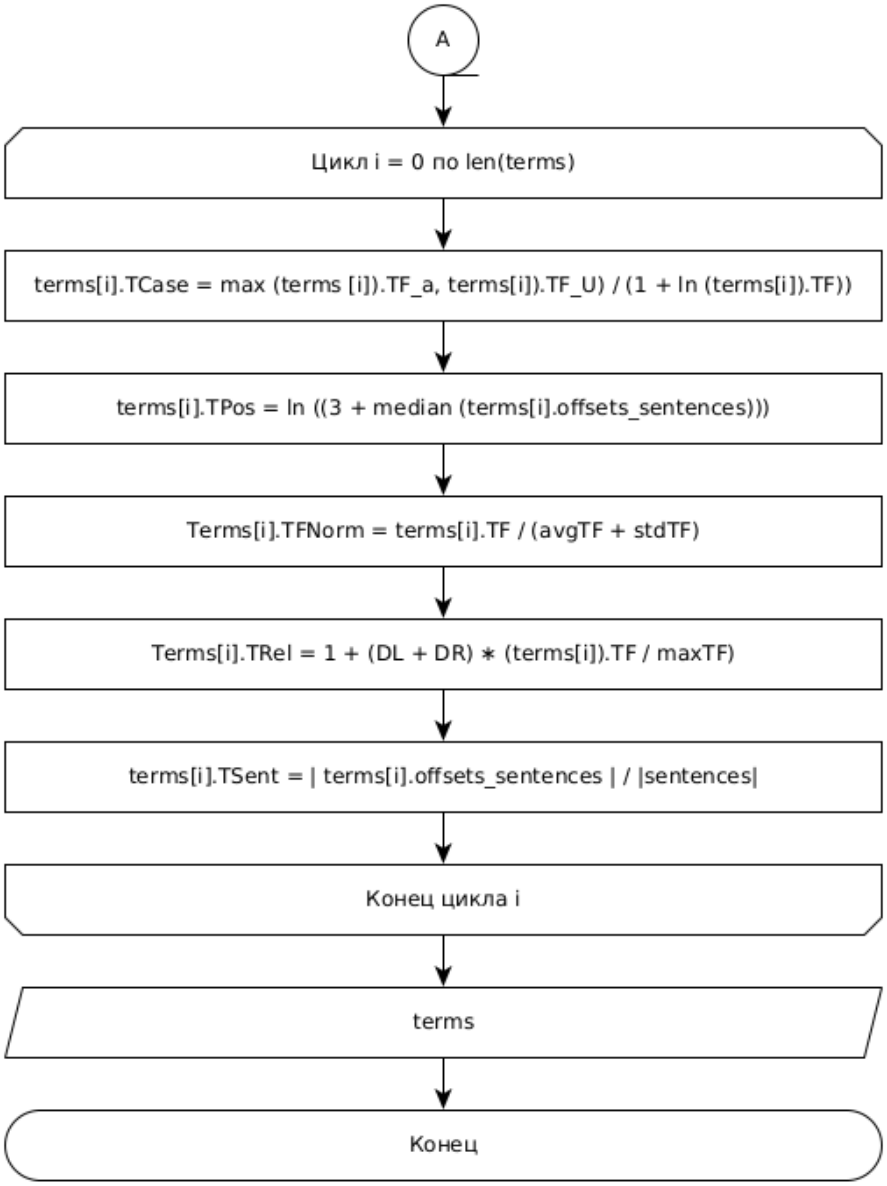


Рисунок 2.7 – Вычисление свойств для каждого термина

4. возможность ввести отдельно текстовую информацию.

На рисунке 2.9 схематично отображены основные классы разрабатываемого программного продукта

Класс MainWindow - это входная точка приложения. Он выполняет основную логику, связанную с обработкой пользовательский запросов к графическому интерфейсу. Он взаимодействует с классом MethodControler, отвечающим за подгрузку, подготовку методов к работе и запуск процесса извлечения ключевых слов. Сами методы представлены в виде классов.

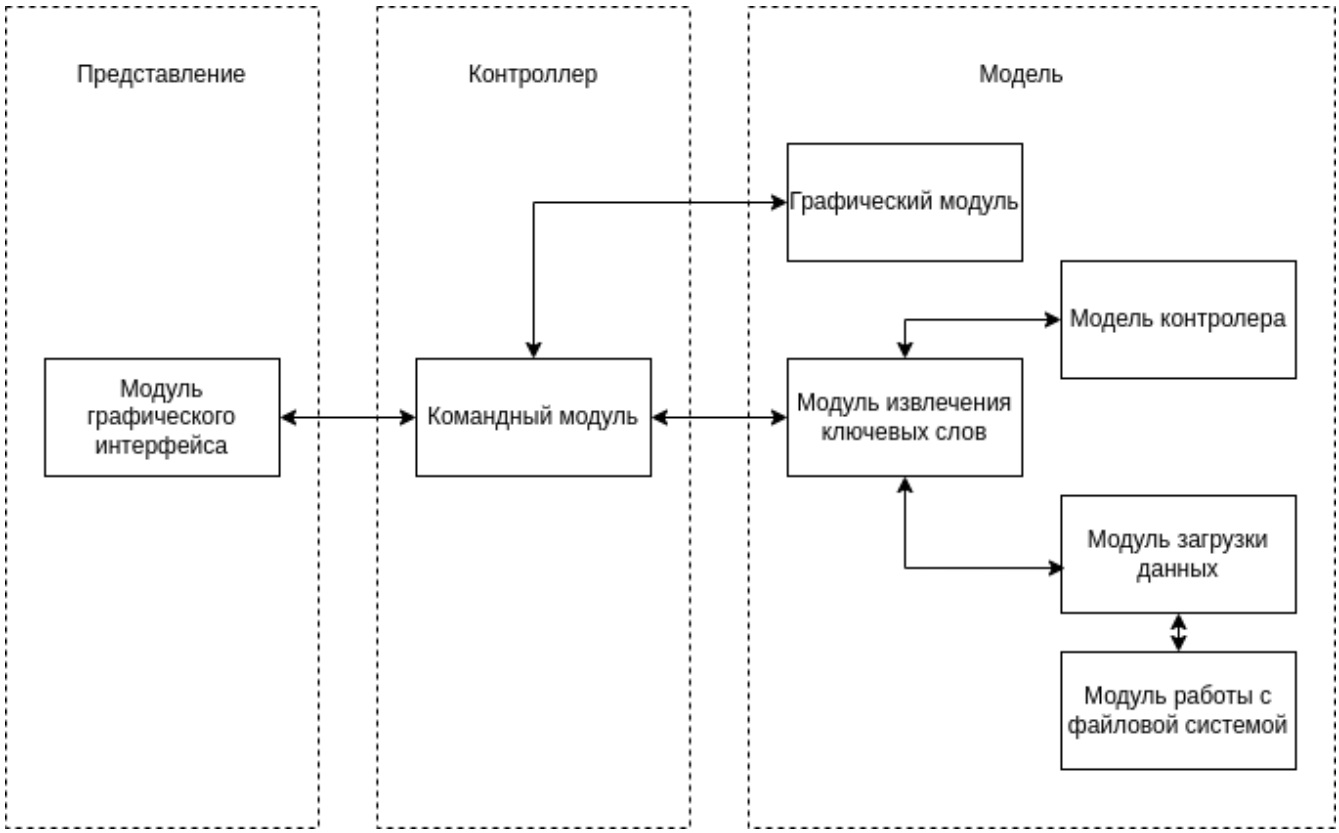


Рисунок 2.8 – Схематичное преставление архитектуры ПО

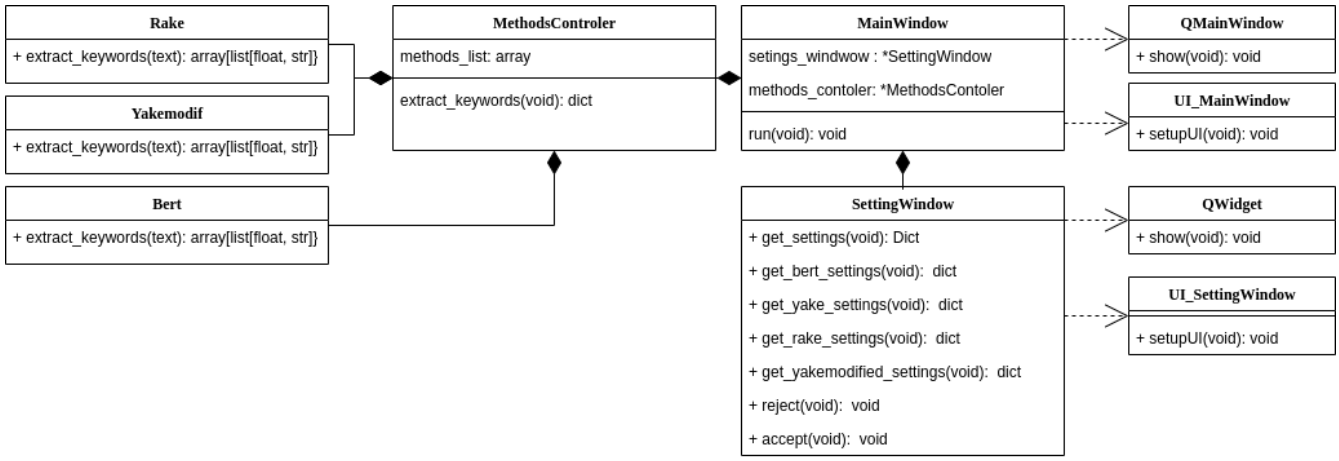


Рисунок 2.9 – Схематическое представление архитектуры классов программного обеспечения

2.3 Вывод

В данном разделе было спроектировано программное обеспечение для извлечения ключевых слов и словосочетаний из электронного документа на русском языке. Представлены диаграммы IDEF0 и описаны основные компоненты

ПО. Результаты проектирования:

1. спроектирована архитектура разрабатываемого ПО;
2. модификация алгоритма путем применения н-грамм;
3. разработана структура ПО.

3 Технологический раздел

В разделе проведен обзор актуальных языков программирования на основе которого был выбран ЯП для реализации разрабатываемого программного обеспечения. Приведено описание используемых библиотек и фреймворков. Приведено описание входных и выходных данных. Продемонстрирована работа ПО на рисунках 3.1, 3.2 Обоснован выбор интегрированной среды разработки IDE и системы контроля версий VCS

3.1 Системные требования

Каждое программное обеспечение требует материальной базы, в рамках которой оно будет функционировать. Описание характеристик такой базы, называется системными требованиями. Данные требования предоставляют пользователю информацию об аппаратном обеспечении, необходимом для использования ПО.

1. Операционная система: Ubuntu 18.04.6 TLS
2. Процессор: Intel(R) Core(TM) i5-8265U CPU @ 1.60GH 3900,00 MHz
3. Жесткий диск: 500 гб
4. Оперативная память: 16гб

3.2 Язык программирования

Продолжающееся развитие компьютерных технологий и широкое их распространие, спровоцировало спрос на развитие и создание новых знаковых систем для записи алгоритмов, которые известны на сегодняшний день как языки программирования (ЯП). Сформируем список из наиболее популярных и проведем обзор для установления их соответствия поставленной задаче.

1. Python;
2. C++;
3. C#;

В данный список не попали языки, относящиеся к веб-разработке, поскольку разрабатываемое программное обеспечение представляет собой декстопное приложение.

3.2.1 Python

Python – это высокоуровневый язык программирования общего назначения, который ориентирован на повышение читаемости кода и удобства использования разработчиком. ЯР Python минималистичен. В то же время стандартная библиотека содержит в себе разнообразный набор полезных, в ходе разработки, функций.

Python поддерживает несколько парадигм программирования, в том числе:

1. структурное;
2. объектно-ориентированное;
3. императивное;
4. аспектно-ориентированное;
5. функциональное.

Основными архитектурными чертами, являются такие особенности, как автоматическое управление памятью, динамическая типизация, механизм обработки исключений, полная интроспекция, гибкие высокоуровневые структуры данных и поддержка многопоточных вычислений. Код в Python организовывается в классы и функции, которые могут объединяться в модули, которые, в свою очередь, могут быть объединены в пакеты.

Эталонной реализацией Python является интерпретатор CPython, поддерживающий большинство активно используемых платформ. Он распространяется под свободной лицензией Python Software Foundation License, которая позволяет использовать его без ограничений в любых приложениях, включая проприетарные. Наиболее часто Python сравнивают с Ruby и Perl. Эти языки обладают примерно одинаковой скоростью выполнения программ и также являются интерпретируемыми.

3.2.2 C++

C++ - это универсальный язык программирования. За исключением второстепенных деталей C++ является надмножеством языка программирования C. Помимо возможностей, которые дает C, C++ предоставляет эффективные и гибкие средства определения новых типов. Используя определения новых типов, точно отвечающих концепциям приложения, программист может разделять разрабатываемую программу на легко поддающиеся контролю части. Такой метод построения программ часто называют абстракцией данных. Информация о типах содержится в некоторых объектах типов, определённых пользователем. Такие объекты просты и надёжны в использовании, в тех ситуациях, когда их тип нельзя установить на стадии компиляции.

Программирование с применением таких объектов часто называют объектно-ориентированным. При правильном использовании этот метод даёт легче контролируемые программы, более короткие и проще понимаемые. C++ предлагает программисту полный набор операторов структурного программирования. Он также обладает очень большим набором операций. Многие операции C++ соответствуют машинным командам, и поэтому допускают прямую трансляцию в код ассемблера. Разнообразие, предоставляемое C++ позволяет выбирать их различные наборы для минимизации результирующего поля. C++ поддерживает указатели на переменные и функции. Указатель на объект программы соответствует машинному адресу данного объекта. Посредством разумного использования указателей можно создавать эффективные программы, которые выполняются быстро, так как указатели позволяют ссылаться на объекты тем же самым путём, как это делает машина.

3.2.3 C#

C# – это объектно-ориентированный язык программирования. Разработан в 1998-2001 годах в компании Microsoft в качестве языка разработки приложений для платформы Microsoft .NET Framework и впоследствии был стандартизирован как ISO/IEC 23270 и ECMA-334. C# относится к семье языков с C-подобным синтаксисом, среди них его синтаксис наиболее близок к C++ и Java. Язык

поддерживает полиморфизм, перегрузку операторов (в том числе операторов неявного и явного приведения типа), имеет статическую типизацию, атрибуты, делегаты, свойства, события, обобщённые методы и типы, анонимные функции с поддержкой замыканий, итераторы, LINQ-запросы, комментарии в формате XML и исключения.

В результате сравнения языков программирования оптимальным является Python. Python будет использоваться для модулей логики и взаимодействия. Установление связи между компонентами будет обеспечено с помощью применения слотов и сигналов из библиотеки PyQt5.

3.3 Формат входных данных

На вход программы ожидается файл формата PDF, содержащий в себе текст, написанный на русском языке, длинной не меньше 50 слов, относящийся к одной теме (статье). Для хранения файлов используется файловая система.

Формат переносимых документов (PDF) представляет собой универсальный тип файлов, который позволяет сохранить шрифты, изображения и сам макет исходного документа, независимо от того, на какой из множества платформ и в каком из множества приложений такой документ создавался. Формат Adobe PDF считается признанным общемировым стандартом в области тиражирования и обмена надежно защищенными электронными документами и бланками [16]

3.4 Библиотеки

Современное программное обеспечение состоит из множества компонентов. Полная реализация всех необходимых модулей самостоятельно, увеличивала бы и без того продолжительный процесс разработки. Для решения данной проблемы используются библиотеки и фреймворки, которые предоставляют часть или полностью готовый функционал. Для реализации данной работы использовались следующие библиотеки:

1. numpy;
2. networkx;

3. segtok;

4. jellyfish;

5. poetry;

6. textract;

NumPy - это фундаментальный пакет для научных вычислений в Python. Который предоставляет многомерный объект массива, различные производные объекты (такие как маскированные массивы и матрицы) и ассортимент подпрограмм для быстрой работы на массивах, в том числе манипулирование формой, сортировка, выбор, дискретные преобразования Фурье, базовая линейная алгебра, основные статистические операции, случайное моделирование и многое другое.

jellyfish - представляет из себя набор функций для стемминга (процесса нахождения начальной формы слов), реализаций методов нахождения редакционного расстояния таких как: расстояние Левенштейна, Дамерау-Левенштейна, Хэмминга и Жаро.

Segtok - библиотека содержащая две модели segmenter и tokenizer. Segmenter предоставляет функционал по разделению текста Индо-Европейских языков на предложения. Tokenizer предоставляет инструмент для разбиения предложений на слова и символы.

Networkx - это библиотека для теории графов и средство моделирования сети, разработанное на языке Python, которое содержит встроенные графы и сложные алгоритмы сетевого анализа. Networkx, позволяет хранить сети в стандартизированных и нестандартизированных форматах данных, анализировать сетевые структуры, создавать модели сетей, разрабатывать новые сетевые алгоритмы и выполнять рендеринг сети. Networkx поддерживает создание простых ориентированных, неориентированных и мульти-графов; во многих стандартных алгоритмах теории графов узлами могут быть любые данные; поддерживается любое измерение граничных значений. Прост в использовании. В рамках работы, используется для определения совместного появления терминов, на этапе оценки свойств методов.

Poetry - это инструмент для управления зависимостями и сборкой пакетов в Python. В Poetry представлен полный набор инструментов, которые могут понадобиться для детерминированного управления проектами на Python. В том числе сборка пакетов, поддержка разных версий языка, тестирование и развертывание проектов.

3.5 Графический интерфейс

Для реализации пользовательского интерфейса была выбрана библиотека PyQt5, которая предоставляет возможность создавать графические интерфейсы для пользователя. Она предоставляет объектно-ориентированные решения, которые включают в себя логическую иерархию между объектами, имеет понятную структуру наследования. Данное пакетное решение является бесплатным, распространяется по лицензии GPL, LGPL. Для проектирования интерфейса использовался QtDesigner. Данный программное обеспечение входит в набор разработчика QT. Служит для разметки интерфейса в формате XML. Формат преобразуется в питоновский класс интерфейса с помощью специального парсера pyuic5.

На рисунках 3.1 - 3.3 представлен интерфейс разрабатываемого ПО.

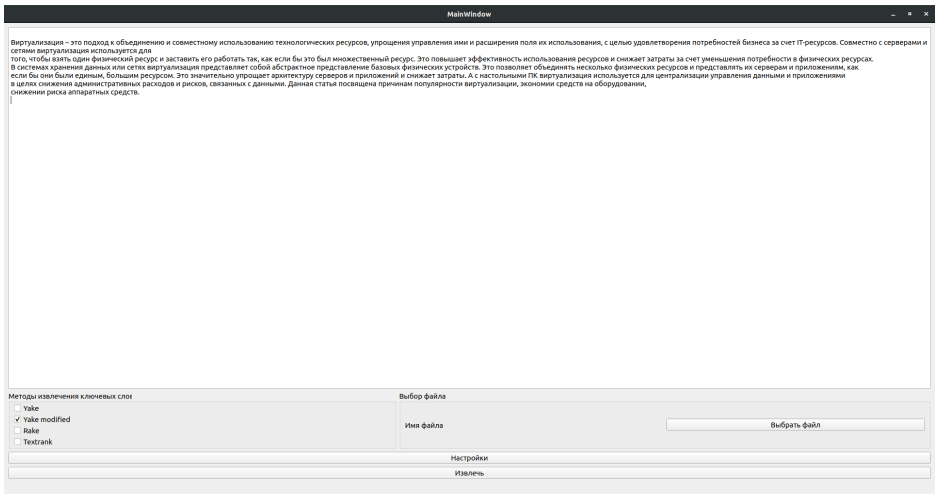


Рисунок 3.1 – Главное окно

Пользовательский интерфейс состоит из нескольких окон. На рисунке 3.1 представлено главное окно программы. В данном окне пользователь может указать фрагмент текста, выбрать файл формата pdf для дальнейшего извлечение

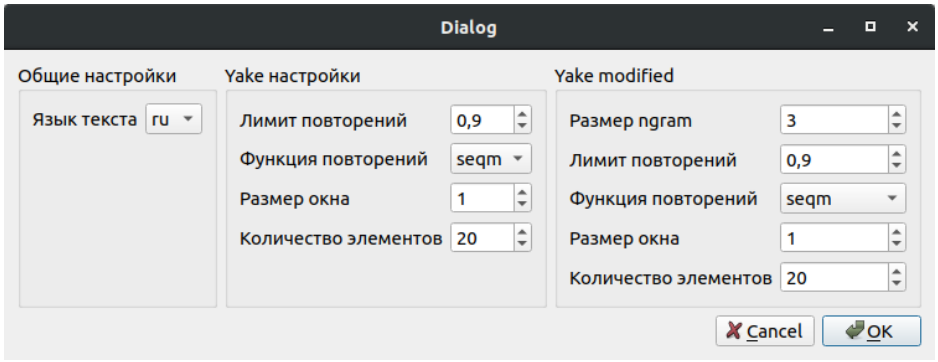


Рисунок 3.2 – Окно настроек методов

```
Документ работы: Результат из поля ввода
Метод: yakemodified
совместному использованию технологических, целью удовлетворения потребностей, удовлетворения потребностей
 бизнеса, использованию технологических ресурсов, подход к объединению, объединению и совместному,
 совместному использованию, использованию технологических, расширения поля, целью удовлетворения,
 удовлетворения потребностей, потребностей бизнеса, счет IT-ресурсов, бизнеса за счет, технологических
 ресурсов, упрощения управления, эффективность использования ресурсов, сетями виртуализация используется,
  виртуализация используется, Виртуализация
```

Рисунок 3.3 – Результат извлечения ключевых слов

текстовой информации, выбрать алгоритмы, открыть окно с параметрами методов, изображенное на рисунке 3.2 и запустить процесс извлечения ключевых слов.

На вход ожидаются:

1. параметры методов;
2. документ формата pdf или тест

На выходе получем список кортежей, состоящий из ключевых слов и оценок. Результат работы алгоритма, отображен на рисунке **??** и представляет собой список извлеченных ключевых слов.

3.6 Среда разработки

Интегрированная среда разработки или IDE (Integrated Development Environment) - специальный программный комплекс, предназначенный для полного цикла написания и тестирования программ на определенном языке.

Интегрированная среда разработки облегчает работу, предоставляя программистам средства для разработки программного обеспечения, такие как

редактор исходного кода, средства автоматизации сборки и отладчик. IDE облегчает визуальное представление файлов и делает его более понятным для пользователя.

Средой разработки для разработки ПО была выбрана IDE PyCharm от компании JetBrains, специализирующейся на производстве инструментов для профессиональной разработки программного обеспечения. Данная среда была выбрана из за следующих удобств и преимуществ:

1. наличие полноценного отладчика как для кода так и для тестов;
2. встроенная подсветка синтаксиса;
3. встроенный терминал;
4. интеграция с системой контроля версий (VCS) git;
5. поддержка множественных конфигураций запуска;
6. встроенный анализатор классов;

3.7 Система контроля версий

Во время процесса разработки мною была использованна система контроля версий Git (https://git-scm.com). Система контроля версий с помощью репозиториев решает проблемы с переносом программного кода на другие устройства, его резервным копированием, а так же дает возможность разделения версий продукта во время разработки, что позволяет при внесении изменений или модификациях всегда иметь рабочею версию проекта.

3.8 Вывод

В результате выполнения данного раздела был выбран язык программирования и инструменты в виде библиотек и фреймворков, необходимых для реализации разрабатываемого программного обеспечения. Проведен детальной обзор разработанного пользовательского интерфейса с применением QT5. Описаны входные и выходные данные разрабатываемого метода. Обоснован выбор интегрированной среды разработки IDE и системы контроля версий VCS

4 Исследовательский раздел

В данном разделе произведен ряд экспериментов с полученным, в ходе написания проекта, программным обеспечением. Для проведения серии экспериментов необходимо подготовить набор тестовых входных данных, представляющих собой тридцать электронных документов на русском языке, формата PDF. Все работы участвующие в эксперименте взяты с сайта cyberleninka.ru Для тестов отбираются тексты с указанными ключевыми словами. Для удобства анализа, многокомпонентные ключевые слова из документов разбиваются на отдельные термины.

Для обеспечения качества проводимых экспериментов необходимо указать критерии по которым будет производиться отбор документов:

1. документ должен содержать в себе текст, а не отсканированные изображения страниц ранее опубликованных работ, поскольку это приводит к невозможности прочтения документа;

2. информация, содержащаяся в документе должна быть целостной, то есть принадлежать одной работе.

4.1 Исследование характеристик метода

В процессе данного эксперимента ставится задача сравнить модифицированный метод с аналогами, способными на работу с документами на русском языке. Для этой цели были выбраны следующие алгоритмы:

1. Rake;

2. Textrank.

В качестве тестовых данных использовались тридцать ранее отобранных документов с ключевыми словами. Исследуется процент от ключевых слов полученных из метода, попавших в пересечение с словами выделенными авторами, так же изучется процент не попавших в пересечение

Для методов были выбраны следующие настройки, отображенные на рисунке 4.1

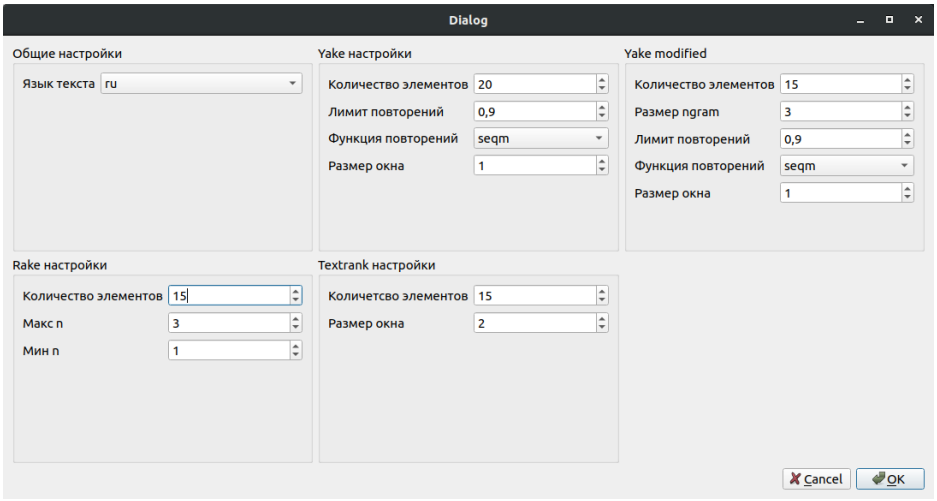


Рисунок 4.1 – Настройки методов

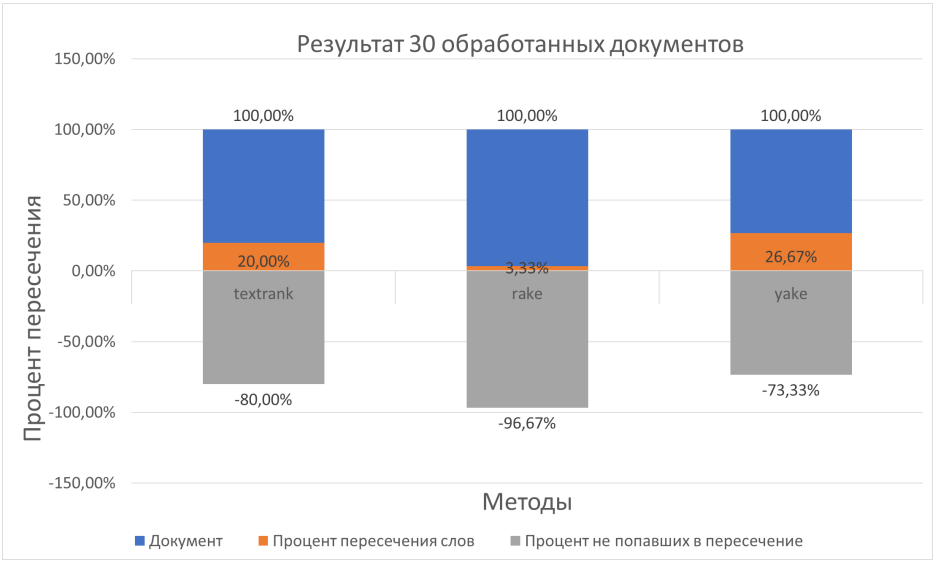


Рисунок 4.2 – Результат обработки 30 документов

По завершению измерений результативности были получены следующие результаты, отображенные в таблице на странице 47 Результатом 30 обработок

	Yake (mod)	Textrank	Rake
Средний % пересечения	26.67%	20%	3.33%
Средний % не попавших в пересечение	73.33%	80%	96.67%

Таблица 4.1 – Результат сравнения методов

документа является средний процент пересечения ключевых слов у метода Yake составляет 26.67% в то время когда у Textrank 20 % у Rake 3.33Процент ключевых

слов не попавших в пересечении у метода Yake составляет 73.33% у Textrank 80% а у метода rake составил 96.67%

4.2 Исследование влияния размерности N-грамм на работу метода

Для проведения данного исследования был выбран документ: "Идентификация личности по фрактальной размерности отпечатков пальцев и системы контроля и управления доступом"[17].

```
Документ работы: identifikatsiya-lichnosti-po-fraktalnoy-razmernosti-otpechatkov-paltsev-i-sistemy
  -kontrolya-i-upravleniya-dostupom.pdf
Метод: yakemodified
размерности, личности, пальцев, Dcp, фрактальной, отпечатков, идентификации, значение, пользователь,
  системы, распознавания, For, СКУД, среднее, биометрические, log, Доклады, число, Lmax, часть

Документ работы: identifikatsiya-lichnosti-po-fraktalnoy-razmernosti-otpechatkov-paltsev-i-sistemy
  -kontrolya-i-upravleniya-dostupom.pdf
Метод: yakemodified
фрактальной размерности, размерности, личности, отпечатков пальцев, размерности отпечатков, идентификации
  личности, пальцев, распознавания личности, Dcp, фрактальной, отпечатков, идентификации, значение,
  пользователь, системы, распознавания, For, СКУД, значение фрактальной, Доклады ТУСУРа

Документ работы: identifikatsiya-lichnosti-po-fraktalnoy-razmernosti-otpechatkov-paltsev-i-sistemy
  -kontrolya-i-upravleniya-dostupom.pdf
Метод: yakemodified
гребней и впадин, размерности отпечатков пальцев, фрактальной размерности отпечатков, размерности,
  личности, отпечатков пальцев, идентификации личности, пальцев, распознавания личности, Dcp, фрактальной,
  отпечатков, идентификации, значение фрактальной размерности, значение, пользователь, системы,
  распознавания, For, размерности Минковского
```

Рисунок 4.3 – Результа извлечения ключевых слов при N = 1 до N = 3

В качестве опорного слова возьмем "пальцев". При N = 2 данное слово преобразуется в словосочетание "отпечатков пальцев". При N = 3 получаем выражением "размерности отпечатков пальцев". На основе этого можно сделать вывод, что метод пригоден для извлечения многокомпонентных ключевых слов.

4.3 Вывод

В результате проведенной исследовательской работы над разработанным решением было установлено, что все требования, поставленные к алгоритму, соблюдены. Метод способен на извлечение многокомпонентных ключевых слов из документов на русском языке.

ЗАКЛЮЧЕНИЕ

По итогу выполнения работы была выполнена цель по разработке метода автоматического извлечению ключевых слов и словосочетаний из документов на русском языке. Помимо выше сказанного были проведены исследования, продемонстрировавшие пригодность метода по работе с документами на кириллице.

В процессе выполнения работы:

1. проанализирована предметная область и произведена классификация существующих методов по извлечению КС;
2. разработана архитектура программного обеспечения;
3. проведена модификация метода и разработаны модули;
4. произведено исследование получившегося ПО.

Из достоинств полученного метода можно выделить следующее:

1. не требует обучения и наличия корпуса текстов;
2. возможность извлечения n-компонентных ключевых слов;
3. не использует тезаурусы.

Из возможных путей развития стоит отметить:

1. добавление в предварительную обработку процесса преобразование слов к начальной форме;
2. добавление автоматического определения языка;
3. обучение алгоритма определению синонимические терминов;
4. обучение алгоритма автоматическому определению языка текста.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Кульмин В. В.* Информационная революция [Электронный ресурс] // Большая российская газета. —. — URL: https://bigenc.ru/technology_and_technique/text/2015889 (дата обращения: 21.1.2022).
2. Textual Analysis: A Beginner's Guid [Электронный ресурс]. —. — URL: http://www1.cs.columbia.edu/~sbenus/Teaching/APTD/McKee_Ch1.pdf (дата обращения: 21.1.2022).
3. *Шереметьева С. О.*, *Осмини П. Г.* Методы и модели автоматического извлечения ключевых слов [Электронный ресурс]. —. — URL: https://cyberleninka.ru/article/n/metody-i-modeli-avtomaticheskogo-izvlecheniya-klyuchevyh-slov (дата обращения: 21.1.2022).
4. *Соколов А. Н.* Внутренняя речь и мышление [Электронный ресурс]. —. — URL: https://search.rsl.ru/ru/record/01008431174 (дата обращения: 22.1.2022).
5. *В. В. Н.*, *Иванов В. В.* Современные методы автоматизированного извлечения ключевых слов из текста [Электронный ресурс]. —. — URL: https://www.elibrary.ru/item.asp?id=27036419 (дата обращения: 25.1.2022).
6. Automatic keyphrases extraction based on NLP and statistical methods [Электронный ресурс]. —. — URL: https://search.rsl.ru/ru/record/01008431174 (дата обращения: 22.1.2022).
7. *Paishikar G.* Keyword extraction from a single document using centrality measures [Электронный ресурс]. —. — URL: https://www.researchgate.net/publication/221205058_Keyword_Extraction_from_a_Single_Document_Using_Centrality_Measures (дата обращения: 23.1.2022).
8. *Michael W.* Berry Text Mining Application and Theory [Текст]. —. — (дата обращения: 23.1.2022).
9. *Michael W.* YAKE! Keyword extraction from single documents using multiple local features [Электронный ресурс]. —. — (дата обращения: 25.1.2022).

10. *Строганов Ю.*, *Бородин Д.* Исследование метода выделения однословных терминов в тематических текстах [Электронный ресурс]. —. — URL: https://cyberleninka.ru/article/n/issledovanie-metoda-vydeleniya-odnoslovnyh-terminov-v-tematicheskih-tekstah (дата обращения: 27.1.2022).

11. *Rose S.*, *Engel D.*, *Cramer N.* Automatic Keyword Extraction from individual Document [Электронный ресурс]. —. — URL: https://www.researchgate.net/publication/227988510_Automatic_Keyword_Extraction_from_Individual_Documents (дата обращения: 27.1.2022).

12. *Luhn H. P.* The Automatic Creation of Literature Abstracts [Электронный ресурс]. — 1958. — Апр. — URL: https://ieeexplore.ieee.org/document/5392672 (дата обращения: 2.2.2022).

13. *Machado D.*, *Barbosa T.*, *Martins B.* Universal Mobile Information Retrieval [Электронный ресурс]. —. — URL: https://link.springer.com/chapter/10.1007/978-3-642-02710-9_38 (дата обращения: 10.2.2022).

14. *Ершов М.* МНОГОКОМПОНЕНТНЫЕ ТЕРМИНЫ СФЕРЫ ТЕПЛОЭНЕРГЕТИКИ [Электронный ресурс]. —. — URL: https://elar.urfu.ru/bitstream/10995/60427/1/978-5-8295-0572-1_2018_01_50.pdf (дата обращения: 26.4.2022).

15. *Гудков В. Ю.*, *Ф. Г. Е.* N-граммы в лингвистике [Электронный ресурс]. —. — URL: http://lab314.brsu.by/kmp-lite/kmp2/Translation/MT/MT-Corpus/n-grammy-v-lingvistike.pdf (дата обращения: 26.3.2022).

16. *Ершов М.* Portable Document Format [Электронный ресурс]. —. — URL: https://helpx.adobe.com/ru/incopy/using/pdf.html (дата обращения: 12.5.2022).

17. *Трифонов Д. И.* Идентификация личности по фрактальной размерности отпечатков пальцев и системы контроля и управления доступом [Электронный ресурс]. —. — URL: https://cyberleninka.ru/article/n/identifikatsiya-lichnosti-po-fraktalnoy-razmernosti-otpechatkov-

paltsev-i-sistemy-kontrolya-i-upravleniya-dostupom (дата обращения: 22.5.2022).

ПРИЛОЖЕНИЕ А

Листинг А.1 – Код модифицированного метода yake

```
# -*- coding: utf-8 -*-

"""Main module."""

import string
import os
import jellyfish
from .Levenshtein import Levenshtein

from .datarepresentation import DataCore


class YakeModified(object):

    def __init__(self, lan="ru", n=3, dedupLim=0.9,
       dedupFunc='seqm', windowsSize=1, top=20, features=None,
                 stopwords=None):
        self.lan = lan

        dir_path = os.path.dirname(os.path.realpath(__file__))

        local_path = os.path.join("StopwordsList",
           "stopwords_%s.txt" % lan[:2].lower())

        if os.path.exists(os.path.join(dir_path, local_path)) ==
           False:
            local_path = os.path.join("StopwordsList",
               "stopwords_noLang.txt")

        resource_path = os.path.join(dir_path, local_path)

        if stopwords is None:
            try:
```

```
                with open(resource_path, encoding='utf-8') as
                    stop_fil:
                     self.stopword_set =
                        set(stop_fil.read().lower().split("\n"))
            except:
                print('Warning, read stopword list as ISO-8859-1')
                with open(resource_path, encoding='ISO-8859-1') as
                    stop_fil:
                     self.stopword_set =
                        set(stop_fil.read().lower().split("\n"))
        else:
            self.stopword_set = set(stopwords)

        self.n = n
        self.top = top
        self.dedupLim = dedupLim
        self.features = features
        self.windowsSize = windowsSize
        if dedupFunc == 'jaro_winkler' or dedupFunc == 'jaro':
            self.dedu_function = self.jaro
        elif dedupFunc.lower() == 'sequencematcher' or
           dedupFunc.lower() == 'seqm':
            self.dedu_function = self.seqm
        else:
            self.dedu_function = self.levs

    def jaro(self, cand1, cand2):
        return jellyfish.jaro_winkler(cand1, cand2)

    def levs(self, cand1, cand2):
        return 1. - jellyfish.levenshtein_distance(cand1, cand2) /
           max(len(cand1), len(cand2))

    def seqm(self, cand1, cand2):
        return Levenshtein.ratio(cand1, cand2)

    def extract_keywords(self, text):
        try:
            if not (len(text) > 0):
```

```
            return []

        text = text.replace('\n\t', ' ')
        dc = DataCore(text=text,
           stopword_set=self.stopword_set,
           windowsSize=self.windowsSize, n=self.n)
        dc.build_single_terms_features(features=self.features)
        dc.build_mult_terms_features(features=self.features)
        resultSet = []
        todedup = sorted([cc for cc in dc.candidates.values()
           if cc.isValid()], key=lambda c: c.H)

        if self.dedupLim >= 1.:
            return ([(cand.H, cand.unique_kw) for cand in
               todedup])[:self.top]

        for cand in todedup:
            toadd = True
            for (h, candResult) in resultSet:
                dist = self.dedu_function(cand.unique_kw,
                   candResult.unique_kw)
                if dist > self.dedupLim:
                    toadd = False
                    break
            if toadd:
                resultSet.append((cand.H, cand))
            if len(resultSet) == self.top:
                break

        return [(cand.kw, h) for (h, cand) in resultSet]
    except Exception as e:
        print(f"Warning! Exception: {e} generated by the
           following text: '{text}' ")
        return []
```

Листинг А.2 – Код контролера

```
import sys
from PyQt5 import QtWidgets
```

```
from interface.ui import MainWindow
from interface.app.SettingsWindow import SettingsWindow

import textract
from methods.yake import Yake
from methods.yake import YakeModified
from methods.rake import Rake
from methods.textrank import TextRank

from prettytable import PrettyTable

class MainWindow(QtWidgets.QMainWindow, MainWindow.Ui_MainWindow):
    def __init__(self):
        super().__init__()

        # Устанавливаем внутренний интерфейс
        self.setupUi(self)

        # Установка дополнительных окон
        self.settings_window = SettingsWindow()

        # Установка связей
        self.select_file_button.clicked.connect(self.browse_files)
        self.ExtractButton.clicked.connect(self.run)

        # Дополнительные внутрнение переменые
        self.files = None

        self.methods_checkbox = {
            'yake': self.yake_checkbox,
            'yakemodified': self.yakemodified_checkbox,
            'rake': self.rake_checkbox,
            'textrank': self.textrank_checkbox,
        }
        self.methods = {
            'yake': Yake,
            'yakemodified': YakeModified,
            'rake': Rake,
            'textrank': TextRank
```

```
        }

    def get_settings(self):
        return self.settings_window.get_settings()

    def get_methods(self):
        return {key: self.methods[key] for key, value in
           self.methods_checkbox.items() if value.isChecked()}

    def setup_methods(self, methods_class, settings):
        methods = {}
        for key, method_class in methods_class.items():
            methods[key] = method_class(**settings[key])
        return methods

    def extract_keywords(self, text, methods):
        result = {}
        for method_name, method in methods.items():
            result[method_name] = method.extract_keywords(text)
        return result

    def display_result(self, result):
        for key, method_data in result.items():
            table = PrettyTable(field_names=['keyword', 'score'])
            for result in method_data:
                table.add_row(result)

            print(table)

    def simpl_result(self, result):
        for key, method_data in result.items():
            print(f"Метод: {key}")
            if key == "rake":
                result_text = ', '.join([str(info[1]) for info in
                   method_data])
            else:
                result_text = ', '.join([str(info[0]) for info in
                   method_data])
            print(result_text)
```

```
            print("\n")


    def browse_files(self):
        files = QtWidgets.QFileDialog.getOpenFileNames(self,
           'Select files', directory='/www/')[0]
        if len(files) > 1:
            self.file_name_label.setText(f'Выбрано {len(files)}')
        else:
            self.file_name_label.setText(files[0].rsplit('/',
               1)[1])
        self.files = files



    def get_text(self):
        for file in self.files:
            yield textract.process(file).decode('utf-8'),
               file.rsplit('/', 1)[1]

    def show_settings(self):
        self.settings_window.show()

    def run(self):
        settings = self.get_settings()
        methods = self.get_methods()
        methods = self.setup_methods(methods, settings)
        for text, filename in self.get_text():
            result = self.extract_keywords(text, methods)
            print(f"Документ работы: {filename}")
            self.simpl_result(result)
            print("\n\n")
```