МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Кафедра информационных систем управления

CROSS-LANGUAGE ФУНКЦИОНАЛЬНОСТЬ АВТОМАТИЧЕСКОГО ПОИСКА В СЕТИ INTERNET РЕЛЕВАНТНЫХ ДОКУМЕНТОВ

Отчёт по преддипломной практике

Исаченко Дмитрия Александровича

студента 5 курса,
специальность «информатика»

Научный руководитель:
доктор технических наук,
профессор И.В. Совпель

Минск 2017

РЕФЕРАТ

Отчёт по преддипломной практике, 24 стр., 8 рис., 9 источников.

Объектом исследования являются системы, позволяющие для произвольного документа обнаружить релевантные ему документы в сети интернет, в том числе на отличном от исходного языке.

Цель работы: исследовать решения поиска текстовых документов релевантных данному, разработать соответствующий алгоритм и, в соответствии с ним, реализовать приложение.

Результатом работы является приложение под платформу Android, взаимодействующее с облачным хранилищем и позволяющие для произвольного документа обнаружить на заданных пользователем языках релевантные ему документы в сети интернет.

Область применения результатов: классификация и анализ текстов, информационный поиск.

ОГЛАВЛЕНИЕ

ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ И СИМВОЛОВ

ЕЯ - естественный язык;

ПОД - поисковой образ документа;

Стоп-слова - слова, не несущие в себе смысловой и содержательной нагрузки, такие как междометия, предлоги и прочие.

Стемминг - это процесс нахождения основы слова для заданного исходного слова. Основа слова необязательно совпадает с морфологическим корнем слова;

TF (term frequency) – частота термина в контексте определённого документа;

IDF (**inverse document frequency**) - обратная частота термина в корпусе документов;

TF-IDF - статистическая мера, используемая для оценки важности слова в контексте документа, являющегося частью коллекции документов или корпуса;

CLIR - разновидность информационного поиска, при которой язык извлечённой информации может отличаться от языка запроса.

ВВЕДЕНИЕ

В наше время огромное количество информации, в том числе текстовые документы, доступны в электронном виде. Информационные системы, оперирующие большими объемами текстовых документов произвольной предметной области и успешно решающие различные прикладные задачи, становятся все более востребованными как предприятиями и организациями, так и отдельными пользователями. При этом обработка информации, представленной в документах на различных языках, не является тривиальной. В связи с этим актуальна задача автоматизации поиска в сети интернет документов релевантных данному, в том числе на отлихных от иходного языках, что позволяет получить максимальное количество различной информации по теме исходного документа. Эту задачу можно свести к формированию поискового запроса, максимально описывающего тему данного документа, что в свою очередь сводится к задаче определения ключевых слов в тексте. Существуют следующие категории методов выделения ключевых слов: статистические, лингвистические и гибридные, которые являются их комбинацией. Популярные статистические методы для измерения важности слов используют статистическую меры TF-IDF, которая учитывает, как часто данное слово встречается в документе и в то же время как редко – в корпусе документов. Такой способ нахождения ключевых слов обладает более высокой точностью по сравнению с другими, в которых не задействуется предварительное обучение системы на корпусе документов.

В данной работе описывается подход по формированию поискового запроса для автоматизации поиска релевантных данному документов, в том числе на отлихных от иходного языках. В связи с тем, что большинство людей сейчас проводят больше времени в своих мобильных телефонах/планшетах, чем в десктопах, приложение будет разрабатываться под платформу Android на языке программирования Java.

ГЛАВА 1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И ПОСТАНОВКА ЗАДАЧИ

1.1 Исследование подходов к решению задачи в одноязычной информационной среде

Задача автоматизации поиска в сети интернет документов релевантных данному относится к классическим задачам информационного поиска и её можно решать одним из двух следующих способов:

- реализовать свою поисковую систему при разработке приложения;
- воспользоваться уже существующей поисковой системой при разработке приложения.

Поисковая система в данном случае устроенна следующим образом.

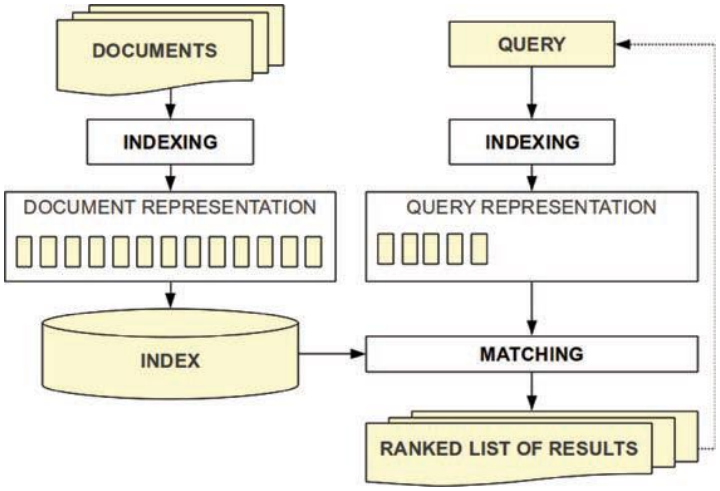

рисунок 1.1 – Одноязычный поиск информации

Основные составляющие поисковой системы: поисковый робот, индексатор, поисковик.

Поисковый робот — программа, являющаяся составной частью поисковой системы и предназначенная для перебора страниц Интернета с целью занесения информации о них в базу данных поисковика. По принципу действия паук напоминает обычный браузер. Он анализирует содержимое страницы, сохраняет его в некотором специальном виде на сервере поисковой машины, которой принадлежит, и отправляется по ссылкам на следующие страницы. Владельцы поисковых машин нередко ограничивают глубину

проникновения паука внутрь сайта и максимальный размер сканируемого текста, поэтому чересчур большие сайты могут оказаться не полностью проиндексированными поисковой машиной. Порядок обхода страниц, частота визитов, защита от зацикливания, а также критерии выделения значимой информации определяются алгоритмами информационного поиска. В большинстве случаев переход от одной страницы к другой осуществляется по ссылкам, содержащимся на первой и последующих страницах. Также многие поисковые системы предоставляют пользователю возможность самостоятельно добавить сайт в очередь для индексирования. Обычно это существенно ускоряет индексирование сайта, а в случаях, когда никакие внешние ссылки не ведут на сайт, вообще оказывается практически единственной возможностью указать на его существование.

Индексатор — это модуль, который анализирует страницу, предварительно разбив её на части, применяя собственные лексические и морфологические алгоритмы. Все элементы веб-страницы вычленяются и анализируются отдельно. Данные о веб-страницах хранятся в индексной базе данных для использования в последующих запросах. Индекс позволяет быстро находить информацию по запросу пользователя.

Индексирование в поисковых системах — процесс добавления сведений роботом поисковой машины в базу данных, впоследствии использующуюся для поиска информации на проиндексированных сайтах. В сведения о сайте чаще всего входят ключевые слова (алгоритм определения ключевых слов зависит от поисковой системы), статьи, ссылки, документы, также могут индексироваться изображения, аудио и т. д. Скорость индексации новых сайтов в поисковой системе Яндекс занимает от одной недели до четырех недель, в Google — от нескольких минут до одной недели.

Поисковый запрос — это какая-то последовательность символов, которую пользователь вводит в поисковую строку, чтобы найти интересующую его информацию. Формат поискового запроса зависит как от устройства поисковой системы, так и от типа информации для поиска. Чаще всего поисковый запрос задаётся в виде набора слов или фразы, иногда — используя расширенные возможности языка запросов поисковой системы.

Как правило, системы работают поэтапно. Сначала поисковый робот получает контент, затем индексатор генерирует доступный для поиска индекс, и наконец, поисковик обеспечивает функциональность для поиска индексируемых данных. Чтобы обновить поисковую систему, этот цикл индексации выполняется повторно.

Поисковые системы работают, храня информацию о многих веб-страницах, которые они получают из HTML страниц. Поисковая система анализирует содержание каждой страницы для дальнейшего индексирования.

Слова могут быть извлечены из заголовков, текста страницы или специальных полей — метатегов. Ряд поисковых систем, подобных Google, хранят исходную страницу целиком или её часть, так называемый кэш, а также различную информацию о веб-странице. Другие системы, подобные системе AltaVista, хранят каждое слово каждой найденной страницы. Использование кэша помогает ускорить извлечение информации с уже посещённых страниц. Кэшированные страницы всегда содержат тот текст, который пользователь задал в поисковом запросе. Это может быть полезно в том случае, когда веб-страница обновилась, то есть уже не содержит текст запроса пользователя, а страница в кэше ещё старая. Поисковик работает с выходными файлами, полученными от индексатора. Поисковик принимает пользовательские запросы, обрабатывает их при помощи индекса и возвращает результаты поиска. Когда пользователь вводит запрос в поисковую систему (обычно при помощи ключевых слов), система проверяет свой индекс и выдаёт список наиболее подходящих веб-страниц обычно с краткой аннотацией, содержащей заголовок документа и иногда части текста. Поисковый индекс строится по специальной методике на основе информации, извлечённой из веб-страниц. Полезность поисковой системы зависит от релевантности найденных ею страниц. Хоть миллионы веб-страниц и могут включать некое слово или фразу, но одни из них могут быть более релевантны, популярны или авторитетны, чем другие. Большинство поисковых систем использует методы ранжирования, чтобы вывести в начало списка «лучшие» результаты. Поисковые системы решают, какие страницы более релевантны, и в каком порядке должны быть показаны результаты, по-разному. Методы поиска, как и сам Интернет со временем меняются. Так появились два основных типа поисковых систем: системы предопределённых и иерархически упорядоченных ключевых слов и системы, в которых генерируется инвертированный индекс на основе анализа текста.

В связи с огромной трудоёмкостью разработки собственной поисковой системы при разработке приложения будет использоваться поисковая система Google. Таким образом задача автоматизации поиска в сети интернет документов релевантных данному сведётся к формированию поискового запроса. Поисковой запрос формируется из ключевых для исходного текста слов. Количество слов в запросе можем варьироваться в зависимости от размера документа. Согласно рекомендациям Google, поисковой запрос должен состоять из ключевых слов, оптимальное количество ключевых слов должно находиться в диапазоне 5-8.

Существуют следующие категории методов выделения ключевых слов: статистические, лингвистические и гибридные, которые являются их комбинацией.

Лингвистические методы основываются на значениях слов, используют онтологии (попытки формализации некоторой области знаний с помощью концептуальной схемы) и семантические данные о слове. Эти методы слишком трудоемки на ранних этапах: разработка онтологий является очень затратным процессом. К тому же, операции лингвистического анализа текстов, выполняемые вручную, являются источником значительного количества ошибок и неточностей и делают сам процесс анализа документов сложным и длительным. Поэтому необходимым условием является наличие доступных программных средств, позволяющих автоматизировать процесс анализа текстов документов.

Лингвистические методы, основанные на графах, представляют большой интерес в области обработки естественного языка. В этих методах основной процедурой является построение семантического графа. Это взвешенный граф, вершинами которого являются термины документа, наличие ребра между двумя вершинами свидетельствует о том, что термины семантически связаны между собой, вес ребра является численным значением семантической близости двух терминов. Ключевые слова отбираются алгоритмами обработки графа. Графовые методы различаются между собой способами отбора множества терминов и определения близости отдельных терминов, которые основаны на статистических параметрах, а также на морфологическом, синтаксическом или семантическом анализе.

Статистические методы основываются на численных данных о встречаемости слова в тексте. Их преимуществами являются универсальность алгоритмов извлечения ключевых слов, отсутствие необходимости в трудоемких процедурах построения лингвистических баз знаний, простота реализации. Максимальную точность и полноту имеют алгоритмы, которые базируются на статистических исследованиях корпусов документов. Алгоритмы, которые предварительно не обрабатывают никаких документов, кроме того, ключевые слова которого необходимо извлечь, обладают сравнительно более низкой точностью. Классическими подходами в области статистической обработки естественного языка можно считать использование метрики TF-IDF и ее модификаций (для выделения ключевых слов), и анализ коллокаций (для выделения словосочетаний).

Проанализировав вышеописанные методы извлечения ключевых слов/словосочетаний, было замечено, что большинство из их состоят из следующих 3-х этапов:

1. Отбор кандидатов: выделяются все возможные слова, фразы, термины или понятия (в зависимости от поставленной задачи), которые потенциально могут быть ключевыми;

2. Анализ свойств: для каждого кандидата нужно вычислить свойства, которые указывают, что он может быть ключевым. Например, кандидат, появляющийся в названии книги, скорей всего является ключевым;
3. Вычисление весов важности и соответствующий выбор ключевых слов/словосочетаний: для всех кандидатов с использованием различных статистических методов (так же может быть задействовано машинное обучение) вычисляются соответствующие им веса важности в документе. Полученные веса затем используются для окончательного определения ключевых слов/словосочетаний.

В связи с трудоёмкостью реализации собственного лингвистического процессора в данной работе для выделения ключевых слов при формировании поискового запроса будет рассмотрен статистический метод, использующий статистическую меру TF-IDF.

1.2 Решение задачи в многоязычной информационной среде

При разработке собственной поисковой системы, поддерживающей обнаружение информации на языке отличной от языка запроса, можно было бы перевести все имеющиеся документы на все возможные языки запросов.

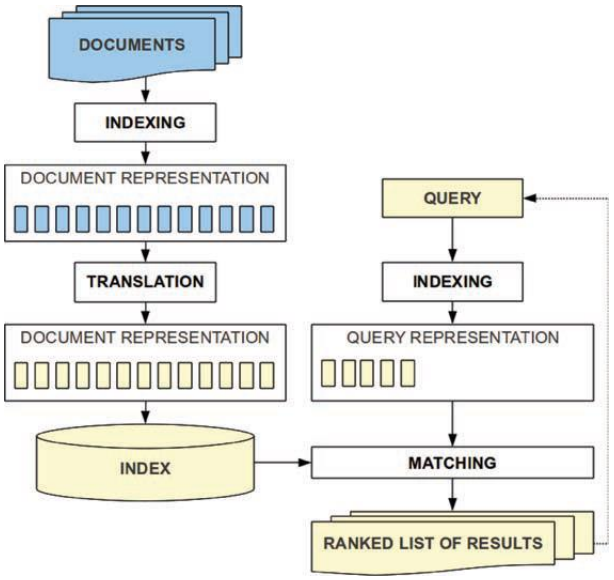

рисунок 1.2 – CLIR с переводом документов

Так же можно было бы ввести промежуточный язык на который бы переводились все документы и поисковой запрос.

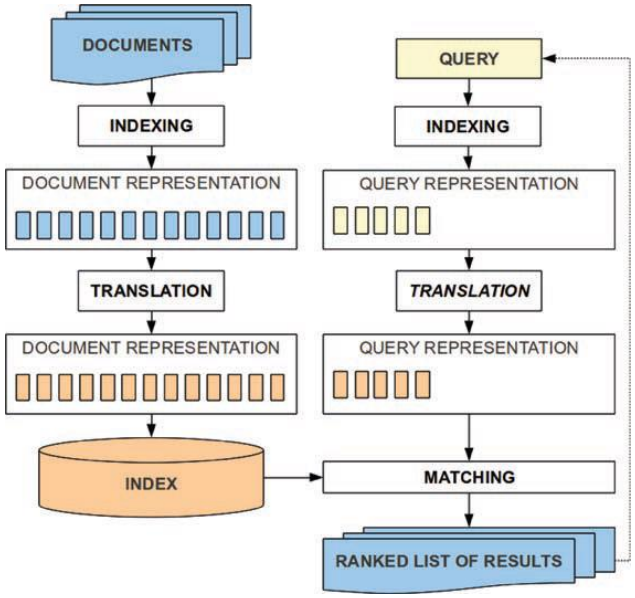

рисунок 1.3 – CLIR с переводом документов и запроса на промежуточный язык

У каждого из данных подходов имеются свои плюсы и минусы, но так как мы решили не разрабатывать собственную поисковую систмему, а воспользоваться уже существующей, то рассмотрим третий подход, основывающийся на переводе запроса.

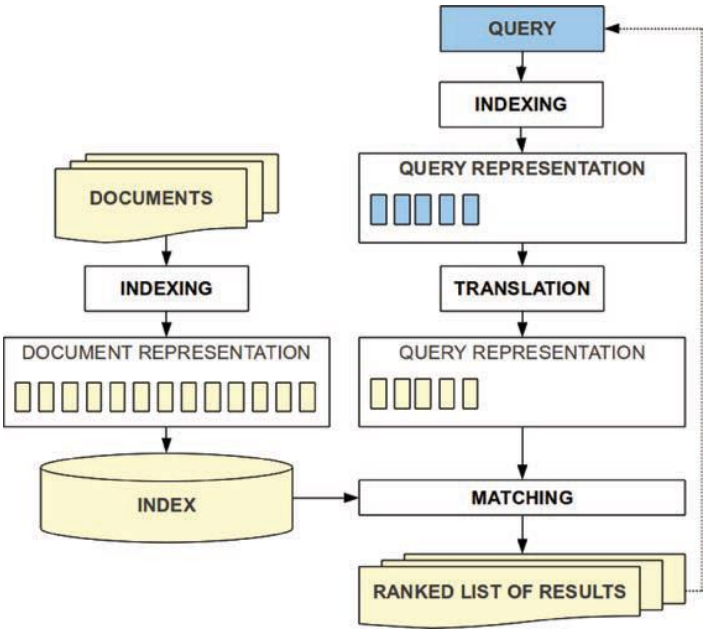

рисунок 1.4 – CLIR с переводом запроса

1.3 Постановка задачи

Требуется разработать мобильное приложение под платформу Android, взаимодействующее с облачным хранилищем и позволяющие для произвольного документа обнаружить на заданных пользователем языках релевантные ему документы в сети интернет.

1.4 Выводы

В первой главе получены следующие результаты:

- выполнен исследование предметной области;
- поставлена задача разработки системы.

ГЛАВА 2. АЛГОРИТМИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ЗАДАЧИ

2.1 Описание алгоритма

Алгоритм:

1. Определить язык исходного текста;
2. Перевести исходный текст на заданные пользователем языки. Если система не обучалась для какого-то из заданных языков, то вместо данного языка перевести текст на английский.
3. Провести токенезацию переведённых текстов(возможно стемминг, в зависимости от языка);
4. Определить для токенов важность их в контексте соответствующего документа;
5. Исходя из полученных весов важности определить наборы ключевых слов;
6. Из составленных наборов ключевых слов сгенерировать поисковые запросы для Google;
7. Используя поисковые запросы, найти релевантные данному документы в сети интернет.

2.2 Стемминг

При реализации данного алгоритма для английского языка будет использоваться стемминг Портера в силу его быстродействия, отсутствия необходимости в предварительной обработке корпуса документов и использования каких-либо баз основ.

Основная идея стеммера Портера заключается в том, что существует ограниченное количество словообразующих суффиксов и вручную заданы некоторые правила. Алгоритм состоит из пяти шагов. На каждом шаге отсекается словообразующий суффикс и оставшаяся часть проверяется на соответствие правилам. Если полученное слово удовлетворяет правилам, происходит переход на следующий шаг. Если нет — алгоритм выбирает другой суффикс для отсечения. Минусами данного алгоритма является то, что он часто обрезает слово больше необходимого, а это затрудняет получение правильной основы слова. Также стеммер Портера не справляется со всевозможными изменениями корня слова (например, выпадающие и беглые гласные).

В зависимости от необходимой производительности и точности можно выбрать алгоритм стемминга из приведенных ниже типов.

2.2.1 Алгоритмы поиска

Простой стеммер ищет флективную форму в таблице поиска. Преимущества этого подхода заключается в его простоте, скорости, а также легкости обработки исключений. К недостаткам можно отнести то, что все флективные формы должны быть явно перечислены в таблице: новые или незнакомые слова не будут обрабатываться, даже если они являются правильными (например, iPhones ~ iPhone), а также проблемой является то, что таблица поиска может быть очень большой.Для языков с простой морфологией наподобие английского размеры таблиц небольшие, но для сильно флективных языков (например, турецкий) таблица может иметь сотни возможных флективных форм для каждого корня.

Таблицы поиска, используемые в стеммерах, как правило, генерируются в полуавтоматическом режиме. Например, для английского слова «run» автоматически будут сгенерированы формы «running», «runs», «runned» и «runly». Последние две формы являются допустимыми конструкциями, но они вряд ли появятся в обычном тексте на английском языке.

Алгоритм поиска может использовать предварительную частеречную разметку, чтобы избежать такой разновидности ошибки лемматизации, когда разные слова относят к одной лемме

2.2.2 Алгоритмы усечения окончаний

Алгоритмы усечения окончаний не используют справочную таблицу, которая состоит из флективных форм и отношений корня и формы. Вместо этого, как правило, хранится меньший список «правил», который используется алгоритмами, учитывая форму слова, чтобы найти его основу[3]. Некоторые примеры правил выглядят следующим образом:

- если слово оканчивается на 'ed', удалить 'ed'
- если слово оканчивается на 'ing', удалить 'ing'
- если слово оканчивается на 'ly', удалить 'ly'

Алгоритмы усечения окончаний неэффективны для исключительных ситуаций (например, 'ran' и 'run')). Решения, полученные алгоритмами усечения окончаний, ограничиваются теми частями речи, которые имеют хорошо известные окончания и суффиксы с некоторыми исключениями. Это

является серьезным ограничением, так как не все части речи имеют хорошо сформулированный набор правил. Лемматизация пытается снять это ограничение.

2.2.3 Алгоритмы лемматизации

Более сложным подходом к решению проблемы определения основы слова является лемматизация. Чтобы понять, как работает лемматизация, нужно знать, как создаются различные формы слова. Большинство слов изменяется, когда они используются в различных грамматических формах. Конец слова заменяется на грамматическое окончание, и это приводит к новой форме исходного слова. Лемматизация выполняет обратное преобразование: заменяет грамматическое окончание суффиксом или окончанием начальной формы.

Также лемматизация включает определение части речи слова и применение различных правил нормализации для каждой части речи. Определение части речи происходит до попытки найти основу, поскольку для некоторых языков правила стемминга зависят от части речи данного слова.

2.2.4 Алгоритмы стемминга на основе корпуса текстов

Одим из главных недостатков классических стеммеров (например, стеммер Портера) является то, что они часто не различают слова со схожим синтаксисом, но совершенно с разными значениями. Например, «news» и «new» в результате стемминга сведутся к основе «new», хотя данные слова принадлежат к разным лексическим категориям. Другая проблема заключается в том, что некоторые алгоритмы стемминга могут быть пригодны для одного корпуса и вызывать слишком много ошибок в другом. Например, слова «stock», «stocks», «stocking» и т. д. будут иметь особое значение в текстах газеты The Wall Street Journal. Основная идея стемминга на основе корпуса текстов состоит в создании классов эквивалентности для слов классических стеммеров, а затем «разбить» некоторые слова, объединенные на основе их встречаемости в корпусе.

2.2.5 Алгоритмы сопоставления

Такие алгоритмы используют базу данных основ (например, набор

документов, содержащие основы слов). Данные основы не обязательно соответствуют обычным словам, в большинстве случаев основа представляет собой подстроку (например, для английского языка «brows» является подстрокой в словах «browse» и «browsing»). Для того, что определить основу слова алгоритм пытается сопоставить его с основами из базы данных, применяя различные ограничения, например, на длину искомой основы в слове относительно длины самого слова (так, например, короткий префикс «be», который является основой таких слов, как «be», «been» и «being», не будет являться основой слова «beside»).

2.3 Определение важности слова в контексте документа

Для оценки важности слова в контексте документа будем использовать статистическую меру TF-IDF, которая является произведением двух статистик: частоты термина в данном документе и обратной частоты термина в корпусе документов. Существуют различные способы определения данных статистик.

Введём следующие обозначения:

- D - корпус документов;
- N - размер корпуса документов;
- t - термин, важность которого хотим определить в документе d.

Тогда:

n(t) = 1 + количество документов, в которых встречается термин t;

f(t,d) = количество раз, которое термин t встречается в документе d.

Способы определения статистики TF:

- по частоте встречаемости (raw frequency): $\mathrm{tf}(t, d) = f(t, d)$;
- логический (boolean frequency): $\mathrm{tf}(t, d) = \begin{cases} 1, f(t, d) > 0 \\ 0, f(t, d) = 0 \end{cases}$;
- логарифмически нормализованный (logarithmically scaled frequency): $\mathrm{tf}(t, d) = \begin{cases} 1 + \log(f(t, d)), f(t, d) > 0 \\ 0, f(t, d) = 0 \end{cases}$;
- нормализованный по максимальной частоте слова (augmented frequency): $\mathrm{tf}(t, d) = 0.5 + \frac{0.5 * f(t, d)}{\max\{f(t', d) : t' \in d\}}$.

Способы определения статистики IDF:

- $idf(t, D) = \log(\frac{N}{n(t)})$;
- $idf(t, D) = \log(1 + \frac{N}{n(t)})$;
- $idf(t, D) = \log(1 + \frac{\max\{n(t'), t' \in d\}}{n(t)})$;
- $idf(t, D) = \log(\frac{N - n(t)}{n(t)})$.

Различные варианты схемы взвешивания TF-IDF часто используются поисковыми системами в качестве основного инструмента при ранжировании по релевантности документов для данного поискового запроса. Так же TF-IDF может быть успешно использован при фильтрации стоп-слов в различных предметных областях.

В качестве статистической меры TF-IDF в данной работе выбрана:

$$tfidf(t, d, D) = (0.5 + \frac{0.5 * f(t, d)}{\max\{f(t', d) : t' \in d\}}) * \log(1 + \frac{N}{n(t)})$$

2.4 Выводы

Во второй главе получены следующие результаты:

– выполнен анализ существующих алгоритмов и их улучшений для извлечения ключевых слов;

– разработан алгоритм для автоматического поиска в сети интернет документов релевантных данному в том числе на отличных от исходного языках.

ГЛАВА 3. РЕАЛИЗАЦИЯ СИСТЕМЫ

3.1 Разработка архитектуры системы

В качестве облачного хранилища была выбрана Firebase Realtime Database в связи с возможностью её бесплатного использования.

При разработке приложения был использован объектно-ориентированный подход. Приложение разбито на модули согласно функциональности. Имеются следующие агенты:

- Агент взаимодействия с облачным хранилищем;
- Агент пользовательского интерфейса, который преобразует команды пользователя и передаёт их соответствующим агентам;
- Агент, инкапсулирующий логику формирования поискового запроса для исходного документа;
- Агент, отвечающий за особенности работы с ОС Android.

3.2 Особенности реализации мобильного приложения

Так как реализованное мобильное приложение на текущий момент не взаимодействует с сервером и все полученные в результате вычислений данные хранятся на самом устройстве, то при обучении брался корпус, состоящий из 20 документов. Чтобы повысить точность и полноту результатов поисковых запросов перед проведением стемминга был добавлен этап фильтрации стоп-слов.

Список стоп слов для английского языка:

a, about, above, after, again, against, all, am, an, and, any, are, aren't, as, at, be, because, been, before, being, below ,between, both, but, by, can't, cannot, could, couldn't, did, didn't, do, does, doesn't, doing, don't, down, during, each, few, for, from, further, had, hadn't, has, hasn't, have, haven't, having, he, he'd, he'll, he's, her, here, here's, hers, herself, him, himself, his, how, how's, i, i'd, i'll, i'm, i've, if, in, into, is, isn't, it, it's, its, itself, let's, me, more, most, mustn't, my, myself, no, nor, not, of, off, on, once, only, or, other, ought, our, ours, ourselves, out, over, own, same, shan't, she, she'd, she'll, she's, should, shouldn't, so, some, such, than, that, that's, the, their, theirs, them, themselves, then, there, there's, , these, they, hey'd, they'll, they're, they've, this, those, through, to, too, under, until, up, very, was, wasn't, we, we'd, we'll, we're, we've, were, weren't, what, what's, when, when's, where, where's,

which, while, who, who's, whom, why, why's, with, won't, would, wouldn't, you, you'd, you'll, you're, you've, your, yours, yourself, yourselves.

3.3 Методика применения разработанного приложения

Разработанная программа обладает интуитивно понятным интерфейсом.

Алгоритм использования приложения для поиска документов в сети интернет релевантных данному:

1. Запустить приложение;
2. Ввести текст документа/выбрать документ из локального хранилища и нажать “Продолжить”;
3. Выбрать актуальные для поиска информации языки и нажать “Продолжить”.

Так же можно просмотреть корпус документов, который использовался при обучении системы, и полученную в результате обучения системы статистику важности различных слов в различных языках.

3.4 Пример использования разработанной системы

Запуск программы и ввод текста для которого необходимо найти в сети интернет релевантные документы.

рисунок 3.1 – Главный экран приложения

По нажатию кнопки “Продолжить” появится экран выбора актуальных для пользователя языков. На данном экране отображены все поддерживаемые для поиска информации приложением языки.

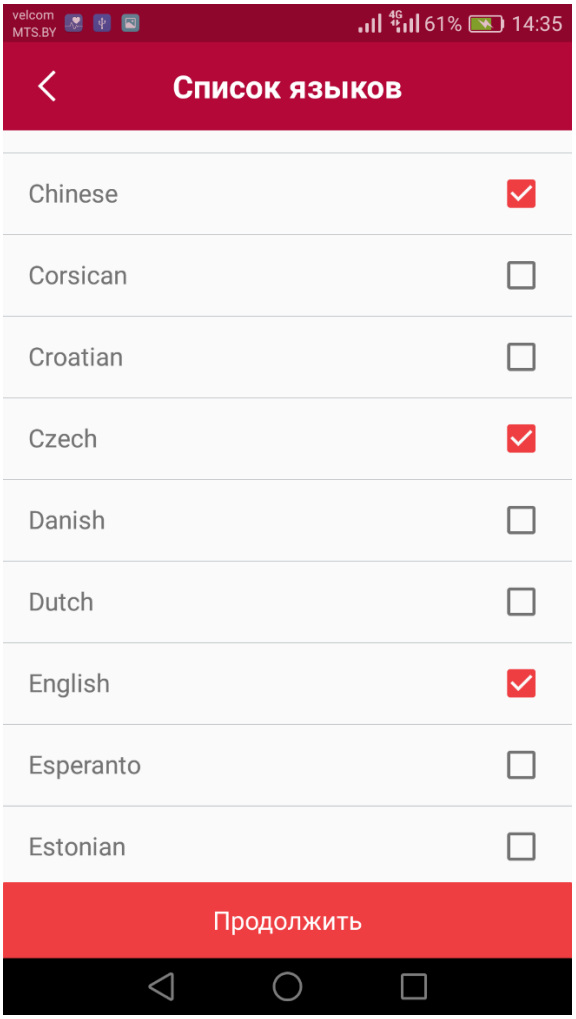

рисунок 3.2 – Экран выбор актуальных пользователю языков

По нажатию кнопки “Продолжить” будет выдан список сформированных приложением запросов для поиска релевантных данному документов на выбранных языках. В дальнейшем список языков можно будет поменять не вводя текст заново.

Экран результата запросов для поиска релевантных данному документов на выбранных языках. А так же экран результатов поиска для определённого запроса в поисковой системе Google приведены ниже.

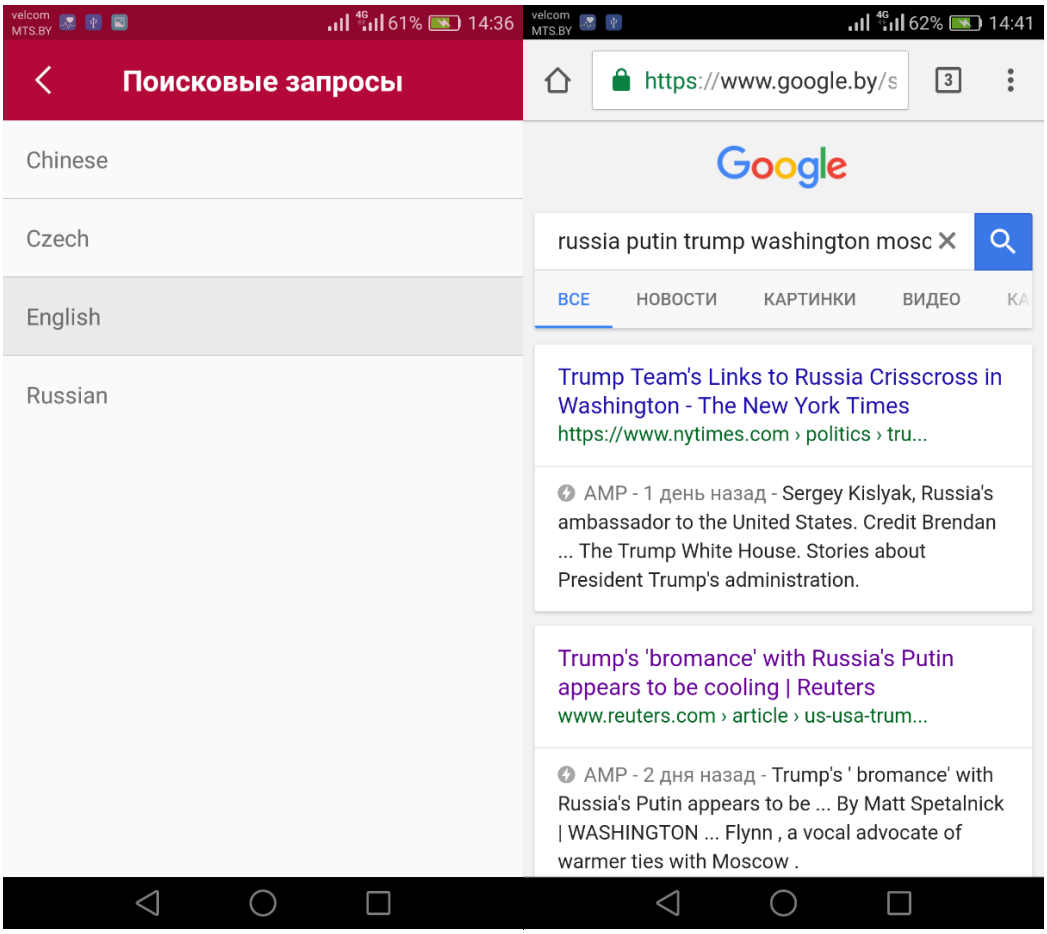

рисунок 3.3 – Результаты поиска релевантных документов

Так же через навигационное меню можно просмотреть корпус документов, который использовался при обучении системы, и полученную в результате обучения системы статистику. Для этого в навигационном меню нужно выбрать пункт “Корпус” / “Словарь” соответственно.

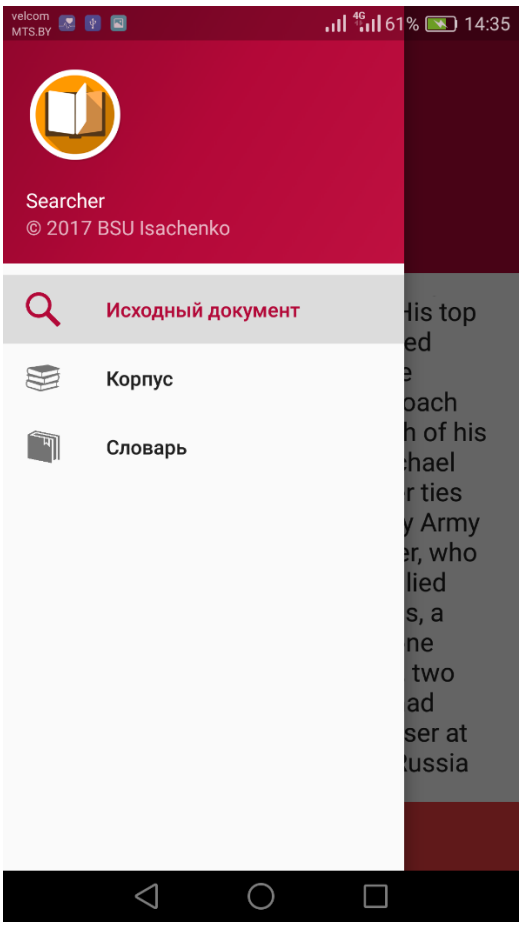

рисунок 3.4 – Навигационное меню приложения

По клику на определённый документ на экране “Корпус” загрузится отдельный экран, позволяющий его прочесть и изучить детальную статистику по данному документу.

3.5 Выводы

В третьей главе получены следующие результаты:

- разработана архитектура и описаны особенности реализации мобильного приложения;
- описана методика применения разработанного приложения.

ЗАКЛЮЧЕНИЕ

В ходе проделанной работы:

1) Выполнено исследование предметной области;
2) Поставлена задача разработки системы;
3) Разработан алгоритм для автоматического поиска в сети интернет документов релевантных данному в том числе на отличных от исходного языках;
4) Разработана архитектура и описаны особенности реализации мобильного приложения;
5) Описана методика применения разработанного приложения.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Manning, C. D. Introduction to Information Retrieval. / C. D. Manning, P. Raghavan, H. Schütze. - Cambridge University Press, 2008. – 581 с.
2. Автоматическая обработка текстов на естественном языке и компьютерная лингвистика: учеб. пособие /Е.И. Большакова, Э.С. Клышински, Д.В. Ландэ [и др.]. - М.: МИЭМ, 2011. - 272 с.
3. Matsuo, Y. Keyword Extraction from a Single Document using Word Co-occurrence Statistical Information. / Y. Matsuo – Tokyo, 2003. – 13 с.
4. Turney, P.D. Learning algorithms for keyphrase extraction. Information Retrieval / P.D. Turney. - Ottawa, Ontario, Canada, 2000. – 477 с.
5. Porter, M.F. An algorithm for suffix stripping. / M.F. Porter – Cambridge, 1997. – 6 с.
6. ANDERKA, M., LIPKA, N., AND STEIN, B. 2009. Evaluating cross-language explicit semantic analysis and cross querying. In Proceedings of the 10th Cross-language Evaluation Forum Conference on Multilingual Information Access Evaluation: Text Retrieval Experiments (CLEF’09). Springer, 50–57 с.
7. Kraiij, W., Nie, J-Y., Simard, M.: Emebdding Web-based Statistical Translation Models in Cross-Language Information Retrieval. Computational Linguistics (2003) – 39 с.
8. The Cross-Language Evaluation Forum (CLEF). http://clef-campaign.org
9. Virga, P., Khudanpur, S.: Transliteration of proper names in cross-lingual information retrieval. In: ACL Workshop on Multilingual and Mixed Language Named Entity Recogni-tion (2003) – 8 с.