МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Кафедра дискретной математики и алгоритмики

КУЛИК Сергей Дмитриевич

ЭВРИСТИЧЕСКИЕ АЛГОРИТМЫ ДЛЯ ВЫДЕЛЕНИЯ КЛЮЧЕВЫХ СЛОВ ТЕКСТОВ ВЕБ-СТРАНИЦ НА СТОРОНЕ КЛИЕНТА

Магистерская диссертация

специальность 1-31 81 09 «Алгоритмы и системы обработки больших объемов информации»

Научный руководитель:
Котов Владимир Михайлович
доктор физико-математических наук, профессор

Допущена к защите:
«___» _____ 2019 г.
Зав. кафедрой дискретной математики
и алгоритмики
доктор физико-математических наук,
профессор В. М. Котов

Минск, 2019

ОГЛАВЛЕНИЕ

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Магистерская диссертация, 59 с., 20 источников, 2 приложения.

Ключевые слова: АНАЛИЗ ТЕКСТОВ; ВЫДЕЛЕНИЕ КЛЮЧЕВЫХ СЛОВ; МАШИННОЕ ОБУЧЕНИЕ; ВЫЧИСЛЕНИЯ НА СТОРОНЕ КЛИЕНТА.

Объект исследования – тексты на естественных языках и ключевые слова в них; записи и заметки в блогах, статьи.

Цель работы – исследование методов для выделения ключевых слов текстов на естественных языках на стороне клиента. Цель включает в себя как исследование известных результатов в рассматриваемой задаче и смежных областях, так и поиск новых.

В ходе работы были исследованы имеющиеся подходы для выделения ключевых слов как на стороне клиента, так и на стороне сервера. Были рассмотрены проблемные вопросы построения тренировочных выборок для рассматриваемой задачи, также предложена методология для сравнения получаемых результатов с известными аналогами. Также рассмотрены проблемные вопросы выделения стоп-слов в текстах.

Результатом является метод для выделения ключевых слов на основе алгоритма логистической регрессии. Для данного метода были также проведены вычислительные эксперименты и сравнение результатов с известными аналогами. Показана переносимость данного метода: будучи обучен на текстах на английском языке, метод также показывает высокие результаты при тестировании на текстах на испанском языке.

Областью применения является разработка на стороне клиента; приложения, работающие с конфиденциальными данными; алгоритмы категоризации текстов на естественном языке.

АГУЛЬНАЯ ХАРАКТАРЫСТЫКА РАБОТЫ

Магістарская дысертацыя, 59 с., 20 крыніц, 2 дадатка.

Ключавыя словы: АНАЛІЗ ТЭКСТАЎ; ВЫЛУЧЭННЕ КЛЮЧАВЫХ СЛОЎ; МАШЫННАЕ НАВУЧАННЕ; ВЫЛІЧЭННІ НА БОКУ КЛІЕНТА.

Аб'ект даследавання – тэксты на натуральных мовах і ключавыя словы ў іх; запісы і нататкі ў блогах, артыкулы.

Мэта работы – даследаванне метадаў для вылучэння ключавых слоў тэкстаў на натуральных мовах на баку кліента. Мэта ўключае ў сябе як даследаванне вядомых вынікаў у разгляданай задачы і сумежных абласцях, так і пабудову новых.

У ходзе работы былі даследаваны існуючыя падыходы для вылучэння ключавых слоў як на баку кліента, так і на баку сервера. Былі разгледжаны праблемныя пытанні пабудовы трэніровачных выбарак для разгляданай задачы, таксама прапанаваная метадалогія для параўнання атрымоўваных вынікаў з вядомымі аналагамі. Таксама разгледжаны праблемныя пытанні выдзялення стоп-слоў у тэкстах.

Вынікам з'яўляецца метад для вылучэння ключавых слоў на аснове алгарытму лагістычнай рэгрэсіі. Для гэтага метаду былі таксама праведзены вылічальныя эксперыменты і параўнанне вынікаў з вядомымі аналагамі. Паказаная пераноснасць дадзенага метаду: быўшы навучаны на тэкстах на англійскай мове, метад таксама паказвае высокія вынікі пры тэставанні на тэкстах на іспанскай мове.

Вобласцю ўжывання з'яўляецца распрацоўка на баку кліента; прыкладанні, якія працуюць з канфідэнцыйнымі дадзенымі; алгарытмы катэгарызацыі тэкстаў на натуральнай мове.

ABSTRACT

Master thesis, 59 p., 20 sources, 2 appendices.

Keywords: TEXT MINING; KEYWORD DETECTION; MACHINE LEARNING; CLIENT-SIDE CALCULATIONS.

The object of the study are texts in natural languages and keywords in them; records and notes on blogs and articles.

The objective of the thesis is the study of methods for keywords detection in texts in natural languages on the client side. The goal includes both: the study of known results in the problem and related fields, and the research of new ones.

During the course of the research, the available approaches for detecting keywords on the client side and on the server side were researched. We've also addressed the problematic issues of building training set for the problem, also proposed a methodology for comparing the obtained results with known analogues. The problematic issues of stop words detection in the texts were also addressed.

The result is a proposed method for extracting keywords based on a logistic regression algorithm. For this method, computational experiments were also conducted and the results were compared with known analogues. The portability of this method was shown: being trained on texts in English, the method also shows good results when tested on texts in Spanish.

The field of application is client-side development; applications that work with confidential data; natural language categorization algorithms.

ВВЕДЕНИЕ

Выделение ключевых слов в текстах является актуальной и исследуемой в данный момент задачей. Оно позволяет грамотно организовать категоризацию документов, а также улучшить качество поиска информации.

В данный момент большое внимание уделяется алгоритмам на основе нейронных сетей и большой уклон сделан на обработку на стороне сервера. Но вместе с этим существуют и некоторые трудности: обработка на стороне третьих сервисов может быть дорогой, а в случае работы с конфиденциальными данными, может быть и вовсе недопустимой. Более того, на данный момент многие компании предоставляют интерфейс программирования приложений для выделения ключевых слов документов только на нескольких языках.

Выделение ключевых слов на стороне клиента призвано решить эти проблемы и найти разумный компромисс между точностью, простотой использования и поддержкой множества языков.

В рамках данной работы будут рассмотрены имеющиеся подходы к решению задачи и проведено исследование, в результате которого будет построен альтернативный имеющимся метод выделения ключевых слов из текстов, применимый на клиентской стороне и показывающий более высокие результаты в задаче предсказания тегов на данных блоговых ресурсов. Помимо этого будут рассмотрены вопросы построения выборок для тестирования, непредвзятого сравнения алгоритмов, выделения стоп-слов в текстах на различных языках, а также расширяемости предложенного алгоритма на случай нескольких языков.

ГЛАВА 1. ОБЩИЕ ПОЛОЖЕНИЯ

1.1 Постановка задачи

Ключевым объектом исследования в диссертации являются ключевые слова. Формализуем и дадим определение этому понятию.

Пусть рассматривается некоторая коллекция документов, где текст каждого документа написан на каком-то из существующих естественных языков. Выберем некоторый произвольный документ из этой коллекции. Ключевыми словами данного документа будем считать одно или несколько слов или словосочетаний, характеризующих данный текст в выбранной коллекции. В широком смысле, ключевые слова документа можно рассматривать как его категории в данной коллекции.

Так например, в коллекции веб-страниц, страница содержащая новость про успешное выступление футбольной команды может иметь ключевые слова “новости” и “спорт”, как и некоторые другие, например, название этой команды. Однако, в коллекции новостных веб-страниц она уже не будет иметь ключевого слова “новости”, так как тогда данное слово не дает нам никаких новых знаний о данной статье в пределах рассматриваемой коллекции.

Задача диссертации – исследование и разработка эвристических алгоритмов для выделения ключевых слов документов веб-страниц на стороне клиента.

Ограничение о выполнении кода на клиентской стороне накладывает некоторые ограничения на класс применяемых алгоритмов, так как это означает, что полученные алгоритмы должны быстро и эффективно работать на обычных персональных ЭВМ, в распоряжении пользователей которых нет кластеров с сотнями или тысячами процессоров и при этом, полученные алгоритмы не должны использовать дополнительных вычислительных мощностей в облачных и подобных сервисах.

1.2 Результаты в смежных областях

Самой близкой по тематике к рассматриваемой работой является работа американских исследователей Стьюарта Роуза (Stuart Rose), Дейва Энгела (Dave

Engel), Ника Крамера (Nick Cramer) и Венди Каули (Wendy Cowley) под названием “Automatic keyword extraction from individual documents” [1], датированная мартом 2010 года.

Работа довольно широко известна как простой эвристический алгоритм для получения ключевых фраз документа и существует несколько реализаций приведенного в ней алгоритма на различных языках программирования, в том числе таких как Java Script и Type Script, использующимися в разработке веб-страниц на стороне клиента. Существует также библиотека, основанная на данном алгоритме на языке программирования Python, которая даже доступна в официальном репозитории пакетов языка PyPI.

Примечательна статья также тем, что там приводится эффективный, единый для всех языков метод выделения стоп-слов в рассматриваемом документе. Отметим однако, что приведенный метод может давать и ложноположительные срабатывания.

Преимуществами работы американских исследователей являются простота алгоритма и асимптотика времени его работы – $O(NlogN)$ для текстов длины N символов, притом асимптотика $O(NlogN)$ получается только за счет использования алгоритмов сортировки, константа которых мала. Минусами для нас являются не всегда высокое качество определения ключевых слов и фраз – будет подробно оценено в Главе 4, а также выделение длинных ключевых фраз, что делает данный алгоритм неприменимым к задачам категоризации документов, а также его неприспособленность к текстам веб-страниц.

Немного другой подход, применимый к анализу данных на клиентской стороне предлагается в работе “TextRank: Bringing Order into Texts” [2] исследователей Рады Михалцеа (Rada Mihalcea) и Пола Тарау (Paul Tarau). В статье предлагается рассмотреть текст как граф, в котором вершины являются словами, а ребра – связями между этими словами. Далее предлагается модифицированная версия алгоритма вычисления ранга Пейджа (алгоритм предложен в 1998 году Сергеем Брином и Ларри Пейджем в работе “The PageRank Citation Ranking: Bringing Order to the Web” [3]). Ранг Пейджа был изначально предназначен для решения задач из области информационного поиска, ключевым же моментом работы Михалцеа и Тарау является адаптация предложенного Пейджем и Брином инструментария для задачи анализа текстов. Предлагается подход с построением

графа по заданному тексту, затем делается предположение о том, что вершины, соответствующие ключевым словам будут связаны с вершинами, имеющими большие значения ранга Пейджа в построенном графе.

Также достойна упоминания статья “Topic Detection by Clustering Keywords” [4] исследователей нидерландского института Telematica Кристиана Вартена (Christian Wartena) и Роджера Брюссее (Rogier Brussee) датируемая сентябрем 2008 года, где рассматривается подход уже несколько более далекий от выполнения всех вычислений на клиенте: вводятся понятия расстояния между термами текста, после чего рассматриваются варианты применения различных алгоритмов кластеризации точек в многомерном пространстве. Этот подход уже более сложен, чем предыдущий, и не может быть так просто реализован на клиенте: требуется предобработка, обучение на некотором корпусе из документов. Еще одним недостатком для нас является то, что множество возможных ключевых слов фиксировано заранее. Большим же преимуществом по сравнению с предыдущей рассматриваемой статьей является высокая точность классификации.

Идеи же представления слов в тексте как точек в многомерном пространстве, между которыми можно вычислить некую меру расстояния были позже рассмотрены и в более общем виде. В качестве наиболее яркого представителя этой идеи хочется выделить работу “Distributed Representations of Words and Phrases and their Compositionality” [5] авторов Томаша Миколова (Tomas Mikolov), Ильи Суцкевера (Ilya Sutskever), Кай Чена (Kai Chen), Грега Коррадо (Greg Corrado) и Джеффри Дина (Jeffrey Dean). Работа была впервые представлена на конференции NIPS в 2013 году и дала начало методу word2vec, который позволяет отобразить слова из естественного языка в точки некоторого многомерного пространства. Также работа нашла отражение и в нейронных сетях, применяющихся для решения задач обработки текстов на естественных языках (Natural Language Processing) в виде специального слоя нейронной сети, имеющего название Embedding-слой.

Если рассматривать более поздние работы по выделению ключевых слов путем ранжирования множества кандидатов с использованием некоторого ограниченного множества тем, стоит отметить, что использование множества “кандидатов” в ключевые слова на основании заданного заранее известной предметной области также рассматривается в работе “Human-competitive automatic topic indexing” [6] исследователя Олены Меделян (Olena Medelyan) из университета

Вайкато, Новая Зеландия. В работе проводится исследование решений в данной области и построение алгоритма MAUI, который, на момент ее написания, обогнал имеющиеся аналоги, решающие задачу в такой же постановке. Данный алгоритм использует наивный байесовский классификатор и случайный лес для выявления ключевых слов.

Если же рассматривать более современные и более вычислительно тяжелые методы на основе нейронных сетей, стоит отметить работу “Keyphrase Extraction Using Deep Recurrent Neural Networks on Twitter” [7] исследователей Ки Жанга (Qi Zhang), Янга Вонга (Yang Wang), Еюна Гонга (Yeyun Gong) и Суанджина Хуанга (Xuanjing Huang), представленную на конференции EMNLP 2016. В работе рассматривается архитектура для глубокой рекуррентной нейронной сети, которая показывает сильные результаты в выделении ключевой темы записи в социальной сети Twitter. Интересен также и подход к построению набора исходных данных, предлагаемый в статье: авторы рассматривают поток сообщений в социальной сети, которые имеют единственный хэштег, таким образом, делая своего рода допущение в исследуемой ими задаче, позволяющее однако показать более высокий результат.

Отметим наконец, что к задаче выделения слов, характерных документу в рассматриваемой коллекции также существуют популярные классические методы, как например, метод TF-IDF, описанный Хуаном Рамосом (Juan Ramos) в 2003 году в работе “Using TF-IDF to Determine Word Relevance in Document Queries” [8]. Вес, который имеет слово как ключевое в этом методе, вычисляется на основании двух величин: TF, или Term Frequency – частота терма, в нашем случае – количество вхождений рассматриваемого слова или фразы в документ и IDF, или Inverse Document Frequency – обратная документная частота, которая характеризует, насколько редок рассматриваемый терм в коллекции. Данная работа будет интересна для нас в большей мере в контексте выделения стоп-слов, так как ограничения, накладываемые решаемой нами задачей не позволяют использовать меру IDF, рассчитанную на большом корпусе документов. Применение данного подхода для выделения стоп-слов будет рассмотрено в Главе 8, подразделе 8.2.

Таким образом, упомянутые существующие работы закладывают фундамент и задают отправную точку для дальнейшего исследования по теме магистерской диссертации. Несмотря на близость тем, описанные в них методы не могут быть

применены напрямую для решения поставленной задачи, однако содержат в себе чрезвычайно полезные идеи и результаты.

1.3 Существующие онлайн-сервисы по выделению ключевых слов

Для выделения ключевых слов на стороне сервера существуют несколько онлайн-ресурсов, предоставляемых в рамках концепции aaS (as a Service, как сервис). Отметим некоторые из них:

- Сервис Azure Cognitive Services, предоставляемый компанией Microsoft для выделения ключевых слов из текстов на английском языке.
- Сервис Cloud Natural Language API, предоставляемый компанией Google для выделения ключевых слов из текстов на английском, китайском и испанском языках.
- Несколько сервисов от не столь крупных компаний или индивидуальных разработчиков, которые в частности реализуют идеи из статей, рассмотренных нами ранее.

Отметим, что данные сервисы, хоть и не реализуют методы выделения ключевых слов на клиенте, но могут быть использованы нами, в частности, в задаче разметки полученной выборки.

1.4 План магистерской диссертации

В данной главе нами была поставлена задача выделения ключевых слов на стороне клиента. В подразделе 1.2 был проведен краткий обзор нескольких работ и публикаций по смежным темам: выделение ключевых слов, задачи анализа текста на естественном языке.

Для построения качественного алгоритма необходимо, в первую очередь, провести более тщательное исследование решаемой задачи. Существующие наборы исходных данных ограничены и не позволяют провести подробное исследование. Поэтому, в главе 2 будут рассмотрены способы построения выборок и наборов исходных данных на основании имеющихся решений, а также блоговых ресурсов.

Продолжая начатое в главе 2, в главе 3 задача построения выборки будет доведена до логического завершения: будет предложен обобщенный метод построения набора исходных данных, рассматривающий произвольный веб-ресурс в качестве источника текстов для обучающей выборки. Отметим также, что

использование сырого тела веб-страницы также невозможно в силу того, что ее навигационные и прочие вспомогательные элементы могут появляться в произвольных местах полученного текста. С целью их очистки и приведения веб-страницы в более подходящий для выделению ключевых слов вид, будет предложен простой эвристический алгоритм.

Глава 4 будет посвящена исследованию и анализу уже полученных в главах 2 и 3 выборок, выявлению ключевых факторов, свидетельствующих о том, что слово является ключевым. Также, в этой главе будет разработан и приведен метод сравнения получаемых в диссертации результатов с известными аналогами – алгоритмами RAKE и TextRank. Будет уделено особое внимание таким моментам, как обеспечение равных условий для алгоритмов сравнения и исключение человеческого фактора в процессе вычислительного эксперимента.

В главе 5 нами будет построен эвристический алгоритм для выявления ключевых слов текстов блоговых ресурсов. Плюсом рассмотрения блоговых ресурсов является тот факт, что они являются идеально размеченными объектами: теги к записям были проставлены живыми людьми – авторами записей, без сведения о том, что записи будут применяться в выборке для обучения алгоритмов и оценки качества. Таким образом, возможно максимально точно сымитировать случай реального мира – в разметке текстов представляется возможность сравнить свой результат с тем, который должен быть получен по мнению автора текста. Алгоритм будет основан на построении множества кандидатов и его ранжировании согласно выбранным факторам. Для поиска весов факторов будет применен алгоритм GridSearch, также известный как поиск по решетке параметров. Вычислительным экспериментом будет показано, что найденное решение превосходит известные аналоги – алгоритмы RAKE и TextRank.

В главе 6 будет рассмотрена возможность применения методов машинного обучения к решению данной задачи. Вначале будет проведено рассмотрение современных state-of-the-art решений для анализа текстов. После этого, будет проведено сведение задачи ранжирования, полученной в главе 5 к задаче машинного обучения, в частности – задаче бинарной классификации. Будет рассмотрена применимость классических алгоритмов машинного обучения к решению задачи: алгоритмов случайного леса, метода опорных векторов, логистической регрессии,

метода ближайших соседей. На основании рассмотренных методов удастся улучшить качество полученного в главе 5 алгоритма.

Глава 7 будет посвящена различным подходам для выделения ключевых слов в текстах на различных языках. Будут рассмотрены, начиная от простейших, методы использования списка стоп-слов, метод построения списка стоп-слов, основанный на использовании метрики TF-IDF, а также эвристический алгоритм, предложенный авторами работы [1]. Будут сделаны выводы о выборе подхода в зависимости от задачи.

В главе 8 будут рассмотрены проблемные вопросы портирования построенного ранее алгоритма на случай нескольких языков. Будет проведен данный перенос для случая испанского языка, рассмотрены варианты обучения заново и использования данных, полученных в результате обучения на текстах английского языка.

Глава 9 будет заключительной. В данной главе будут подведены итоги исследования и кратко описан проделанный объем практической работы.

ГЛАВА 2. ПОЛУЧЕНИЕ НАБОРОВ ИСХОДНЫХ ДАННЫХ ДЛЯ ЗАДАЧИ ВЫДЕЛЕНИЯ КЛЮЧЕВЫХ СЛОВ

2.1 Обзор готовых решений. Проблемные вопросы

Часто для задач классификации и машинного обучения существуют готовые наборы исходных данных, составленные исследователями, решающими смежные задачи в рассматриваемой области, или же программистами-одиночками. Наборы хранятся в соответствующих репозиториях. Примером такого репозитория может служить хранилище датасетов Калифорнийского университета в Ирвайне, а также Интернет-ресурс Kaggle, проводящий соревнования по машинному обучению в различных областях.

Во время исследования вопросов построения обучающей и контрольной выборок возникла проблема, связанная с отсутствием в свободном доступе наборов исходных данных, подходящих для рассматриваемой задачи. Так, ресурсы SNAP [9] (университет Стенфорда) и DMS (Data Mining and Search Group, академия наук Венгрии) [10] предлагают в качестве датасетов по блогам только графы социальных связей пользователей ресурса LiveJournal. Это ограничение связано с мерами по избежанию возможного нарушения авторских прав, так как записи пользователей могут подпадать под защиту. Таким образом, найти готовый набор исходных данных такого рода является практически невыполнимой задачей.

В качестве альтернативы можно обратиться к научным исследованиям в смежной области. Так, в работах [1, 2, 6] часто приводятся методы, использующие научные статьи с аннотациями для построения набора исходных данных для задачи теггирования текстов. Ввиду более свободных условий лицензирования, а также нахождения в открытом доступе они могут быть использованы для подобных целей. Преимуществом подобных наборов исходных данных является точность разметки, однако есть и несколько недостатков: узкая предметная область текстов – они относятся только к научным исследованиям, а также малый размер получаемого набора исходных данных.

Таким образом, некоторые небольшие выборки, покрывающие частные случаи могут быть найдены в открытых источниках, однако для полноты картины необходимо рассмотреть варианты для построения собственного набора исходных данных, который дополнит имеющиеся данные и покроет все нужды проводимого исследования.

Далее будут рассмотрены несколько способов сделать это – построение выборки по данным аннотированных тегами ресурсов – на примере блогового ресурса WordPress, а также построение выборки по данным любого рода с автоматическим теггированием с помощью интерфейсов программирования приложений от сторонних сервисов, в том числе, рассмотренных в разделе 1.3.

2.2 Разработка формата представления собранных выборок

В силу того, что данные собираются из разных источников и различными методами, необходимо формализовать формат, в котором эти данные будут храниться. Формат должен быть достаточно гибок и в то же время, быть удобным для проведения вычислительных экспериментов и проверки гипотез.

2.2.1 Нормализация текста

Нормализованным видом текста веб-страницы будем называть следующий его вид:

- Отсутствуют теги HTML-разметки;
- Отсутствует любое выделение текста;
- Все переводы строк заменены на пробелы;
- Отсутствуют символы не из ASCII-таблицы. Для символов Юникода допускается перекодирование ASCII-символами.

Мы будем рассматривать нормализованный текст страницы, а также нормализованный текст ее заголовка.

2.2.2 Нормализация тегов

Нормализованным тегом или нормализованным ключевым словом назовем последовательность символов, которая удовлетворяет следующим условиям:

- Является нормализованным текстом;
- Не содержит прописных символов: в процессе нормализации они должны быть заменены на строчные;
- Не содержит символа решетки (начала хэштега) в начале.

Третье требование необходимо для того, чтобы привести два популярных представления тега к одному: в Интернете встречаются примерно в равной пропорции как тег “#коты”, так и просто тег “коты”.

2.2.3 Формат представления выборки

Таким образом, теперь мы можем ввести формат представления текста в выборке. Один текст будет представлен словарем в формате JSON со следующими полями:

- **content_raw** – содержимое рассматриваемой веб-страницы или текста “как есть”, со всеми тегами и верстской без изменений в формате HTML;
- **content_clean** – нормализованный текст рассматриваемой веб-страницы;
- **title_raw** – содержимое заголовка веб-страницы со всеми тегами и верстской без изменения, в формате HTML;
- **title_clean** – нормализованный заголовок рассматриваемой веб-страницы;
- **tags** – список нормализованных тегов рассматриваемой веб-страницы.

2.3 Получение выборки посредством анализа блоговых ресурсов

Рассмотрим первый из методов самостоятельного получения выборки, описанных в конце подраздела 2.1: анализ ресурса, содержащего записи блогов.

Блогом называется сайт, на котором есть лента новостей одного человека; личная страница в Интернете. Такие ресурсы, как правило, содержат записи, сделанные человеком на различные темы. Характерной особенностью блогов являются теги, которыми помечаются многие записи. Теги служат ключевыми словами записей, которые используются для их категоризации и упорядочивания.
Таким образом, практически любая заметка в блоге может в рамках проводимого исследования служить элементом обучающей выборки.

После проведенного исследования блоговых ресурсов был найден ресурс WordPress, который предоставляет REST API для доступа к заметкам по заданному тегу.

Сбор выборки был организован следующим образом. В качестве тегов, которые будут задаваться ресурсу, были выбраны 3000 самых популярных слов английского языка. Далее, по каждому из них был сделан запрос в REST API ресурса на получение не более 10 самых последних статей. Полученные данные были приведены к виду, описанному в подразделе 2.2: тексты и теги были нормализованы.

В результате было получено 28722 статей, суммарным объемом 227 мегабайт. Характерной особенностью используемого API оказалось медленное время ответа и слабая пропускная способность. В связи с этим, запросы делались с таймаутом в 10 секунд, в 5 потоков. В случае неуспеха, делалась задержка, длительность которой возрастала экспоненциально: гарантированно домножалось на 1.2, после чего добавлялось случайное число между 0 и 0.15. Это позволило собрать выборку и соблюсти баланс между скоростью работы и облегчением нагрузки на сервер.

В качестве плюсов данного метода стоит отметить высокое качество выборки: тексты были размечены людьми, которые их писали. Минусами же является сравнительно маленький размер текстовой базы, неадаптированность для получения произвольного вида выборок, наличие преимущественно только текстов на английском языке.

2.4 Получение выборки посредством анализа структуры произвольного веб-ресурса. Частный случай

Был рассмотрен и второй из предложенных в подразделе 2.1 методов самостоятельного сбора выборки. Метод заключается в выборе некоторого множества веб-страниц, составляющих выборку, выделении в них главного тела и выделении в главном теле ключевых слов посредством API, предоставляемых третьими сервисами.

Был рассмотрен частный случай сбора такого корпуса статей на примере свободной энциклопедии “Википедия”.

Отметим некоторые причины, по которым использование Википедии облегчает задачу. Во-первых, это единое форматирование, которое используется в статьях и позволяющих более просто выделить главное тело статьи. Во-вторых, это

возможность перехода на случайную статью на выбранном языке. В-третьих, это стиль самого ресурса, гарантирующий, что случайная статья будет содержать грамотный и осмысленный текст на английском языке.

Посредством рассмотрения этих особенностей, были выявлены следующие особенности, использованные в составлении датасета:

- Главное тело статьи в документно-объектной модели ее HTML-документа задается тегом div с параметром **class** равным **mw-parser-output**.
- Случайная статья на заданном языке может быть получена путем перехода по адресу вида **https://<код_языка>.wikipedia.en/wiki/Special:Random.**

Таким образом, становится возможным получить качественную выборку текстов на практически любом известном языке, однако, не аннотированную тегами. Аннотировать ее тегами можно с помощью API сервисов по выделению ключевых слов, рассмотренных в подразделе 1.3.

Таким образом, плюсом данного метода является возможность получения выборки на произвольном языке, однако минусом является тот факт, что статьи тегами не аннотированы, и использование сторонних API может лишь частично решить эту проблему. В результате, возможности для построения выборки становятся шире, однако качество ухудшается из-за автоматической разметки.

ГЛАВА 3. ОБОБЩЕННЫЙ МЕТОД ПОЛУЧЕНИЯ ВЫБОРКИ

3.1 Высокоуровневое описание

В случае с произвольным ресурсом как источником текстов для выборки, необходимо построение более сложных алгоритмов. Пример такого алгоритма был разработан и реализован также в рамках производственной практики.

Основой метода получения выборки по произвольному множеству страниц является поисковый робот, функцией которого является обнаружение и передача обработки найденных веб-страниц. Важной деталью в реализации робота являются так называемые рабочие (в английской литературе – workers), функцией которых является загрузка данных из найденных роботом страниц до тех пор, пока количество обработанных страниц не станет удовлетворять условиям выборки.

Таким образом, робот сможет получить некоторое множество страниц в “сыром” виде. Этот вид, однако непригоден для использования непосредственно в выборке. Причиной является то, что полученные страницы содержат лишние, или “шумовые” элементы. Примерами таких элементов могут являться панель навигации, фотографии, ссылки на другие статьи сайта и прочее. Необходимо провести детекцию главного тела веб-страницы – содержательной части, очищенной от лишних элементов. Для этого был разработан и реализован эвристический алгоритм, рассматриваемый в разделе 3.3.

После того, как главное тело веб-страницы выделено, может быть проведено выделение ключевых слов с помощью API третьих сервисов: многие из них выделяют квоту бесплатных запросов.

3.2 Построение робота

Архитектура робота, ищущего веб-страницы страницы, главное тело которых подходит для включения в набор исходных данных выбрана в соответствии с шаблоном проектирования master-slave (хозяин-подчиненный).

Хозяин в данном случае реализует следующий функционал:

- Поддержка очереди адресов веб-страниц, которые должны быть обработаны, и возможно включены в множество веб-страниц, образующих датасет. В некоторых реализациях очередь может являться отдельным элементом в архитектуре;
- Получение и назначение подчиненным URL веб-страниц, которые должны быть загружены и обработаны;
- Отправка подчиненным сигналов о завершении работы;
- Определение принадлежности веб-страницы конструируемому набору исходных данных;
- Определение необходимости включения адреса веб-страницы в очередь адресов, которые необходимо обработать.

  Подчиненные же призваны решать следующие задачи:
- Скачивание и обработка назначенной веб-страницы;
- В случае необходимости включения текста веб-страницы в набор исходных данных, выделение главного тела веб-страницы и включение ее в набор исходных данных в принятом формате.

Как принято, в данном шаблоне проектирования, хозяин может быть только один, так как им решается задача координации всего процесса построения набора исходных данных. Подчиненных может быть несколько. Данная схема была протестирована на одном ЭВМ, с использованием нескольких потоков для обеспечения параллелизма. Она также может быть расширена на несколько ЭВМ, путем назначения одной из ЭВМ роли “хозяина”, а остальным – роли “подчиненного”. Стоит отметить, что в этом случае хозяин является единственной точкой отказа, поэтому в случае с распределенной схемой может иметь смысл вынести очередь из хозяина, а также реализовать механизм active-passive failover.

3.3 Эвристическое выделение главного тела веб-страницы

Был построен эвристический алгоритм для выделения главного тела веб-страницы на основании анализа ее DOM – документно-объектной модели. DOM-модель является представлением содержимого страницы в виде корневого дерева, где корневой является вершина, соответствующая корневому тегу <html>, содержащего внутри всю остальную верстку. Каждая вершина в данной модели

может иметь вершин-непосредственных потомков, задаваемых непосредственно вложенными в тег вершины тегами и определяющие элементы верстки более низкого уровня.

Алгоритм основывается на эвристическом утверждении, что возможно выделить одну вершину документно-объектной модели, которая будет содержать в себе только информацию, содержащую главное тело документа. На практике, такая вершина либо существует, либо же есть вершина со схожими свойствами, которая содержит в себе всю существенную информацию веб-страницы и не может быть расщеплена в виду того, что никакой из ее непосредственных детей не содержит всей существенной информации.

Таким образом, по вершинам документно-объектной модели строится *множество кандидатов* – тех вершин, которые предположительно содержат в себе всю существенную информацию. Далее, из множества кандидатов выбирается та вершина, которая содержит наименьшее количество видимых символов.

Определим некоторые величины, использующиеся далее. Обозначим через *textChars* общее количество видимых символов на странице, а через *nodeChars* – общее количество видимых символов в фрагменте текста, содержащемся в рассматриваемой в данной момент вершине. Также зададим два параметра:

- Параметр *minSplitRate*, задающий минимальную долю текстовой информации вершины из множества кандидатов, которую необходимо содержать потомку вершины из множества кандидатов для того, чтобы иметь возможность быть включенным в это множество. В силу эвристического утверждения о том, что количество текста в “шумовых” элементах верстки не должно быть значительным, имеет смысл выбирать $minSplitRate > 0.5$.
- Параметр *minMainBodyRate*, задающий минимальную долю всего текста страницы, который необходимо содержать вершине из множества кандидатов.

Данные параметры должны выбираться в зависимости от специфики рассматриваемого класса веб-страниц. Хорошо себя показали следующие значения: $minSplitRate = 0.7$, $minMainBodyRate = 0.3$.

Определим правило, по которому строится множество кандидатов. Первой вершиной, включенной в данное множество является корень дерева. Далее, итеративно происходит следующий процесс: среди непосредственных потомков последней из выбранных вершин выбирается тот, который содержит не менее, чем

$$\min\{ textChars \cdot minMainBodyRate,\ nodeChars \cdot minSplitRate\}$$

В случае, если такой вершины нет, процесс построения множества кандидатов является оконченным.

В силу того, что *minSplitRate > 0.5*, каждый элемент множества кандидатов породит не более чем одного элемента этого множества. Таким образом становится возможным и итеративное вычисление нового кандидата. Также, при minSplitRate > 0.5 верно то, что последний найденный кандидат и будет являться вершиной, содержащей главное тело документа.

Отметим, что данный алгоритм был реализован в рамках работы и его код на языке Python, а также примеры работы доступны на ресурсе GitHub (см. Приложение 1).

3.4 Выводы

Рассмотренный алгоритм был также реализован в общем виде в рамках магистерской диссертации. Это позволило создавать наборы исходных данных по произвольным веб-ресурсам путем определения всего двух методов: проверки URL веб-страницы на вхождение в набор исходных данных и проверки URL веб-страницы на необходимость добавления этого URL в очередь веб-страниц на обработку.

Для данной реализации также был реализован пример – сбор набора исходных данных текстов по веб-ресурсу Quora, ориентирующийся на сбор текстов ответов на заданные на веб-ресурсе вопросы. В данном примере метод, проверяющий URL на вхождение в датасет выполнял проверку на его соответствие ответу, а метод, проверяющий URL на необходимость добавления в очередь проверял его на принадлежность следующим трем классам интернет-адресов:

- Ответ на заданный вопрос;
- Список ответов на заданный вопрос;
- Профиль пользователя

Отметим, что ресурс Quora выбран лишь для демонстрации возможностей. Любой ресурс, адреса страниц которого можно разделить на принадлежащие набору исходных данных и не принадлежащие может быть также обобщен для задачи сбора набора исходных данных.

Дальнейшими направлениями работы по совершенствованию полученного алгоритма могут быть улучшения алгоритма выделения главного тела страницы и использование метода Simhash для более качественной дедупликации результатов.

ГЛАВА 4. ИССЛЕДОВАНИЕ ИСХОДНЫХ ДАННЫХ РАССМАТРИВАЕМОЙ ЗАДАЧИ

4.1 Исследование данных ресурса WordPress. Построение гипотез

В первую очередь интерес представляет исследование данных, которые наиболее близки к рассматриваемой задаче, и в которых делается наименьшее число допущений и упрощений таких, как, например, использование третьих сервисов для выделения ключевых слов, либо единый и энциклопедичный стиль. Такими данными представляется набор исходных данных сайта WordPress.com, полученный в главе 2. Проведем исследование этого набора.

Начнем исследование с визуального обзора случайно выбранных текстов из данной выборки. Попытаемся высокоуровнево выделить особенности из данных текстов. Возьмем для примера не очень большую заметку, являющуюся оценочным суждением автора касательно статью на электронном ресурсе CDC.gov – центр по контролю за болезнями и их распространением в Соединенных Штатах Америки. Для удобства анализа, указанные автором ключевые слова выделены жирным шрифтом.

Massive News: **Chickenpox vaccine** linked to **widespread increase** in **shingles** [CDC coverup]

Most kids today will never know the misery of a bout of **chickenpox** because of the **chickenpox vaccine**, but it looks like all of us could be paying a pretty big price for it in the form of **shingles** When you get **chickenpox**, the virus will remain latent in your body. If it's reactivated later in life, it can reappear as **shingles**. Before the **vaccine** was introduced, most adults were able to avoid getting **shingles** because their exposure in their communities to natural **chickenpox** regularly boosted their cell-mediated immunity to it. In other words, exposure to **chickenpox** in adults helped protect them from getting **shingles** Author Gary S. Goldman is a former research analyst for the Los Angeles Department of **Health**, and he monitored the introduction of the **chickenpox vaccine**. He says that by 2000, he was hearing a lot of anecdotal accounts from school nurses about an inexplicable rise in **shingles** cases among students. He found that the **vaccine** was not only accelerating the recurrence of **shingles** among children who had naturally gotten **chickenpox**, but it was also boosting the chances of adults getting **shingles**. Goldman says the CDC stopped him from making his data public https://www.naturalnews.com/2018-06-28-explosive-research-**chickenpox**-**vaccine**-linked-to

-widespread-**increase**-in-**shingles**.html Don t take vaccines unless you have a death wish. They are created and designed to MURDER you and your family.

Ключевые слова: **chickenpox**, disease, **health**, **increase**, link, **massive**, **news**, **shingles**, sickness, **vaccine**, **widespread**.

Итак, автором было приведено 11 ключевых слов. Каждое из них является именно словом, а не словосочетанием – такой случай также, безусловно, имеет место.

Из одиннадцати ключевых слов автора, в тексте или заголовке встретилось восемь: не встретились слова disease (с англ. “болезнь”, в заметке в целом обсуждается влияние вакцины на заражение болезнями, но в явном виде этого слова в тексте нет), link (с англ. “ссылка”, по всей видимости, автор приводит ссылку на новостной ресурс и отмечает это в тегах) и sickness (с англ. “болезнь”, аналогично первому слову). Из этого можно сделать вывод, который важен для алгоритмов, которые строят ответ на основе ранжирования множества кандидатов в ключевые слова, которые выбираются как подстроки из текста: не все указанные человеком ключевые слова в случае “реального мира” можно получить, ранжируя подстроки и комбинацию слов текста.

Из примера также следует гипотеза о том, что немаловажным для выделения ключевых слов из текста веб-страницы является заголовок. В данном случае, в заголовке встречается семь ключевых слов из одиннадцати при том, что в тексте и заголовке в целом встречается только восемь. В то же время, так как рассматривается случай веб-страницы, заголовок может быть извлечен всегда – чаще всего он находится в теге `title` её HTML-разметки, либо в специальном поле ввода, если речь идет о блогах и подобных редакторах статей. В любом случае, данная гипотеза заслуживает детального исследования и разбора.

Отметим также, что отдельные слова, хорошо характеризующие текст появляются в нем часто. Так, в рассматриваемом примере, слово `chickenpox` встречается в тексте семь раз – первое место по встречаемости, если не учитывать стоп-слова: `of`, `the`, `in`, `to`, `it`, `a`, `was`. Слово `shingles`, отмеченное ключевым, также встречается в тексте семь раз и делит со словом `chickenpox` первое место по встречаемости. Также в тексте относительно часто присутствует слово `vaccine` – четыре вхождения и третье место по количеству вхождений, сразу после `chickenpox` и `shingles`. Таким образом, число вхождений слова в текст также является важным фактором, который необходимо использовать и исследовать.

Отметим, что в алгоритме RAKE [1] число вхождений является основным фактором, используемым при ранжировании кандидатов. Следовательно необходимо изучить, как связаны частое вхождения слова в текст в качестве подстроки и его отношение к ключевым словам. Отметим также сразу, что большая встречаемость слова в тексте еще не свидетельствует о его принадлежности ко множеству ключевых слов: первое место по встречаемости в тексте без удаления любых слов занимает слово “the” (определенный артикль в английском языке), а второе – слово “of” (указанием на принадлежность чего-либо чему-либо в английском языке). Оба этих слова, очевидно, не несут смысловой нагрузки, поэтому не могут быть ключевыми. Такие слова называются стоп-словами и их необходимо определять и исключать из текста. Различные подходы к выделению таких слов будут рассмотрены в одной из глав данной диссертации.

Также можно рассмотреть позиции, на которых полученные ключевые слова встречаются в тексте впервые, сделав эвристическое предположение о том, что ключевое слово или фраза должны впервые встречаться в начале текста. На основании этого наблюдения можно выделить такой фактор, как первая встречаемость слова и, например, штрафовать кандидатов в функции ранжирования пропорционально позиции первого вхождения. Также на основании позиций возможна реализация такого фактора, как *распространенность* слова в тексте, равная разности позиции его последнего вхождения в текст и первого вхождения в текст.

Наконец, необходимо заметить, что все предыдущие факторы дают заметное преимущество тем ключевым словам, которые состоят из одного терма (слова). Действительно, пусть имеется некоторая строка S^+, содержащая в себе более короткую, в терминах количества полных слов – мы рассматриваем *только* строки, состоящие из целого числа слов – строку S, например, в качестве префикса. В таком случае можно нетрудно показать истинность следующих утверждений:

- Строка S^+ встречается в любом тексте не большее число раз, чем строка S, являющаяся ее подстрокой.
- Если строка S^+ встречается в заголовке в качестве подстроки, то в качество подстроки в заголовке встречается и подстрока строки S^+, в том числе и строка S. Следовательно, значение фактора встречаемости строки S^+ в

заголовке также не может превосходить данного значения для более короткой ее подстроки S .

- Если строка S является префиксом строки S^+, то строка S^+ первый раз в тексте встретится не раньше, чем строка S. Следовательно, значение штрафа для строки S^+ не может быть меньше, чем значение штрафа для строки S.
- Резюмируя, значение функции ранжирования по трем построенным в подразделе факторам для более длинной строки S^+ всегда будет не меньше значения для ее более короткого префикса S .

В то же время, такое ранжирование не является до конца верным. Например, при выделении ключевых слов статьи про искусственный интеллект, стоит ожидать кандидатов “искусственный”, “интеллект”, “искусственный интеллект”, из которых последний, более длинный и, очевидно, более релевантный, будет иметь значение функции ранжирования не больше, чем первый – слово “искусственный”, как было показано абзацем выше. На самом деле, достаточно лишь одного вхождения слова “искусственный” в текст без слова “интеллект” рядом для того, чтобы слово “искусственный” имело значение функции ранжирования выше. Также и со словом “интеллект”: число его вхождений в текст наверняка будет строго больше, чем число вхождений словосочетания “искусственный интеллект”, поэтому, проигрывая лишь из-за штрафа за чуть более позднее первое вхождение, кандидат “интеллект” скорее всего также будет ранжирован выше из-за большего числа вхождений. Вывод: необходимо вводить факторы, которые будут уравнивать более длинных и более коротких кандидатов в ключевые слова.

В алгоритме RAKE [1] это сделано посредством добавления в функции ранжирования встречаемостей всех отдельных слов кандидата. Этот подход, однако, имеет довольно весомый недостаток: таким образом алгоритм отдает предпочтение более длинным кандидатам, которые могут быть избыточными. Мы также экспериментально установим в следующем подразделе тот факт, то люди чаще выбирают более короткие ключевые слова.

Поэтому нами в рамках исследования предлагается следующий вариант: сделать множитель перед вхождением отдельных слов в текст гиперпараметром, который можно настроить с помощью алгоритмов комбинаторной оптимизации или машинного обучения. Более того, видится что значение этого гиперпараметра должно варьироваться от длины кандидата в словах, так как при рассмотрении

кандидата длиной в три слова мы имеем суммарно больше вхождений “подслов”, чем у кандидата длиной в два слова. Таким образом, для каждой длины кандидата в словах введем свой гиперпараметр.

Итак, на примере текста с ключевыми словами нами было построено несколько гипотез о том, что может являться факторами при решении данной задачи, или гиперпараметрами, которые можно настроить. Далее будет проведен более детальный их анализ.

4.2 Анализ построенных гипотез

Анализ проводился на основании построенных в главах 2 и 3 наборов исходных данных с использованием скриптового языка Python версии 3.5.2. Скриптовая программа, использованные при анализе текста может быть найдена в репозитории с кодовой базой магистерской диссертации по относительному пути `masters/dataset_analysis`.

Анализ с разных точек зрения будет проведен в различных подразделах данного подраздела.

4.2.1 Основные метрики размеченных людьми текстов

Рассмотрим блоговые данные – заметки сайта WordPress. При подсчете статистики использовалась построенная ранее выборка из 28722 статей на английском языке.

Статистика по встречаемости ключевых слов следующая: в среднем автор добавляет к своей заметке 6.79 ключевых слов, однако из них только 3.07 слова встречаются в том множестве кандидатов, которое строит алгоритм RAKE и прочие алгоритмы для выделения ключевых слов без применения нейронных сетей. Таким образом подтверждается гипотеза о том, что не все слова, которые автор считает ключевым встречаются непосредственно в написанном им тексте. Однако примерно половина из них теоретически может быть найдена алгоритмами на основании построения множества кандидатов.

Исследование также показало, что 1.41 слово, указанное как ключевое, встречается как подстрока в заголовке заметки, что, считаем, довольно много и составляет почти половину тех слов, которые вообще теоретически могут быть найдены ранжированием подстрок текста.

Что же касается средней длины тега, или, ключевого словосочетания, она составляет 1.28 слова. Таким образом мы эмпирически убедились в том, что

ключевые слова, вводимые пользователем имеют относительно небольшую длину и ни в коем случае не являются фразами.

4.2.2 Отличия для случая автоматической разметки

Напомним, что в рассмотрении различного рода наборов исходных данных, были также рассмотрены и те наборы, которые не были размечены людьми, однако используют разметку с помощью мощных алгоритмов на стороне сервера на основании рекуррентных нейронных сетей. Данные наборы также были исследованы на предмет того, как различается разметка человеком и разметка нейронной сетью.

Для примера был взят набор исходных данных, полученный сбором статей со свободной энциклопедии “Википедия” на английском языке. Набор состоит из 7639 текстов.

Было найдено два основных отличия.

Первое отличие состоит в количестве генерируемых слов, которые распознаются как ключевые. В случае с автоматической разметкой, это количество составило 9.75 слов против 6.79 для случая блогов. Выходит так, что алгоритмы такого плана фиксируют число генерируемых ключевых слов равным десяти и выводят меньше только в случае крайне коротких текстов.

Второе отличие более интересно. Оно состоит в природе генерируемых данными алгоритмами ключевых слов. Только 1.00 слово, сгенерированное на стороне сервера без информации о заголовке встречается как подстрока в заголовке статьи, из чего следует, что люди чаще выносят в заголовок ключевые слова, чем машины выделяют ключевые слова из заголовка. Что еще интереснее, 9.63 слова (из 9.75) являются элементами множества кандидатов, которое генерируют алгоритмы для распознавания слов на клиенте. В то же время, люди не склонны данным образом теггировать свои заметки. В этом заключается основная разница между разметкой людьми и большей частью алгоритмов.

4.3 Оценка известных алгоритмов на построенных выборках

В данном подразделе будет установлена методология проведения вычислительного эксперимента для оценивания построенных алгоритмов. Будет также произведено оценивание двух основных аналогов для выделения ключевых слов на стороне клиента – RAKE [1] и TextRank [2], что необходимо для

соотношения результатов алгоритмов, построенных в диссертации с известными аналогами.

4.3.1 Методология оценивания

Перед оцениванием известных алгоритмов для выделения ключевых слов на стороне клиента ставится следующая задача: оценить качество генерации ключевых слов на собранных наборах исходных данных, исключив возможные ошибки, связанные с неправильной трактовкой работ или реализацией описанных результатов.

Данная цель исключает, таким образом, возможность самостоятельной реализации описанных алгоритмов. Поэтому были использованы наиболее авторитетные реализации от исследователей, в том числе – авторов статей, сильно связанных с данными алгоритмами.

Выборки же были разбиты на тренировочную и проверочную части следующим образом:

- Первые 1000 текстов были отнесены к обучающей выборке;
- Все оставшиеся тексты были отнесены к проверочной выборке, которая не должна участвовать в подборе гиперпараметров никакого из алгоритмов.

Наконец, функцией оценивания результата на выборке было принято суммарное количество правильно определенных ключевых слов, то есть тех, которые встретились в разметке.

В приведенных результатах был использован набор исходных данных, собранный с ресурса WordPress в главе 2, как наиболее интересующий: при наличии возможности сравнения с полностью размеченными людьми текстами, этот вариант наиболее предпочтителен.

4.3.2 Оценивание алгоритма RAKE

В качестве эталонной реализации алгоритма RAKE была взята реализация из открытого репозитория GitHub автора с логином aneesha, доступно по адресу [12]. Данная реализация на момент 25 ноября 2018 года имеет следующие показатели: 54 наблюдателя, 652 вхождения в раздел “избранное” других пользователей, 456 “форков” – клонов, которые делаются с целью сохранения копии кода для дальнейшей работы и исследований. Реализация с минимальными изменениями была портирована на язык программирования Python версии 3.5.2.

Были получены следующие результаты:

- Обучающая выборка: **163**
- Проверочная выборка: **5629**

Возникает вопрос о том, почему полученные результаты столь невысоки. Ответом видится тот факт, что алгоритм Rapid Automatic Keyword Extraction, ввиду своей функции ранжирования, дает преимущество тем кандидатам, которые имеют б**о**льшую длину. В то же время, как показано в подразделе 4.2.1, среднее количество слов в теге составляет 1.28 слова, а часто ключевым может быть и вовсе одно слово.

Таким образом, возможным представляется сделать следующий вывод: *как правило, алгоритм RAKE генерирует ключевые слова отличающиеся от тех, что ставят люди – авторы текста, если это произвольный текст на естественном языке.*

4.3.3 Оценивание улучшенной версии алгоритма TextRank

Отметим, что алгоритм TextRank был впервые описан в работе [2] в 1998 году, однако позже был улучшен в работе “Variations of the Similarity Function of TextRank for Automated Summarization” [11] аргентинских исследователей Федерико Барриоса (Federico Barrios), Федерико Лопеса (Federico Lopez), Луиша Аргерича (Luis Argerich), Росита Вахенхаузер (Rosita Wachenchauzer) в 2017 году. В работе рассматриваются различные варианты функции похожести для алгоритма TextRank, некоторые из которых позволяют добиться существенного прироста в качестве алгоритма по сравнению с предыдущей известной версией.

Работа примечательна еще и тем, что авторы приводят свою реализацию отраженных ими идей. Она доступна в системе открытых репозиториев GitHub по ссылке [13] и имеет следующие показатели по состоянию на 25 ноября 2018 года: 25 наблюдателей, 453 добавления в “избранное”, 135 “форков”. Реализация также доступна в виде пакета в открытом репозитории пакетов для языка программирования Python PyPI под именем `summa`.

По результатом запуска на выборках были получены следующие результаты:

- Обучающая выборка: **617**
- Проверочная выборка: **18615**

Данные результаты существенно превосходят алгоритм RAKE – примерно в три с половиной раза. Это может быть обусловлено спецификой получаемых алгоритмом результатов: алгоритм TextRank объединяет два ключевых слова в одну

фразу только в тех случаях, когда оба из них присутствуют в списке первых выбранных, а также встречаются в рассматриваемом тексте последовательно. Данный подход приводит к тому, что объединения отдельных слов в фразы редки, что в большей мере соответствует специфике тегов в заметках в блогах.

Тем не менее, данные результаты также пока далеки от теоретически достижимого результата, что оставляет простор для исследования различных алгоритмов генерации ключевых слов на стороне клиента.

ГЛАВА 5. АЛЬТЕРНАТИВНЫЙ ПОДХОД К ЗАДАЧЕ ВЫДЕЛЕНИЯ КЛЮЧЕВЫХ СЛОВ НА СТОРОНЕ КЛИЕНТА

5.1 Основные положения

5.1.1 Параметры для оптимизации

В подразделе 4.1 на примере записи в блоге были рассмотрены факторы, влияющие на принадлежность слова или фразы множеству ключевых. Соберем воедино эти факторы:

- Факт вхождения и количество вхождений данного слова или фразы в заголовок;
- Количество вхождений слова или фразы в анализируемый текст;
- Позиция первого вхождения рассматриваемого слова или фразы в рассматриваемый текст, пропорционально которой назначается штраф;
- Количество вхождений в текст отдельных слов рассматриваемой фразы;

Данное множество факторов может быть расширено, однако будет использовано в данном виде в первом построении алгоритма. Отметим, что фактор – количество вхождений в текст используется в качестве основного в алгоритме RAKE.

Предлагается сделать веса, назначаемые факторам гиперпараметрами, которые возможно оптимизировать, например, с помощью алгоритмов комбинаторной оптимизации, такими как метод локальных оптимизаций или перебор по сетке параметров.

5.1.2 Построение множества кандидатов в ключевые слова

Данные взвешенные факторы будут использованы для ранжирования кандидатов во множество ключевых слов. Как было показано в подразделе 4.2.1, средняя длина выставленного человеком тега составляет 1.28 слова, поэтому в данном алгоритме при построении множества кандидатов будем рассматривать все различные строки из **не более трех** последовательных слов анализируемого текста, не содержащих между собой знаков пунктуации и стоп-слов. Стоит отметить, что существует множество подходов к выделению стоп- или шумовых слов. На практике, однако, часто используются их списки такие, как например список

стоп-слов Фокса (Fox stoplist в англ. литературе), который был получен Кристофером Фоксом (Christopher Fox) как результат работы “A Stop List for General Text” [14] в 1989 году. Данный список будет использован и в нашем алгоритме в версии для английского языка. Данное решение мотивировано тем, что его использование показывает эффективность в различных работах по анализу текстов а также тем, что выбранные в подразделах 4.3.2 и 4.3.3 реализации алгоритмов также используют список Фокса для фильтрации шумовых слов, что позволит создать более равные условия при сравнении алгоритмов. Список Фокса состоит из 425 слов.

5.1.3 Ранжирование кандидатов

Для ранжирования кандидатов во множество ключевых слов, каждому из них назначается балл, соответствующий релевантности кандидата C множеству ключевых слов. Балл вычисляется следующим образом:

$$score_C = \sum f_{C,i} \times F_i \quad (1)$$

Здесь F_i соответствует весу фактора, который оптимизируется для наилучшего качества, а $f_{C,i}$ – значению данного фактора для кандидата C в рассматриваемом тексте.

После того, как веса для всех кандидатов вычислены, производится сортировка кандидатов по убыванию назначенного балла. Выбираются первые 10 кандидатов и объявляются ключевыми словами.

5.2 Подходы к оптимизации гиперпараметров

Как было описано в нескольких предыдущих подразделах, выбор ключевых слов зависит от весов, которые даются факторам, описанным в подразделе 5.1.1. Возникает задача поиска таких их значений, которые дали бы наилучший результат, то если наилучшую полноту множества ключевых слов при том, что их количество в разделе 5.1.3 мы ограничили десятью.

Прежде всего, зададим метрику, оптимизацию которой будем проводить. В качестве данной метрики выберем просто абсолютное количество ключевых слов, которые совпали с теми, что даны для соответствующего текста в выборке. Данная метрика устроит нас, так как позволит решить задачу сравнения результата с

имеющимися аналогами, а также получить и общее представление о качестве решения задачи.

Оптимизация гиперпараметров в данной задаче подразумевает под собой выбор таких значений вектора F из формулы (5.3.1, 1), для которых сумма значений метрики качества по всем текстам из контрольной выборки даст наилучший результат, то есть наибольшее число совпавших с ответом ключевых слов. Напомним, что методика разбиения наборов исходных данных на выборки была описана в подразделе 4.3.1.

Было рассмотрено три подхода к оптимизации обозначенной метрики:

- Полностью случайный метод: генерация большого количества векторов F с оценкой результата при каждом из них на обучающей выборке.
- Метод локальных оптимизаций: выбор случайной компоненты вектора F с ее незначительным изменением и перемещением текущего оптимизируемого объекта в новую точку, если значение метрики качества на измененном векторе F лучше, чем значение данной метрики на векторе F без изменений;
- Метод перебора по решетке параметров: каждой из компонент вектора F задаются минимальное и максимальное возможное принимаемое значение, а также шаг. Таким образом в пространстве размерности $|F|$ полученные разбиения задают сетку, во всех узлах которой затем вычисляется значение целевой функции на обучающей выборке. Из этих значений, в предположении о том, что минимизирующее эмпирический риск решение является достаточно качественным, выбирается наибольшее, а в качестве вектора F - точка, в которой данное значение было достигнуто.

Отметим, что третий метод позволил достичь наилучшего результата на обучающей выборке, поэтому был использован в конечном итоге для получения окончательных значений вектора F на выборке из текстов ресурса WordPress на английском языке.

5.3 Технические детали реализации

Подбор параметров и тестирование алгоритма на обучающей выборке были реализованы на языке C++ стандарта 2017 года, в отличии от предыдущих практических результатов. Этот выбор мотивирован тяжелым переборным характером поставленной задачи, при котором решение на языке Python будет

работать значительно медленнее: исследования показывают, что данный язык проигрывает языку программирования C++ в скорости работы в 100-200 раз.

Язык C++ характерен также меньшим набором инструментов для работы с текстом, нежели язык Python. Разбиение текста на лексические единицы, его фильтрация и нормализация были реализованы нами самостоятельно ввиду отсутствия подходящих вариантов в средствах языка. В работе с текстом также необходимо обратить внимание и на тот факт, что стандартные функции для приведения языка к нижнему регистру не работают для символов не из таблицы символов ASCII, что не покрывает алфавитов всех языков, кроме английского. Данная проблема была решена использованием модуля `locale` из библиотеки `boost`, являющейся популярным средством для расширения возможностей данного языка.

Оптимизируемый вектор параметров состоял из пяти компонент:

- Вес фактора, соответствующего числу вхождений кандидата в текст: перебирался от 0 до 1000 с шагом 100;
- Вес фактора, соответствующего числу вхождений кандидата в заголовок, перебирался от 0 до 2000 с шагом 200;
- Вес фактора, соответствующего штрафу за позицию первого вхождения кандидата в текст, перебирался от 0 до 5 с шагом 1;
- Вес фактора, соответствующего числу вхождений отдельных слов кандидата в текст для кандидатов длины 2, перебирался от 0 до 20 с шагом 4;
- Вес фактора, соответствующего числу вхождений отдельных слов кандидата в текст для кандидатов длины 3, перебирался от 0 до 20 с шагом 4.

Данный набор разбиений дает сетку с $11 \times 11 \times 6 \times 6 \times 6 = 26136$ узлами. Перебор значений во всех узлах занял 19 минут на ЭВМ с процессором Intel CORE i7 vPro и объемом оперативной памяти DDR3, равным 8 гигабайтам.

Стоит отметить, что важной оптимизацией является предпросчет значений каждого фактора для каждого текста до начала оптимизации гиперпараметров. Он может быть сделан за время $O(N \log N)$ с использованием встроенных в язык C++ красно-черных деревьев (реализованы в классах `std::map`, `std::set`) для хранения промежуточной информации по кандидатам, либо с асимптотикой сложности $O(N)$ с использованием встроенных в стандартную библиотеку типов языка C++ хэш-таблиц, реализованных в классах `std::unordered_map`, `std::unordered_set`. Отметим, что несмотря на лучшую асимптотику, решение с использованием

хэш-таблиц в данном случае оказывается хуже из-за недостаточно больших объемов текстов.

5.4 Вычислительный эксперимент. Выводы

В результате вычислительного эксперимента лучшим оказался вектор весов факторов со следующими параметрами:

- Вес вхождения в текст: **200**;
- Вес вхождения в заголовок: **2000**;
- Штраф за позицию первого вхождения: **1**;
- Бонус за вхождение в текст отдельных слов для кандидатов длиной в два слова: **4**;
- Бонус за вхождение в текст отдельных слов для кандидатов длиной в три слова: **4**.

На тренировочной выборке наилучший результат был достигнут с описанными выше значениями гиперпараметров. Результат на тренировочной выборке равен **1237**.

На контрольной выборке было определено правильно **35320** слова. Этот результат почти в 2 раза превосходит результат, показанный улучшенной версией алгоритма TextRank, приведенный в подразделе 4.3.3 и в шесть раз – результат, показанный алгоритмом RAKE, приведенный в подразделе 4.3.2.

Стоит отметить, что данный результат был достигнут с большим количеством информации, чем используется в алгоритмах RAKE и TextRank. Так, в данных алгоритмах не используется информации о заголовке текста, которая, как показано в подразделе 4.1, может являться сильным индикатором, помогающим определить принадлежность кандидата множеству ключевых слов. Важным отличием также является тот факт, что в алгоритме RAKE с одинаковым весом учитываются вхождение кандидата в текст целиком и вхождение его отдельных слов в текст. Здесь же это гиперпараметры, которые должны быть настроены и оптимизированы.

Однако с другой стороны, требования алгоритма позволяют адаптировать его к подавляющему большинству задач реального мира: наличие заголовка является желательным, а не обязательным требованием, хотя из любой веб-страницы он может быть извлечен. Также стоит отметить, что множества ключевых слов, которые были использованы в обучающей и контрольной выборках были практически непересекающимися, что свидетельствует о том, что однажды настроив гиперпараметры алгоритма на небольшой выборке текстов используемого языка,

они могут быть использованы в практически всех его текстах, в том числе, и текстах той тематики, которая не присутствовала в выборке для настройки гиперпараметров.

Таким образом алгоритм за счет использования обычно имеющейся дополнительной информации сумел показать превосходящие два популярных аналога результаты в бенчмарке на размеченных тегами записях в блоговых ресурсах, что говорит о его возможной применимости. Работа, однако, на этом не закончена: необходима адаптация алгоритма для различных языков, которая потребует и исследования методов выделения стоп-слов в различных натуральных языках. Также важной частью работы является и оформление алгоритма в чистовом, готовом для реализации виде, который может быть использован в репозиториях пакетов языков программирования. Также необходимо исследовать поведение алгоритма на расширенном множестве гиперпараметров и возможность применения машинного обучения.

ГЛАВА 6. ПРИМЕНЕНИЕ МЕТОДОВ МАШИННОГО ОБУЧЕНИЯ ДЛЯ ЗАДАЧИ ВЫДЕЛЕНИЯ КЛЮЧЕВЫХ СЛОВ

В главах 4 и 5 были рассмотрены общеизвестные методы для выделения ключевых слов в тексте, затем был приведен альтернативный подход, основывающийся на рассмотрении более широкого пространства признаков и оптимизации гиперпараметров, задающих веса этих признаков. Этот метод, однако, использовал простой оптимизационный алгоритм для настройки параметров. В данной главе будут рассмотрены подходы к оптимизации схожих параметров, основанные на методах машинного обучения.

6.1 Общие положения

При рассмотрении возможности применения методов машинного обучения к задаче анализа текста стоит отметить два основных варианта. Наиболее популярными на момент написания работы являются методы, построенные на основе Embedding-подхода в нейронных сетях. Данный подход основывается на концепции векторного представления слова, в которой каждому слову из некоторого словаря, соответствующего естественному языку ставится в соответствие вектор, описывающий данное слово.

Самым популярным примером Embedding-подхода является работа Миколова, Чена, Коррадо и Дина “Efficient Estimation of Word Representations in Vector Space” [15], которая дала начало технологии word2vec. Интересной особенностью, давшей популярность этой работе стала интерпретируемость результата: так, например, род объекта мог быть изменен путем выполнения простых алгебраических операций на построенных векторах. Полученные алгоритмы были развиты и далее: например, при помощи концепции мешка слов (в англоязычной литературе – “bag of words”), были расширены до анализа цельных текстов, что получило названия doc2vec или paragraph2vec. Случай же перевода одного слова в векторное представление был также проанализирован и решен с большим качеством с помощью подхода ELMo Embeddings, который был описан в статье Питерса, Ньюмана, Айвьера, Гарднера, Кларка, Ли и Зеттлмоейера “Deep contextualized word representations” [16], которая

на конец 2018 является решением, имеющим наибольшее качество из известных (т. н. “state of the art”-решение в англоязычных источниках).

Данные решения, однако, не являются полностью подходящими для решения задачи на стороне клиента. Так, решение описанное выше имеет репозиторий [17], из которого возможно скачать пре-тренированные модели для векторного представления слов на английском языке. Размеры этих моделей велики – даже сжатая модель размера “small” – наименьшего из доступных имеет вес 60 мегабайт, что делает очень сложным ее использование в условиях, когда доступно небольшое количество ресурсов – например, на веб-страницах. Также отметим, что такой размер модели делает сложным расширение эвристического алгоритма на случаи нескольких языков, так как для каждого из них будет необходима подготовка своей модели, также занимающей немало места.

Поэтому имеет смысл рассмотрение других подходов. В частности, рассмотрим варианты использования классических алгоритмов машинного обучения. В частности, в данной главе будет рассмотрено использование ансамблей решающих деревьев (решающие леса), метода ближайших соседей, метода опорных векторов и логистической регрессии.

6.2 Сведение к задаче классификации. Построение обучающей и тестовой выборок

Классические методы машинного обучения не могут быть применены к данным рассматриваемой нами задачи в изначальном виде. Это вызвано тем, что даже нормализованный по правилам из подраздела 2.2.1 текст, разбитый на токены, не может быть подан на вход никакому из рассматриваемых алгоритмов, так как во-первых, слова несравнимы между собой, во-вторых – классические алгоритмы дают на выходе вероятности принадлежности данного им объекта рассматриваемому классу, а не список наиболее подходящих под какое-то условие подобъектов одного объекта (применительно к задаче выделения ключевых слов – подстрок поданной на ввод строки). Это приводит к необходимости преобразования входных данных.

Заметим, что данное преобразование может быть реализовано использованием в качестве признаков тех факторов, которые использовались при ранжировании объектов из множества кандидатов в главе 5. Так, объектами классификации можно сделать объекты из множества кандидатов, а признаками – факторы описанные в подразделе 5.1.1. Отметим также сразу и проблему, возникающую при данном

подходе. Проблема связана с тем, что факторы для ранжирования зависят от размера текста, поэтому, объединив все факторы всех текстов в одну выборку можно получить ситуацию, в которой малозначимый объект-кандидат из большего текста имеет большие значения решающих факторов, чем ключевое слово из меньшего по размеру текста. Таким образом, необходимо нормировать частоты некоторым образом, не давая описанным ситуациям случиться. Для этого, добавим всем объектам еще один признак: количество слов в тексте, из которого был выбран соответствующий объект-кандидат в ключевые слова.

Классом же в данной задаче будет факт принадлежности заданного объекта-кандидата множеству ключевых слов в том тексте, из которого данный объект был выбран. Отметим, что одно и то же словосочетание может быть выбрано в качестве объекта-кандидата несколько раз, если оно встречается в разных текстах. В этом случае, оно встретится в выборке столько раз, в скольких текстах оно встретилось. Также, на практике у таких объектов будет различаться и последний признак, а именно – признак количества слов в тексте. Принадлежность всех таких кандидатов одному тексту также не гарантируется.

Таким образом, из выборки статей ресурса WordPress, размеченных их авторами и построенной в главе 2 была построена выборка из 5,418,512 образцов. Данная выборка была затем уменьшена до 4,407,526 образцов путем исключения из рассмотрения тех текстов, в которых множество кандидатов не содержит ни одного ключевого слова. Отметим факт несбалансированности классов в данной выборке, обусловленный малым количеством ключевых слов относительно множества кандидатов. Так, только 79481 объект выборки имел положительный класс.

Выборка была далее разбита на обучающую и контрольную части согласно описанной в подразделе 4.3.1 методологии. Данное разбиение дает возможность сравнить результаты с теми, что были получены в данной работе ранее.

6.3 Оценивание качества алгоритмов машинного обучения

Необходимо выбрать метрику для сравнения результатов алгоритмов классификации между собой. Отметим, что ввиду несбалансированности классов, метрика точности не является применимой: классификатор, выдающий всегда отрицательный класс будет иметь точность около 98.5%, что не соответствует в действительности качественному классификатору.

Поэтому, основной метрикой для сравнения алгоритмов классификации между собой предлагается использовать площадь под кривой ошибок. Важным в

рассматриваемой задаче ее свойством является инвариантность относительно отношения цены ошибки I и II рода. Отметим, что использование данного метода дает еще одно требование к применяемым алгоритмам – возможность для каждого объекта не просто предсказать его класс, но и вероятность, с которой объект относится к этому классу.

6.4 Вычислительный эксперимент

На основании приведенных в подразделах 6.1-6.3 положений, можно провести вычислительный эксперимент с несколькими алгоритмами машинного обучения. Для целей вычислительного эксперимента использовался скриптовый язык программирования Python версии 3.5.2, а также библиотека scikit-learn версии 0.20.1. Преимуществом данной библиотеки является разнообразие реализованных алгоритмов, что позволяет быстро опробовать несколько подходов.

Первым был опробован алгоритм, основывающийся на лесе решающих деревьев. В целях эксперимента было зафиксировано число деревьев равное 100. Для разбиения выборки в вершине дерева на несколько подвыборок, соответствующих прямым потомкам этой вершины был использован критерий Джини. Разбиение вершины продолжалось до тех пор, пока в ней имелось более двух элементов выборки. В результате, данный метод дал следующие результаты на контрольной выборке:

- Точность: 0.9787298137761323
- Площадь под кривой ошибок: 0.7582779865832676

Далее был опробован алгоритм, основывающийся на методе ближайших соседей. Для поиска ближайшего соседа к точке, был использован алгоритм Ball Tree, имеющий логарифмическую сложность поиска. Число ближайших соседей было выбрано равным 23; использована Евклидова метрика как показатель удаленности объектов. Метод дал следующие результаты на контрольной выборке:

- Точность: 0.9819373663990071
- Площадь под кривой ошибок: 0.6741857631379751

Отдельно отметим тот факт, что точность метода, основанного на поиске ближайших соседей выше, чем у метода, основанного на ансамбле решающих деревьев, однако при этом площадь под кривой ошибок значительно ниже. Это обусловлено тем, что метод на основании анализа ближайших соседей в гораздо большей степени склонен в данной задаче относить объекты к отрицательному классу.

Далее был опробован алгоритм, основывающийся на использовании метода опорных векторов. В отличии от предыдущих двух алгоритмов, он является более вычислительно сложным. В частности, обучение имеет сложность выше, чем квадратичная от числа образцов, что делает невозможным применение метода на объемах обучающей выборки. Поэтому, для применения данного метода, из обучающей выборки было выбрано случайное подмножество двух с половиной тысяч объектов. После обучения на таком небольшом подмножестве с силой регуляризации C=1.0, метод дал следующие результаты:

- Точность: 0.9819545203434371
- Площадь под кривой ошибок: 0.6332001990095096

Относительно низкие показатели качества можно объяснить тем фактом, что обучающая выборка была очень сильно прорежена для того, чтобы иметь возможность обучить классификатор за разумное время.

Наконец, был опробован алгоритм на основании метода логистической регрессии с l2-регуляризацией (регуляризация Тихонова) с силой регуляризации, равной 1.0. Были получены следующие результаты:

- Точность: 0.9817070805696725
- Площадь под кривой ошибок: 0.8466516896844374

Таким образом, метод на основании логистической регрессии показал себя лучшим образом в данной задаче. Интересно отметить, что как и в случае простого ранжирования, который был рассмотрен в пятой главе, полученная модель является линейной.

Для однозначно лучше себя показавшего метода логистической регрессии имеет смысл более детальный перебор параметров. Был проведен перебор параметра силы регуляризации C по степеням 10 от 10^{1} до 10^{-5} включительно а также весов факторов из следующих вариантов: равные веса и сбалансированные – выбранные таким образом, что суммарные веса по всем образцам совпадают.

Наилучшей оказалась комбинация со сбалансированными весами и силой регуляризации C равной 1. Данные параметры дали следующие результаты:

- Точность: 0.8199768726569042
- Площадь под кривой ошибок: 0.8615924409536379

В данном случае сильно заметна потеря в точности по сравнению с остальными решениями – она вызвана балансировкой весов классов, которая приводит к увеличению ложноположительных ответов. Несмотря на это, главная

выбранная метрика – площадь под кривой ошибок заметно увеличивается, что свидетельствует о более высоком качестве данного классификатора. Поэтому в дальнейшем, будем рассматривать именно алгоритм с нормализацией весов.

6.5 Применение полученного алгоритма к решению задачи

В предыдущем подразделе путем вычислительных экспериментов было установлено, что наиболее хорошо с точки зрения площади под кривой ошибок себя показывают модели на основе логистической регрессии. Построенный алгоритм, однако, еще не решает задачу нахождения ключевых слов. В данном подразделе он будет преобразован его для решения рассматриваемой в магистерской диссертации задачи.

Для этого полученная модель может быть сериализована с помощью стандартных средств языка программирования. Будем пользоваться предсказанной вероятностью вхождения объекта-кандидата в список ключевых слов как функцией ранжирования. Как и в предыдущем решении, будем выбирать первые 10 кандидатов относительно этого фактора. Это позволит сравнить качество полученного решения с качеством решения из главы 5.

В результате тестового запуска, на контрольной выборке было определено правильно **39081** слов, что превосходит предыдущее полученное решение на **3671** слово, или примерно на **10%**.

Таким образом, в данной главе путем применения методов машинного обучения удалось установить, что наиболее подходим для данной задачи является метод логистической регрессии с балансировкой весов классов. Данный метод, будучи применен к сведенной к задаче классификации задаче о выделении ключевых слов дал десятипроцентный прирост к метрике качества на контрольной выборке и на момент написания главы является наиболее точным из рассмотренных в данной магистерской диссертации.

ГЛАВА 7. ПОДХОДЫ К ВЫДЕЛЕНИЮ СТОП-СЛОВ НА СТОРОНЕ КЛИЕНТА

В предыдущих главах нами были построены эффективные алгоритмы для выделения ключевых слов из текстов на английском языке. Интерес представляет также и более общая задача: выделение ключевых слов в текстах на произвольных языках.

В то же время, решение поставленной задачи на произвольном языке всегда требует некоторых усилий по фильтрации стоп-слов данного языка. Стоп-словами или шумовыми словами называются слова, имеющие некоторые характеристики ключевых слов, например, высокую частотность, однако не являющиеся таковыми из-за неинформативности. Наример, стоп-словами являются предлоги или междометия.

7.1 Общая характеристика стоп-слов (шумовых слов)

Шумовые слова могут делиться на общие и зависимые.

К общим можно отнести предлоги, суффиксы, причастия, цифры, частицы, междометия и т. п. Общие стоп-слова всегда исключаются поисковыми сервисами из запросов (за исключением поиска по строгому соответствию поисковой фразы), а также игнорируются при построении инвертированного поискового индекса.

Зависимые стоп-слова зависят от поисковой фразы. Идея многих алгоритмов, в частности, заключается в том, чтобы по-разному учитывать отсутствие просто слов из запроса и зависимых стоп-слов из запроса в найденном документе. Например, при запросе Пушкин Александр Сергеевич, поисковой машине есть смысл отобразить все документы, содержащие:

- Пушкин, Александр, Сергеевич
- Пушкин, Александр
- Пушкин, Сергеевич
- Пушкин

Но вряд ли есть смысл отображать документы, содержащие только:

- Александр, Сергеевич
- Александр
- Сергеевич

То есть, в данном запросе шумовыми словами являются Александр и Сергеевич. Зависимые стоп-слова отличаются тем, что в поисковом запросе их

следует учитывать только при наличии в искомом документе значимых ключевых слов.

7.2 Использование списков стоп-слов

Самым простым решением задачи выделения стоп-слов является использование списков стоп-слов таких, как список стоп-слов Фокса [14] для случая английского языка. В ходе исследования задачи не было найдено списков стоп-слов для других языков, по которым имелась бы публикация. Однако, существует довольно большое количество источников, которые могут быть использованы.

Так, например, библиотека для работы с текстами на естественном языке для языка программирования Python nltk [18], также известная под названием Natural Language Toolkit предоставляет разработчикам метод stopwords, находящийся в пакете nltk.corpus. Данный метод предоставляет списки стоп-слов для 21 языка, среди которых английский, русский, испанский, итальянский и прочие.

Также существуют и менее авторитетные проекты с открытым исходным кодом, в которых собрано еще большее количество иностранных языков. Например, проект stopwords-json [19] на платформе Github, предоставляющий стоп-слова по 50 различным языкам.

Таким образом, существует множество источников списков стоп-слов, способных предоставить список для практически любого естественного языка. Отметим также, что данный метод является наиболее точным из рассмотренных, так как список составляется и модерируется людьми. Тем не менее, далее будет рассмотрено несколько подходов к построению списка стоп-слов для ситуации, когда заранее данный список недоступен.

7.3 Подходы, основанные на предобучении

Самым известным и, в то же время, довольно эффективным подходом, основанным на предобучении является подход на основе анализа значений метрики TFIDF для слов рассматриваемого текста.

Метрика TF – означающая Term Frequency (частоту терма) является значением количества встречаемостей рассматриваемого слова в рассматриваемом тексте. Метрика DF – Document Frequency, означающая документную частоту является значением количества различных документов коллекции, в которых встретился рассматриваемый терм. Одновременно со значением метрики DF рассматривается

также значение метрики IDF – Inverse Document Frequency, получаемое следующим образом:

$$IDF = \frac{1}{DF}$$

Само же значение метрики TFIDF для рассматриваемого слова в документе является отношением его частоты встречаемости к его документной частоте, таким образом:

$$TFIDF = \frac{TF}{DF} = TF \times IDF$$

Утверждается, что слова с высоким значением метрики TFIDF в тексте являются, с высокой долей вероятности, ключевыми словами данного текста в рассматриваемой коллекции. Это наблюдение основывается на соображениях, исходя из которых документная частота ключевых слов отдельно взятого документа невелика, в то же время, в самом документе эти слова, как характеризующие содержание, должны встречаться довольно часто. Отметим, что несмотря на тот факт, что данный метод применим для выделения ключевых слов, он неприменим для выделения ключевых слов на стороне клиента, так как для получения значения метрики IDF необходимо хранить словарь со всеми словами рассматриваемого языка и их обратными документными частотами.

В то же время, для стоп-слов значение метрики DF будет стремиться к размеру коллекции, так как стоп-слова такие как предлоги и местоимения встречаются в практически каждом тексте. Поэтому, при достаточно большой коллекции, по которой вычисляется метрика IDF, относительно высокое значение метрики TF будет нивелировано стремящемся к размеру коллекции значением метрики IDF. Таким образом можно заключить, что при достаточно большой коллекции текстов взяв слова с наименьшими значениями метрики TFIDF (усредненной по всем текстам) можно получить список стоп-слов в заданной коллекции текстов.

Данный подход был реализован в пакете masters.stop_words.tool и позволяет быстро сгенерировать список стоп-слов по заранее заданной коллекции текстов, что может быть полезно при расширении описанного в Главах 5, 6 алгоритма для случая новых языков с использованием заранее сгенерированного списка стоп-слов.

7.4 Эвристические подходы для произвольных языков без предобучения

Несмотря на тот факт, что для многих языков существуют готовые списки стоп-слов, а также алгоритмы для выделения стоп-слов по заранее заданной

коллекции – как, например, рассмотренный в предыдущем подразделе, некоторый интерес также представляют алгоритмы для выделения стоп-слов без предобучения и работающие, в теории, для произвольных естественных языков. Подобный алгоритм, например, бегло рассмотрен в работе [1] и основывается на построении графа по рассматриваемому тексту. Вершинами данного графа авторы предлагают выбрать слова, а ребрами – связи между этими словами. Положим, что рассматриваемом графе существует ребро между двумя вершинами, если слово, соответствующее одной из вершин идет непосредственно после слова, соответствующего другой из вершин.

Построив граф описанным выше способом, выберем из него некоторую долю вершин с наибольшей степенью. Эти вершины и будут соответствовать стоп-словам в данном тексте.

Отметим однако, что хоть и данный метод позволяет выделить из текста кандидатов в стоп-слова, он также подвержен ложноположительным срабатываниям, из-за которых в качестве стоп-слов могут быть определены и ключевые слова текста, так как встречаемость в тексте ключевых слов также будет высока. Таким образом, данный метод может быть использован, но ввиду возможных критичных для выделения ошибок, лучшим представляется вариант с использованием списков стоп-слов – загруженных заранее или же сгенерированных по коллекции документов методом из предыдущего подраздела.

ГЛАВА 8. РАСШИРЕНИЕ АЛГОРИТМА НА СЛУЧАЙ ДРУГИХ ЯЗЫКОВ

8.1 Применение алгоритма для случая испанского языка

Алгоритм, полученный в главе 6 был реализован и протестирован для случая английского языка. В то же время, необходимо рассмотреть возможность использования алгоритма и для других языков. В данной главе будет рассмотрена переносимость алгоритма для текстов на испанском языке.

Для построения выборки был использован ресурс WordPress. Аналогично способу, рассмотренному во второй главе, была построена обучающая и тестовая выборка для блоговых записей на испанском языке. Для этого в качестве базовых тегов для поиска записей были взяты 10 тысяч наиболее часто встречаемых в испанском языке слов. Таким образом удалось получить 36516 статей, из которых 1000 была использована для настройки гиперпараметров, а остальные 35516 были использованы для оценки качества алгоритма.

В качестве списка стоп-слов был выбран список, предоставляемый библиотекой [19] для испанского языка. Отметим, что кроме замены списка стоп-слов со списка Фокса (английского списка) на испанский список стоп-слов, изменений не потребовалось.

Далее были проведены вычислительные эксперименты. Для предложенного подхода на основании машинного обучения было рассмотрено два случая:

- Случай с обучением параметров алгоритма с помощью логистической регрессии на одной тысяче заметок полученного корпуса статей на испанском языке;
- Случай с использованием параметров алгоритма, полученных с помощью логистической регрессии для случая корпуса статей на английском языке, рассмотренного в Главе 6.

Также были проанализированы, аналогично Главе 4, результаты алгоритмов RAKE и TextRank на заданном корпусе статей.

8.2 Результаты. Сравнительный анализ

Прежде всего, приведем результаты алгоритмов RAKE и TextRank, полученные на построенном корпусе статей.

Алгоритмом RAKE были получены следующие результаты:

- Обучающая выборка: 142 ключевых фразы;
- Тестовая выборка: 3829 ключевая фраза.

Алгоритмом TextRank были получены следующие результаты:

- Обучающая выборка: 836 ключевых слов;
- Тестовая выборка: 25980 ключевых слов.

Далее приведем результаты, полученные предложенным в данной работе алгоритмом. Для получения данных результатов был использован метод логистической регрессии на построенной способом, приведенным в Главе 6 задаче бинарной классификации.

Прежде всего, приведем результат, полученный моделью, тренировка которой осуществлялась на текстах на английском языке с использованием списка английских стоп-слов, то есть, списка Фокса. Данной моделью на тестовой выборке правильно было определено 31489 ключевых фраз.

Рассмотрим теперь модель, с нуля натренированную на тренировочной выборке, состоящей из специально отведенных 1000 статей на испанском языке из полученной выборки. Данная модель показала несколько лучший результат: 33296 ключевых фраз.

Отметим, что несмотря на необходимость тренировки, данные результаты показывают тот факт, что модель, обученная для одного языка может быть успешно использована и в других. Так, в испанском языке, обучение модели на текстах этого языка дало при тестировании прирост только в пять процентов.

Отметим также и тот факт, что полученные результаты численно не превосходят аналогичные результаты для случая английского языка несмотря на больший размер коллекции статей на испанском языке. Это объясняется тем фактом, что тексты на испанском языке труднее поддаются данному алгоритму, в частности, из-за наличия нескольких падежей у местоимений а также наличия большого количества форм у глаголов.

8.3 Замечания о расширяемости алгоритма для других языков

При расширяемости алгоритма на случаи языков, отличных от английского и испанского, необходимо учитывать особенности данного языка. Так, в русском языке у существительных имеется шесть падежей, что заметно осложняет алгоритм подсчета частот слов, ввиду необходимости учитывать разные падежи одного и того

же существительного как одно слово. Данная проблема решается алгоритмами стемминга. Так, например, Портер (Porter) еще в 1980 году предложил в своей работе “An algorithm for suffix stripping” [20] алгоритм для стемминга слов. Данный алгоритм был расширен на случаи многих языков, в том числе и русского, и доступен в библиотеке nltk.

Также, в случае, например, китайского языка, необходима реализация разбиения текста на фрагменты и работы с этими фрагментами как со словами. В случае японского языка такое разбиение также необходимо, так как один иероглиф японского языка означает один слог и данные иероглифы необходимо, в первую очередь, разбить на группы, обозначающие отдельные слова.

ГЛАВА 9. ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ

9.1 Демонстрация результатов

Алгоритм, полученный в главе 6 был реализован на языке JavaScript, который используется в разработке клиентской части интернет-ресурсов. Алгоритм был реализован для случая английского языка. Исходный код данной реализации алгоритма доступен в репозитории GitHub, содержащем реализации всех описанных в данной магистерской диссертации подходов.

Также, был создан мини-портал в Глобальной Сети Интернет, на котором возможна демонстрация работы алгоритма. Для ввода исходных данных предусмотрено два поля ввода, для получения результата также добавлено отдельное поле веб-страницы. Созданный ресурс также обладает функцией подстановки в поля ввода предзаполненного текста для облегчения тестирования и ознакомления.

9.2 Наличие в репозиториях

Алгоритмы, описанные и построенные в данной диссертации были также реализованы в виде пакета языка программирования JavaScript. Данный пакет был загружен в npm – репозиторий пакетов с открытым исходным кодом для данного языка.

9.3 Участие в конференциях

Фрагменты из данной работы, рассматриваемые в Главах 2, 3, 6 были представлены на 76 Конференции студентов, магистрантов и аспирантов БГУ. Данный доклад позже был рекомендован к опубликованию в сборнике трудов обозначенной конференции.

ЗАКЛЮЧЕНИЕ

Таким образом, в рамках магистерской диссертации в первую очередь была поставлена задача выделения ключевых слов на стороне клиента и рассмотрены работы по темам, тесно связанным с поставленной. Целью работы являлось исследование и построение эффективных алгоритмов для решения задачи выделения ключевых слов.

В первую очередь в рамках работы были рассмотрены вопросы построения выборок для обучения алгоритмов и замеров качества. Эта задача была рассмотрена вначале с точки зрения имеющихся подходов, затем было предложено несколько решений - от частных случаев к общим. Таким образом, задача сбора выборок была решена. Была также разработана и методология оценивания.

Далее, было предложено несколько альтернативных имеющимся подходов для решения поставленной задачи. Были рассмотрены методы машинного обучения и их применимость для решения задачи выделения ключевых слов. Полученное решение превзошло два рассматриваемых аналога, однако стоит отметить, что при этом, в отличии от них, требовало предобучения.

Далее были рассмотрены вопросы переносимости алгоритма на случаи языков отличных от английского. В первую очередь был рассмотрен вопрос выделения стоп-слов в различных языках, после чего была сделана попытка применить полученный алгоритм к текстам на испанском языке. Предложенный алгоритм также достиг высоких результатов.

Наконец, был реализован простой интернет-ресурс с демонстрацией данного алгоритма, также создана реализация, загруженная в репозиторий пакетов npm. Доклад по материалам диссертации был представлен на конференции студентов, магистрантов и аспирантов БГУ.

Таким образом, поставленные нами в начале диссертации цели были достигнуты.

ПРИЛОЖЕНИЕ 1. РЕПОЗИТОРИИ С РЕАЛИЗАЦИЯМИ ОПИСАННЫХ АЛГОРИТМОВ

Основной репозиторий в системе GitHub доступен по следующей ссылке: https://github.com/zxqfd555/masters-work. Данный репозиторий содержит реализации алгоритмов, описанных в данной диссертации на языках программирования Python и/или C++.

Репозиторий с кодом, содержащий только реализацию созданного пакета для языка программирования JavaScript находится по ссылке: https://github.com/zxqfd555/k-words.

Страница реализованного пакета в npm – репозитории пакетов для языка JavaScript находится по ссылке: https://www.npmjs.com/package/k-words.

ПРИЛОЖЕНИЕ 2. ИНТЕРНЕТ-РЕСУРС ДЕМОНСТРАЦИИ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ

Интернет-ресурс с демонстрацией полученных результатов доступен по следующей ссылке: https://sites.google.com/view/zxqfd555-test-keywords/home.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Automatic Keyword Extraction From Individual Documents / S. Roze, D. Engel, N. Cramer, W. Cowley : Text Mining, Application and Theory, 2010. – 18 с.
2. TextRank: Bringing Order Into Text / R. Mihalcea, P. Tarau, 2004. – 8 с.
3. The PageRank Citation Ranking: Bringing Order to the Web / L. Page, S. Brin, R. Motwani, T. Winograd, 1998. – 17 с.
4. Topic Detection by Clustering Keywords / C. Wartena, R. Brussee – Databases and Expert Systems, 2008. – 10 с.
5. Distributed Representations of Words and Phrases and their Compositionality / T. Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean, 2013. – 9 с.
6. Human-competitive automatic topic indexing / O. Mendelyan : Hamilton, New Zealand, 2009. – 244 с.
7. Keyphrase Extraction Using Deep Recurrent Neural Networks on Twitter / Q. Zhang, Y. Wang, Y. Gong, X. Huang, 2016. – 10 с.
8. Using TF-IDF to Determine Word Relevance in Document Queries / J. Ramos, 2003. – 4 с.
9. SNAP – SNAP: Network Datasets: LiveJournal Social Network [Электрон. ресурс], – Режим доступа: https://snap.stanford.edu/data/soc-LiveJournal1.html. – Дата: 25.06.2018.
10. LiveJournal Data | Data Mining and Search Group [Электрон. ресурс], – Режим доступа: https://dms.sztaki.hu/en/letoltes/livejournal-data. – Дата: 25.06.2018.
11. Variations of the Similarity Function of TextRank for Automated Summarization / F. Barrios, F. Lopez, L. Argerich, R. Wachenchauzer, 2017. – 8 с.
12. aneesha/RAKE: A python implementation of the Rapid Automatic Keyword Extraction [Электрон. ресурс]. – Режим доступа: https://github.com/aneesha/RAKE. – Дата : 19.05.2019.
13. summanlp/textrank: TextRank implementation for Python 3 [Электрон. ресурс]. – Режим доступа: https://github.com/summanlp/textrank. – Дата : 19.05.2019.
14. A Stop List for General Text / C. Fox

15. Efficient Estimation of Word Representations in Vector Space / T. Mikolov, K. Chen, G. Corrado, J. Dean
16. Deep contextualized word representations / M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, L. Zettlemoyer, 2018. – 15 с.
17. ELMo: Deep contextualized word representations [Электрон. ресурс]. – Режим доступа : https://allennlp.org/elmo. – Дата : 01.03.2019
18. nltk – Natural Language Toolkit [Электрон. ресурс]. – Режим доступа: https://www.nltk.org. – Дата : 19.05.2019
19. 6/stopwords-json: Stopwords for 50 languages in JSON format [Электрон. ресурс]. – Режим доступа: https://github.com/6/stopwords-json. – Дата : 19.05.2019
20. An algorithm for suffix stripping / M. F. Porter