МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

 **«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра информационных технологий**

**КУРСОВАЯ РАБОТА**

**ПРОТОТИП ПРОГРАММЫ РЕФЕРИРОВАНИЯ ДОКУМЕНТОВ**

Работу выполнила\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_А.А. Гаврилова

 (подпись)

Направление 02.03.03 «Математическое обеспечение и администрирование информационных систем»

Направленность (профиль) «Технология программирования»

Научный руководитель

канд. пед. наук, доц.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Н.Ю.Добровольская

 (подпись)

Нормоконтролер

ст.преп.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_А.В.Харченко

 (подпись)

Краснодар

2018

**РЕФЕРАТ**

Курсовая работа 25 ст., 5 рис., 8 источников.

ЛЕКСИКО-СЕМАНТИЧЕСКИЙ АНАЛИЗ, РЕФЕРИРОВАНИЕ ДОКУМЕНТОВ, СТЕММИНГ, СТЕММЕР ПОРТЕРА, WORD2VEC

Цель работы: изучить методы лексического и семантического анализа текста, и на основе этих инструментов разработать прототип программы реферирования текстов.

В ходе курсовой работы изучены основные методы, применяющиеся для решения задачи автоматического реферирования, а именно, алгоритмы стемминга, предобработки текстов, в том числе приведены критерии оценивания программ для составления аннотаций. Кроме того, были изучены основные аспекты работы с библиотекой gensim и технологией Word2Vec и с другими инструментами синтаксического и семантического анализа текстов.

 В ходе проделанной работы был описан алгоритм решения задачи автоматического аннотирования и по нему разработана программа, применяющая изученные методы.

СОДЕРЖАНИЕ

Введение……………………………………………………………………..….…4

1 Реферирование текстов………...……………………………………………….5

1.1 История алгоритмов решения задачи аннотирования. ....….............…....5

1.2 Классификация методов решения задачи реферирования………...…...8

1.3 Алгоритмы контекстно-зависимого аннотирования………...……........ 11

1.3.1 Математическая модель алгоритма составления КС-аннотаций…………………... ………………………..………....…...……. 11

1.3.2 Требования, предъявляемые к аннотациям…………................…... 12

1.4 Методики оценивания программ аннотирования....…. ….………....…. 13

2 Описание алгоритма прототипа программы составления аннотаций......….15

2.1 Препроцессинг исходных данных….……………….………………...… 17

2.2 Реализация алгоритма стемминга. Тест Портера...…….……...……......18

2.3 Обучение модели Word2Vec…………………….……….……...………. 19

2.4 Формирование и обрезка аннотации………..…………..………..……... 21

3 Результаты работы и возможные модификации прототипа ………………..22

Заключение……………………………………………………………………….24

Список использованных источников…………………………………………...25

**ВВЕДЕНИЕ**

В настоящее время развитие технологий и Интернета дарит человечеству возможность доступа к огромному количеству информационных ресурсов, что дает людям больше возможностей для самообразования. Наряду с этим в связи с огромными объемами электронных документов возрастает потребность в обработке неструктурированной текстовой информации, повышении качества и эффективности уже использующихся методов.

Ранее с такими трудностями людям приходилось справляться самим, например, писать конспекты, рефераты, сокращать написанный текст. Говоря об официальной документации, ее аннотированием занимались только специалисты гуманитарного профиля, лингвисты и филологи. Но с появлением компьютеров и развитием их вычислительной точности, решением такой насущной проблемы стала заниматься такая наука как компьютерная лингвистика. Одной из ее задач является автоматическое реферирование и аннотирование текста – процесс сокращения текстового документа с использованием программного обеспечения, с целью выделения главной информации текста. Методы, применяющиеся для ее решения, относятся к таким развивающимся областям, как machine learning и data mining.

В курсовой работе предлагается разработать прототип программы автоматического реферирования.

**1 Реферирование документов**

**1.1 История алгоритмов решения задачи аннотирования**

Рефераты, сокращения, конспекты используются с нами с самого появления письменности, так что проблема реферирования и сокращения тестов была очень давно и сегодня потеряла свою актуальность. В нашу эпоху технологий и информации найти нужную информацию без поисковой системы не представляется возможным, а для их быстрой и эффективной работы разработчикам приходится добавлять все новые и новые инструменты ускорения и оптимизации поиска. Одним из таких инструментов является механизм составления аннотаций.

Первые подходы решения задачи автоматического реферирования были разработаны в конце 50-х годов XX века Луном и дополнены в следующее десятилетие Эдмундсоном в 1969 [9]. Лун и Эдмундсон предложили генерировать аннотации из фрагментов, характеристиками которых выступали частота встречаемости, наличие слов из заголовка и слов-маркеров, расположение предложения в тексте. Однако их подход никак не учитывал структуру исходного документа.

В 1997 году Д. Марк в своей работе «The Rhetorical Parsing, Summarization, and Generation of Natural Language Texts» продолжил изучения этого вопроса. Он предложил подход с использованием RST-деревьев (рис. 1), представляющих собой деревья разбора, построенных на основе риторического статуса фрагментов, например «основной смысл», «обстоятельство», «причина» и т.д. В таком дереве наиболее значимые с точки зрения смысла фрагменты расположены в листьях, и степень значимости убывает с расстоянием до листа. Данный подход позволяет гибко настраивать параметры аннотации (например, делать ее более лаконичной или информативной), однако весьма затратен в смысле вычислительных ресурсов, поэтому может быть неэффективен на больших объемах информации.



Рисунок 1 – RST-дерево

Формальное деление фрагментов текста на значимые и незначимые сводится к задаче бинарной классификации машинного обучения. По этой причине возникла идея использования наивного байесовского классификатора – это такой простой вероятностный классификатор, основанный на применении теоремы Байеса со строгими (наивными) предположениями о независимости (формула 1):

 $P\left(x\right)=\frac{P\left(c\right)P(c)}{P(x)}$ (1)

В качестве обучающей выборки был использован корпус научных статей с аннотациями. В качестве определяющих параметров были использованы следующие:

1. длина предложения,
2. содержание определенных речевых конструкций,
3. положение в параграфе (начало, середина, конец),
4. содержание ключевых (высокочастотных) терминов,
5. содержание акронимов.

В дальнейшем были придуманы более интересные и развитые подходы к решению данной задачи. Например, алгоритм, названный Grasshopper (Кузнечик), базируется на теории поглощающих случайных блужданий на графе, развитой самими авторами. Он основан на идеях, принадлежащих классу машинного обучения без учителя (unsupervised machine learning), и призван свести к минимуму избыточность содержания аннотации.

Последние разработки в области автоматического аннотирования ежегодно представляются на конференции Ассоциации Вычислительной Лингвистики (Association for Computational Linguistics), потому что с каждым годом появляются все новые методы и разработки, связанные с решением задачи автоматического реферирования.

Несмотря на большую актуальность и необходимость систем автоматического реферирования на практике, на пути разработки таких продуктов возникают трудности.

Одной из проблем в применении извлекающих методов – это требование сильного сжатия – объем реферата или аннотации зачастую должен составлять всего 5-10% исходного текста. Очень сложно уложить все главные аспекты крупного текста в такой маленький объем. Кроме этого, проблемой является недостаток различных лингвистических ресурсов (толковые, лексические и частотные словари, грамматики, тезаурус) и сложность автоматического создания текстов на естественном языке. К сожалению, на сегодняшний день очень мало подобных материалов в открытом доступе, а ведь считается, что большая часть успеха в построении моделей машинного обучения – это качественные и хорошо подготовленные данные. Помимо этого, большинство алгоритмов, построенных на машинном обучении, требуют больших вычислительных ресурсов.

 **1.2 Классификация методов решения задачи реферирования**

 Методология, применяемая для реферирования документов, очень разнообразна (рис. 2), в ходе разработки могут быть задействованы как и лексические, подходы, так и синтаксические, учитывающие порядок слов в тексте и ранжирующие по этому параметру значимость слов, наряду с ними имеют место и семантические методы, но чаще всего используется несколько подходов сразу.

По типу получаемого реферата выделяются следующие подходы [6]:

1. экстракция (излечение предложений, квазиреферирование),
2. абстракция (извлечение содержания, Content Extraction).

Экстракция – извлечение из исходного текста наиболее важных и существенных информационных блоков (абзацев, предложений). Наиболее значимые предложения отбираются на основе позиционных (местоположение в документе, разделе, абзаце с определением типа выделенного элемента, например, заголовок, заключение или вступление) и тематических критериев. Модификацией такого подхода является использование особых маркеров, фиксирующих значимость предложений (например, фразы типа «в итоге», «важно», «наше исследование показало» и др.). Методы данного блока отличает наличие оценочной функции (Scoring Function) важности информационного блока или предложения.

Такой подход применял в своей работе Г. Эдмундсон, вес предложений он вычислял по формуле 2:

 $W= a\_{1}\*W\_{stat}+a\_{2}\*W\_{key}+a\_{3}\*W\_{location}$(2)

Другим подходом с противоположной задумкой является абстракция – генерация реферата с порождением нового текста, содержательно обобщающего первичный документ. Такой подход характеризуется такими этапами, как:

1. анализ исходного текста с генерацией внутреннего представления,
2. семантическое сжатие внутреннего представления,
3. синтез нового текста.

Для реализации данных методов необходимы мощные вычислительные ресурсы для систем обработки естественных языков (NLP –

natural language processing), в том числе грамматики и словари для синтаксического разбора.

Не менее важной является классификация по уровню анализа исходного текста. Выделяют следующие подходы[1]:

1. классические методы – в них применяется модель линейных весовых коэффициентов;
2. статистические методы – основаны на использовании статистических параметров, прежде всего на частоте встречаемости слов в тексте;
3. позиционные методы – опираются на предположение о том, что информативность текстового блока находится в зависимости от его позиции (места) в тексте;
4. индикаторные методы – основаны на идентификации фраз с помощью индексации их специальными словами – маркерами.

Также в отдельную группу выделяются методы, основанные на машинном обучении – они своеобразное развитие статистического подхода. Тут также есть много различных подходов, это и модели, основанные на тематических признаках (топ-10 частотных слов документа, имя + фамилия, слова заголовка), или могут быть рассмотрены структурные признаки (вопросительные предложения, прямая речь, союзы, синтаксические сложности предложений). Для построения таких моделей в большинстве случаев используются нейронные сети, пример такой сети на рисунке 2.



Рисунок 2 – Нейронная сеть

**1.3 Алгоритмы контекстно-зависимого аннотирования**

Особенностью контектно-зависимого аннотирования является то, что при ее составлении играет большую роль не только сам текст, но и слова запроса и близкие с ними по смыслу, поэтому такие алгоритмы повсеместно используются в любых поисковых системах.

В общем случае схема такого алгоритма выглядит так:

1. фрагментация. Используется метода скользящего окна: текст обрабатываемого документа разбивается на фрагменты, которые представлены либо предложениями, либо их связной частью, если длина предложения превосходит заданную длину аннотации. В первом случае фрагменты, очевидно, не пересекаются, во втором же возможны наложения, которые учитываются на завершающем этапе алгоритма;
2. оценивание фрагментов при помощи определенных характеристик;
3. ранжирование фрагментов по убыванию весов;
4. построение аннотации путём последовательного извлечения элементов отсортированного списка фрагментов, пока выполняется ограничение на длину текста, исключая добавление пересекающихся фрагментов.

**1.3.1 Математическая модель составления КС-аннотаций**

Лексемой называется слово как абстрактная единица морфологического анализа [2]. В одну лексему объединяются различие словоформы одного и того же слова. В словарях каждая лексема представляется одной и словоформ, которую называют формой или леммой, а сам процесс сведения называют лемматизацией.

В общем виде задача построения аннотации к тексту может быть записана следующим образом. Входные данные:

1. документ, для которого строится аннотация, $D=\{d\_{i}\}$ – множество лексем;
2. запрос пользователя, $Q=\{q\_{i}\}$ – множество лексем;
3. коллекция документов для обучения $Coll=\{a\_{i}\}$;

На выходе получаем аннотацию (множество лексем) $A=A\left(D\right)=\{a\_{i}\}$.

**1.3.2 Требования, предъявляемые к аннотациям**

Формируемые системой аннотации должны удовлетворять ряду требований. Их можно разделить на требования, которые выдвигает пользователь, и требования, которые накладываются реализацией системы. Можно назвать следующие требования, выдвигаемые пользователем [6]:

1) аннотация должна давать представление о том, какую информацию содержит документ о предмете запроса;

 2) быть достаточно короткой, чтобы ее анализ пользователем не занимал много времени. Человек должен иметь возможность охватить сформированный текст «одним взглядом»;

 3) в случае большого документа, она должна давать представление о том, какие части документа несут релевантную информацию.

С точки зрения реализации к алгоритму формирования аннотации предъявляются следующие требования:

1) конечная вычислительная трудоемкость. Обработка для создания аннотаций является, с точки зрения поисковой системы, вспомогательной задачей и не должна приводить к заметному замедлению системы;

2) максимальное использование информации полученной при поиске на этапах отбора и взвешивания документов;

3) приемлемый требуемый объем оперативной памяти.

**1.4 Методики оценивания программ аннотирования**

 Для повышения эффективности разрабатываемых программ составления аннотаций необходима система их оценивания. Так, можно выделить два основных подхода для оценки их качества, — внутренний (intrinsic) и внешний (extrinsic) [4].

1. Внутренние методы оценки

Первый подход нацелен на решение таких вопросов как:

– являются ли предложения аннотации грамматически правильными;

– является ли текст аннотации связным;

– содержит ли аннотация все основные обсуждаемые темы исходного документа.

 Для проведения оценивания могут использоваться независимые тестирования, также сравнения независимых экспертов с аннотациями, порожденными людьми. Наряду с этим используется сравнение с такими аннотациями в автоматическом режиме, без человеческого участия. Для этого применяются следующие метрики:

 – точность (сколько общих предложений содержат аннотации),

 – ранг («значимость» предложений и отдельных слов),

 – полезность («информативность» аннотации),

 – содержательность (похожесть извлеченных словарей).

1. Внешние методы оценки

 В отличие от внутренних методов внешние предполагают оценивание аннотации с точки зрения решения задач, поставленных перед нею, то есть косвенное оценивание. Критерии могут быть весьма разнообразны:

 ­ – релевантность, то есть насколько аннотация помогает пользователю найти желаемое;

 – эффективность с точки зрения системы, в которую встроен модуль аннотирования и др.;

 – ­понимание при прочтении (reading comprehension) аннотации, то есть возможность ответить на основные вопросы по тексту на основе аннотации.

 Выбор того, какой метод оценивания будет эффективнее, напрямую зависит от алгоритма решения задачи, от потребностей системных архитекторов и нужд конечного пользователя программы.

**2 Описание алгоритма прототипа программы составления аннотаций**

Для разработки прототипа программы реферирования выбран алгоритм контекстно-зависимого аннотирования [7]. Он учитывает вхождения слов, очень близких по смыслу к словам запроса (в данном случае в роли запроса берется название документа, ведь часто именно в нем отражается основная суть, но при желании мы можем просто заменить содержимое переменной на любую строку, заданную пользователем).

Алгоритм составления аннотации включает в себя следующие шаги:

1. предварительная подготовка документа и слов запроса (получения контента релевантных элементов);
2. разбиение полученного контента на пассажи. Пассаж – это фрагмент текста, содержащий законченную мысль;
3. формирование кластеров из N пассажей;
4. расчет параметров ранжирования пассажей по взвешенной оценке и определение кластера с наибольшим рангом;
5. ранжирование кластеров по взвешенной оценке и определение кластера пассажей самым большим рангом;
6. запуск алгоритма обрезки аннотации до n ≤ k символов (в данном случае k = 300) и формирование аннотации. Блок-схема алгоритма обрезки аннотации представлена на рисунке 4.

Алгоритм выбора пассажей для аннотации реализован следующим образом:

1. все пассажи исходной страницы группируются в кластеры по N пассажей в кластере с шагом в 1 пассаж. В настоящем алгоритме значение N = 4. То есть в первый кластер входят по порядку следования в тексте 1, 2, 3 и 4 предложения, во второй кластер – 2,3,4,5 предложения и т.д.;

 2) для каждого пассажа с номером i рассчитываются взвешенный

параметр ранжирования $R\_{i}$ по формуле 3:

 $ R\_{i}=\frac{4\*\frac{U\_{kwp\_{i}}}{U\_{wq}}+1,8\*\frac{T\_{r\_{i}}}{U\_{wq}}+\frac{T\_{s\_{i}}}{U\_{wq}}+0,1\*D\_{s\_{i}}}{4+1.8+1+0.1}$ (3)

где

$U\_{wq}$ – количество уникальных слов нашего заголовка, исключая стоп-слова (предлоги, союзы);

 $U\_{kwp\_{i}}$ – количество уникальных слов заголовка, стоящие в той же словоформе, что и в названии документа;

$T\_{r\_{i}}$ – количество уникальных слов-переходов, найденных в каждом пассаже для всех слов запроса (не учитываем слова из $U\_{kwp\_{i}}$). Для этого был применен алгоритм стемминга – то есть учитываются только те слова, основа которых совпадает со стемами (основами) слов нашего заголовка;

$T\_{s\_{i}}$ – количество вхождений ближайших синонимов слов запроса (также исключаем слова, отобранные вышеизложенными параметрами). Для этого предварительно была обучена модель Word2Vec на данных русской Wikipedia и сгенерированы для каждого слова запроса топ-10 ближайших по смыслу слов;

$D\_{s\_{i}}$ – наличие хотя бы одного знака «тире» или «дефис» (параметр равен 1, если в пассаже есть хотя бы один знак, и равняется 0, если таких знаков не обнаружено).

3) далее производится кластеризация пассажей с параметром N = 4 (можно поменять, если будет необходимо) – то есть группируем пассажи во всевозможные непересекающиеся группы. Затем для каждого кластера рассчитывается сумма $S\_{i}$ по формуле 4:

 $S\_{i}≔ \sum\_{i=1}^{N}R\_{i}$ (4)

4) выбирается кластер пассажей с максимальным значением суммы;

5) запускается процесс образки аннотации до 300 символов и формируется аннотация.

**2.1 Препроцессинг исходных данных**

Процедура подготовки данных к анализу, в процессе которой их приводят в соответствие с требованиями, определяемыми спецификой задачи, ­– носит название предобработки данных (Data Preprocessing).

Предварительная обработка данных является важным шагом в процессе интеллектуального анализа данных, потому что плохо обработанные данные часто приводят к недопустимым значениям и невозможным комбинациям данных. При анализе данных, не имеющих защиты от подобных проблем, можно прийти к неверным выводам и получить результаты плохого качества.

Обычно предобработку можно разделить на два этапа:

– очистка – включает обработку дубликатов, противоречий, очистку данных от разного шума, на этом же этапе происходит восстановление нарушенной структуры, полноты и целостности данных;

– оптимизация – включает снижение размерности входных данных, выявление и исключение незначащих признаков.

Для того, чтобы обработать текст и разбить его на пассажи, сначала следует разбить исходный текст на предложения. Первое, что возникает в голову ­– просто разбить его функцией split() по пробелам, но, как показывает практика, зачастую у предложений куда более сложная структура, включаются различные вставки, формулы, переносы строки, поэтому используем готовый инструмент токенизации по предложениям библиотеки nltk – sent\_tokenize(text).

Следующее, что мы должны сделать в ходе предобработки ­– разбить наш заголовок на токены и сформировать список ключевых слов, по которым будет производиться расчет параметров ранжирования, будет проще написать для этого функцию, так как она понадобится и для обработки пассажей. Для этого нам нужно проделать следующие действия:

1) применить токенизацию (можно воспользоваться nltk.word\_tokenize);

2) удалить все знаки препинания;

3) очистить наши данные от стоп-слов (слова, не несущие смысловой нагрузки, вводные слова, служебные части речи, например, «что», «это», «как», «быть» и пр.).

**2.2 Реализация алгоритма стемминга. Тест Портера**

.

Стемминг — это процесс нахождения основы слова для заданного исходного слова. Основа слова не обязательно совпадает с морфологическим корнем слова. Существует несколько типов алгоритмов стемминга, которые различаются по отношению производительности, точности, а также как преодолеваются определённые проблемы стемминга.

Есть несколько вариаций реализации данного алгоритма, но мною для расчета параметра $T\_{r\_{i}}$ - поиск слов-переходов, то есть тех же ключевых слов запроса, только в других словоформах, – был выбран алгоритм усечения окончаний [8].

Плюсом такого алгоритма является отсутствие необходимости прописывать справочную таблицу, состоящую из корня и формы, как в некоторых других реализациях алгоритма стемминга. Вместо этого используется свод правил для удаления суффиксов и окончаний с конца слова, то есть все, что нам нужно сделать – описать списки возможных суффиксов и окончаний слов русского языка.

Например, на вход алгоритма попадает слово «бегавшая», происходит разделение слова на составные части по формуле 5:

 $бегавшая=бега+вш+ая$ (5)

 В результате получаем основу «бега», или, по-другому, стем.

Одной из самых известных реализаций алгоритма стемминга является стеммер Портера. Он был написан в 1980 году Мартином Портером и по сей день имеет широкое распространение.

В 2000 году Мартин Портер выпустил официальную свободную реализацию алгоритма, после этого продолжал эту работу в течение нескольких следующих лет, разработав Snowball, фреймворк для создания алгоритмов стемминга, и улучшенных стеммеров английского языка, а также стеммеров для некоторых других языков. В ходе этого времени им был написан особый критерий эффективности алгоритма – Тест Портера, который и сейчас используется для оценивания алгоритмов стемминга.

Вышеизложенный и реализованный алгоритм успешно проходит такое тестирование [8].

**2.3 Обучение модели Word2Vec**

Первоначальная модель алгоритма не включала в себя поиск близких по семантике слов, а ограничивалась точным вхождением и наличием словоформ слова. Но потом возникла идея учитывать синонимы, то есть сгенерировать небольшой словарь близких по значению слов. Для реализации такого подхода был выбран инструмент Word2vec.

Word2vec — это инструмент (набор алгоритмов) для расчета векторных представлений слов, реализует две основные архитектуры — Continuous Bag of Words (CBOW) и Skip-gram [3]. На вход подается корпус текста, а на выходе получается набор векторов слов.

Принцип работы такого инструмента можно описать следующим образом: находятся связи между контекстами слов, говоря более формально, решается задача максимизации косинусной близости между векторами слов (скалярное произведение векторов), которые появляются рядом друг с другом, и минимизации косинусной близости между векторами слов, которые не появляются друг рядом с другом.

В основе word2vec применяется нейросеть прямого распространения (Feed-forward Neural Network). CBOW и Skip-gram— это нейросетевые архитектуры, которые описывают, как именно нейросеть «учится» на данных и «запоминает» представления слов. Принципы у обоих архитектур разные. Принцип работы CBOW — предсказывание слова при данном контексте, а skip-gram наоборот — предсказывается контекст при данном слове.

Для обучения модели word2vec хорошего качества нужен очень большой корпус. В прототипе программы были использованы данные русской Википедии (около 50 K записей). Фильтрация и лемматизация данных не проводилась, так процесс построения очень долгий и затратный.

После загрузки модели генерируем топ-10 близких по семантике слов для каждого ключевого слова запроса, для этого используем m\_most\_simular (рис. 3). Рассчитывая вхождения таких слов по пассажам и вычисляем параметр $T\_{s\_{i}}$.



Рисунок 3 – Результаты работы word2vec на слове «реферат»

**2.4 Формирование и обрезка аннотации**

Алгоритм обрезки аннотации включает в себя следующие шаги:

1. На вход алгоритма поступает кластер пассажей с максимальным значением$ S\_{i}$, содержащая выбранные для аннотации пассажи, для каждого из которых вычислены: взвешенная оценка по параметрам $R\_{i}$, Длина каждого пассажа в символах $L\_{i}$, требуемая длина аннотации k =300 символов;
2. Из пришедших на вход алгоритма пассажей выбирается один пассаж, имеющий максимальную взвешенную оценку $R\_{i}$;
3. Если количество символов уже добавленных в аннотацию пассажей превышает лимит в 300 символов, то пассаж обрезается по целому слову и после него ставится многоточие таким образом, чтобы количество символов обрезанного пассажа не превышало 300. Искомая аннотация построена;
4. Если количество символов уже добавленных в аннотацию пассажей меньше 300, но больше 200 символов, то в аннотацию этот пассаж выводится целиком. Искомая аннотация построена;
5. Если количество символов этого пассажа меньше 200, то анализируются 2 пассажа слева и справа от уже добавленного в аннотацию пассажа. Для обоих пассажей находится их параметр ранжирования, и чей параметр, соответственно больше, тот пассаж мы и добавляем. Обрезаем до 300 символов по целому слову, если не конец предложения, ставим многоточие;
6. Если параметры оказались равными, выбираем пассаж с минимальной длиной;
7. Если длины более 300 символов – обрезаем и аннотация построена, если же нет – назад к пункту 5.

Формирование аннотации происходит на шагах 3,4 алгоритма.

**3 Результаты работы и возможные модификации прототипа**

Разработан прототип программы автоматического аннотирования документов с использованием запроса, заданного пользователем или же просто заголовка. Код программы написан на языке Python 3 с использованием библиотек nltk (для лексико-семантического анализа) и gensim (для написания модели).

Для тестирования алгоритма нами была выбрана статья «Технология автоматизированного реферирования документов юридической направленности».

Разбив текст на пассажи и рассчитав параметры ранжирования, получаем кластер, содержащий основную мысль данной статьи (рисунок 4):



Рисунок 4 – Кластер с наибольшим весом для статьи

Далее обрезаем кластер и формируем аннотацию (рисунок 5):



Рисунок 5 – Конечный результат работы алгоритма

Как видим кластер во многом отражает основные моменты нашего заголовка, и мысль текста передает достаточно верно, конечный результат тоже неплохо отражает содержание, но второе предложение не несет существенного смысла, и лучше бы было выбрать предложение с формальным определением автоматического реферирования.

Возможные модификации алгоритма:

* изменить алгоритм обрезки аннотации, не выбирать предложения с похожим смыслом, для этого возможно, стоит ввести новые параметры ранжирования;
* увечить корпус для обучения нашей модели word2vec, а еще лучше – обучить его на текстах какой-то определенной тематики, например, на статьях и диссертациях технических специальностей;
* заменить алгоритм стемминга на поиск в тексте однокоренных слов, это поможет улучшить эффективность нашей программы;
* перейти от рассмотрения пассажей к расчету веса отдельных предложений.

**ЗАКЛЮЧЕНИЕ**

 Цель курсовой работы – изучение методов лексического и семантического анализа текста, и на основе этих инструментов разработка прототипа программы реферирования текстов – была достигнута.

 Для написания программы реферирования изучены возможности таких библиотек языка Python 3, как nltk (работа с токенами) и gensim (построение модели word2vec), освоены современные инструменты синтаксического и семантического анализа (алгоритмы стемминга и лемматизации). С использованием этих навыков был реализован прототип системы аннотирования документов и приведены некоторые результаты написанного алгоритма.

Результаты работы программы показали, что сгенерированные аннотации отражают ключевые моменты исходного текста и соответствует тематике заголовка документа, однако выбор некоторых предложений мог быть произведен и точнее. Несмотря на некоторые недостатки, при дальнейшей модификации описанный алгоритм автоматического реферирования может применяться на практике.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

 1 Автоматическая обработка текстов на естественном языке и анализ данных/ Большакова Е.И., Воронцов К.В., Ефремова Н.Э., Клышинский Э.С., Лукашевич Н.В., Сапин А.С. – НИУ ВШЭ, 2017. – 269 c.

2 Автоматическая обработка текстов на естественном языке и компьютерная лингвистика : учеб. пособие / Большакова Е.И., Клышинский Э.С., Ландэ Д.В., Носков А.А., Пескова О.В., Ягунова Е.В. — М.: МИЭМ, 2011. — 272 с.

3 Word 2vec // (Engl). – URL:

https://en.wikipedia.org/wiki/Word2vec [16 декабря 2018]

4 Алгоритм контекстно-зависимого аннотирования KGCDA – URL: https://istina.msu.ru/media/publications/article/0fb/ffe/6077093/Statya\_isprav.docx [4 декабря 2018]

5 Браславский П., Колычев И. Автоматическое реферирование веб-документов с учетом запроса – URL: http://elar.urfu.ru/bitstream/10995/1424/1/IMAT\_2005\_25.pdf [28 ноября 2018]

6 Современные методы автоматического реферирования – URL: https://cyberleninka.ru/article/v/sovremennye-metody-avtomaticheskogo-referirovaniya [5 декабря 2018]

7 Салтыков А., Куротченко С., Дорохин Р. Алгоритм контекстно-зависимого аннотирования – URL:

http://romip.ru/romip2010/08\_rookie\_annotation.pdf [15 ноября 2018]

8 Russian stemming algorithm // (Engl). – URL: http://snowball.tartarus.org/algorithms/russian/stemmer.html [26 Ноября 2018]

9 Daniel C. Marcu , Graeme H. The rhetorical parsing, summarization, and generation of natural language texts// (Engl). – 1998. – 396 c.