Санкт-Петербургский государственный университет

Факультет прикладной математики – процессов управления

**Кафедра технологии программирования**

**Семенов Валерий Геннадьевич**

**Выпускная квалификационная работа бакалавра**

**Поисковые запросы и их связи с веб-сайтами**

Направление 010400

Прикладная математика и информатика

Научный руководитель,

кандидат физ.-мат. наук,

доцент

Должиков В.В.

Санкт-Петербург

2017

Оглавление

[Введение 4](#_Toc475040705)

[Постановка задачи 5](#_Toc475040706)

[Обзор литературы 6](#_Toc475040707)

[История поисковых систем 7](#_Toc475040708)

[Характеристики поисковых запросов 8](#_Toc475040709)

[Стоп-слова 9](#_Toc475040710)

[Ограничение расстояния между словами 10](#_Toc475040711)

[Операторы 10](#_Toc475040712)

[Понятия полноты и точности 13](#_Toc475040713)

[Структура страницы поисковой выдачи поисковых систем 14](#_Toc475040714)

[Расширенный поиск 17](#_Toc475040715)

[Основные положения принципов работы 18](#_Toc475040716)

[Модуль индексирования 18](#_Toc475040717)

[Базы данных поисковых систем 19](#_Toc475040718)

[Системы выдачи результатов 19](#_Toc475040719)

[Принципы и методы работы краулера BeeBot 20](#_Toc475040720)

[Математическая формулировка задачи о многоруком бандите и алгоритмы ее решения 22](#_Toc475040721)

[Алгоритм Гиттенса 23](#_Toc475040722)

[Алгоритм UCB1 24](#_Toc475040723)

[Базы данных поисковой системы 26](#_Toc475040724)

[Обработка результатов запроса пользователя 28](#_Toc475040725)

[Поиск с использованием частоты термина 29](#_Toc475040726)

[Векторный поиск 30](#_Toc475040727)

[Поиск с использованием релевантности страниц 33](#_Toc475040728)

[Параллельный поиск 35](#_Toc475040729)

[Классификация текстовых документов с использованием метода «Наивный Байес» 36](#_Toc475040730)

[«Наивный Байес» при поиске новостей 38](#_Toc475040731)

[Выводы 41](#_Toc475040732)

[Заключение 42](#_Toc475040733)

[Список литературы 43](#_Toc475040734)

# Введение

В современном мире такой ресурс как информация становится важнейшим ресурсом. Владение данным ресурсом в наиболее полной степени помогает человечеству эффективнее и быстрее решать различные задачи. Различные поисковые системы позволяют находить различные необходимые интернет-пользователям интернет-ресурсы в максимально короткое время. Системы и алгоритмы ранжирования интернет-ресурсов, используемые в поисковых системах дают возможность пользователям из всего многообразия документов, находящихся в сети Интернет находить наиболее подходящие, т.е. те, которые наиболее точно и полно отражают результаты, которые требуются пользователю сети Интернет. Также различные сервисы поисковых систем осуществляют классификацию документов, находящихся в сети Интернет в соответствии с их типами, соответственно данные сервисы также дают пользователям возможность осуществлять более эффективный поиск, например, при поиске пользователем некоторых фильмов, пользователю стоит воспользоваться видео сервисом, при поиске некоторых картинок, естественно, сервисом, содержащим графические изображения. Функции расширенного поиска поисковых систем дают возможность осуществлять поиск документов, находящихся в определенных форматах, на определенных языках и появившихся в сети в определенный диапазон времени.

Поисковая система представляет собой совокупность программных средств, реализующих различные алгоритмы, направленные на осуществление наиболее эффективного поиска, а также аппаратных средств, осуществляющих функционирование программных средств. Реализация и алгоритмы, которые реализуют программные средства поисковой системы являются коммерческой тайной, однако общие и некоторые распространенные идеи устройства поисковых систем и алгоритмы работы отдельных компонентов поисковых систем известны широкому кругу пользователей. Алгоритмы работы отдельных компонентов поисковых систем, реализации этих алгоритмов и методы взаимодействия этих компонент между собой представляют в совокупности поисковую систему.

# Постановка задачи

* Изучение основных компонент поисковой системы
* Изучение некоторых алгоритмов работы компонентов поисковой системы. К компонентам поисковой системы можно отнести, например, алгоритмы работы так называемого краулера – программы, осуществляющей загрузку ссылок на определенной интернет-странице, а также осуществляющей выбор интернет-страницы для сканирования на каждом последующем шаге сканирования.
* Изучение некоторых способов индексирования документов т.е. способов записи документов в базу данных поисковой системы, а соответственно способов хранения информации в базе данных поисковой системы.
* Изучение некоторых способов формирования выдачи результатов найденных документов по запросу пользователя, включающих в себя алгоритмы ранжирования страниц.
* Изучение методов перенаправления запросов на обработку определенным сервисам на примере некоторой системы.
* Разработка и реализация нового более эффективного алгоритма поиска документов, относящихся к определенной тематике.

# Обзор литературы

Для получения общего представления о структуре поисковых систем, их предназначения и взаимодействия между собой оказалась полезной ссылка на интернет-страницу[9], а также ссылка [2]. Подробное рассмотрение работы такого компонента поисковой системы как Crawler (краулер) было представлено в научной статье[10] на примере краулера BeeBot, также в данной статье были представлены алгоритмы работы данного краулера, которые, в зависимости от тематики сайта, подходили в большей или меньшей степени для разных сайтов. Основные идеи подходов в ранжировании результатов поисковыми системами были представлены по интернет-ссылке[13], а также были дополнены статьей[14]. Презентации Юрия Лившица [15], [17], а также интернет-страница [16] помогли дополнить знания об идеях ранжирования интернет-страниц поисковой системой «Google» с помощью введения такого понятия как PageRank.На интернет-странице[18] принципы работы оптимизации результатов поисковых запросов с помощью параллельного поиска в поисковой системе «Яндекс». В презентации [19], а также по интернет-странице [20] описаны идеи и формулы классификации документов «Наивный Байес», которые были модифицированы мною в данной работе для написания небольшого поискового сервера, осуществляющего поиск интересующих пользователя новостей.

# История поисковых систем

В данном разделе были использованы материалы со страницы:

[1] Введение в поисковые системы

<http://works.doklad.ru/view/4yqZXQ0Pous/all.html>

а также материалы со страницы:

[2] Поисковые системы: состав, функции, принципы работы:

<http://www.studfiles.ru/preview/5866029/page:20/>

Когда интернет только начинал развиваться, число пользователей интернета было сравнительно небольшим, вследствие чего не было острой необходимости в быстром поиске требуемой информации. Доступ к сети имели практически только сотрудники научно-исследовательских сфер. Однако, в 1990 году британский ученый, создатель всемирной паутины World Wide Web, протокола передачи данных http, а также языка гипертекстовой разметки html - Сэр Тимоти Джон Бернерс-Ли создал первый в мире каталог интернет-сайтов **info.cern.ch.** Данный каталог имел список интернет-ресурсов, отсортированных по тематике и был первым шагом помощи интернет-пользователю в поиске необходимой информации в интернете.

Первым крупным проектом стала поисковая система Yahoo, появившаяся в 1994 году. Из-за того, что количество сайтов на нем росло с геометрической прогрессией, вскоре Yahoo добавил возможность поиска по каталогам. В некотором смысле, Yahoo стала поисковой системой, однако осуществление поиска производилось только по интренет-ресурсам, занесенным в каталог Yahoo. В середине 1990-х интернет каталоги стали набирать популярность, однако позже они потеряли популярность вследствие появления поисковых систем. В настоящее время поиск с использованием каталогов крайне неэффективен из-за невероятно большого количества интернет-ресурсов в интернете. Даже крупнейший на настоящий момент каталог DMOZ, включающий в себя информацию о нескольких миллионах интернет-ресурсах, содержит в тысячи раз меньше интернет-ресурсов, чем крупнейшие поисковые системы. В настоящее время занесение в каталог даже одного процента интернет-ресурсов от всего количества интернет ресурсов в интернете практически невозможно ввиду трудоемкости данной работы.

В 1997 году два студента Стэндфордского университета в качестве исследовательского проекта создали поисковую систему «Google», ставшей впоследствии одной из крупнейших поисковых систем в мире.

В том же 1997 году поисковая система Yandex-web была представлена на выставке Softtool (ежегодная выставка информационных технологий в России), впоследствии ставшая самой популярной поисковой системой среди русскоязычного населения.

# Характеристики поисковых запросов

В данном разделе, включающем в себя также такие параграфы как:

«стоп-слова», «ограничение расстояния между словами», «операторы», «понятия полноты и точности», были использованы материалы со страницы:

[3] Поисковые системы Интернет: основные принципы организации и формирование поисковых запросов

<http://e-ikt.uginfo.sfedu.ru/lectures/lecture4/lecture4_6.html>

Поисковый запрос – некоторый набор слов или некоторая фраза, по которой осуществляется поиск некоторой информации пользователем.

Ключевое слово или фраза, по которому осуществляется поиск, вводятся в поисковой строке некоторой поисковой системы. Далее ключевое слово, набор слов или фраза обрабатываются поисковыми алгоритмами, и поисковые системы предоставляют пользователю результат. Также поисковый запрос может включать в себя знаки препинания. При составлении элементарных простейших поисковых запросов нет необходимости использования углубленных знаний языка запросов. При запросе поисковая система выдает все веб-сайты с указанием страниц или того контекста документа, в котором были найдены необходимые слова. При этом результаты выводятся в определенном порядке, зависящем от популярности интернет-ресурса, от конкретных предпочтений пользователя или иных факторов. Однако при расширенном поисковом запросе имеется острая необходимость в знании и умении правильного применения языка запросов, если у поисковой системы нет возможности их автоматического составления. Данные навыки позволяют сделать использование поисковых запросов более эффективным и быстрым.

Поисковые системы обычно не учитывают регистр написания поисковых слов, то есть не имеет значения с большой или с маленькой буквы пользователь написал свой поисковый запрос (с включенной или выключенной клавишей Caps Lock) или некоторые слова при поиске. Только в некоторых исключительных случаях регистр слов учитывается для улучшения эффективности при поиске.

Если при поиске не удалось найти какой-либо интернет-ресурс, содержащий некоторые слова из текущего поискового запроса, то выдача результатов поисковыми машинами может производиться по словам, из списка которых исключены ненайденные слова.

Если требуется при поисковом запросе найти интернет-ресурсы, содержащую некоторую цитату, т.е. некоторые набор слов в определенном порядке, при этом сами слова должны быть найдены в определенной, написанной пользователем форме, то необходимо данную цитату заключить в двойные кавычки.

Любой запрос при передаче поисковой системе принимается и обрабатывается в соответствии с правилами языка запросов. Языки запросов подобно некоторой формуле, согласно которой сопоставляются все интернет-ресурсы, и поисковыми машинами выдаются те, которые удовлетворяют данной формуле. Например, при запросе, содержащем единственное слово, будет получен результат, в котором содержатся хотя бы одно использование этого слова. При этом при использовании в поиске некоторых слов, поиск осуществляется также с использованием всех видоизменений этих слов. Например, при поиске слова «машина» будут также найдены документы «машины», «машину», «машине», «машин» и т.д. При поиске с учетом морфологии учитываются такие аспекты слов как: часть речи слова(глагол, прилагательное, деепричастие, существительное и т.д.), форма заданного слова (склонение слова, его падеж, род слова и т.д.). Однако поиск слов не производится по однокоренным словам типа «ключ» и ключник. Если требуется осуществить поиск только с использованием конкретных форм слова, то необходимо заключить данное слово в двойные кавычки (упоминалось ранее) или использовать расширенный поиск, который существует практически у всех популярных поисковых ресурсов.

## Стоп-слова

В поисковых запросах фигурируют такое понятие как стоп-слова. Понятие стоп-слов используется для слов, не имеющих существенной информации в поисковом запросе. Данные слова обычно являются самыми часто употребляемыми словами и выражены, например, частицами, предлогами или артиклями. Стоп-слова чаще всего только замедляют поиск необходимой информации и отрицательно сказываются на получении действительно необходимой информации при поиске. Если все же необходим учет этих слов при поиске, то необходимо взять эти слова в двойные кавычки. Стоп-слова можно условно разделить на два типа:

1. Общие слова, например частицы, предлоги, числительные (написанные цифрами). Эти слова встречаются практически во всех документах, поэтому обычно не несут какой-либо смысловой нагрузки. Однако двузначные числа и имеющие более количество цифр обычно уже не выделяются поисковыми системами как слова, не несущие в себе смысловой нагрузки. Эти числа могут быть датами, номерами телефонов т.е. полезной информацией при поиске.
2. Некоторые зависимые слова. Поиск может быть осуществлен и без этих слов, но упоминание их используется слишком часто в других документах.

Обычно каждый поисковый ресурс составляет свой список стоп-слов, однако существуют самые повторяющиеся слова, которые используются почти всеми поисковыми сервисами. Ниже приведен список некоторых подобных слов.

Впереди, сто, если, когда-нибудь, несколько, кто-то, серьезно, однако, среди, должны.

Однако, стоит отметить, что хотя понятие стоп-слов в настоящее время используются некоторыми поисковыми системами, крупнейшие поисковые системы используют другие алгоритмы, автоматически не учитывающие стоп-слова при поиске.

## Ограничение расстояния между словами

Существует такое понятие как ограничение расстояния между словами в запросе. Это понятие выражается в виде целого положительного числа и используется в случаях, когда расстояние между какими-либо двумя найденными словами, содержащимися в поисковом запросе превышает это наперед заданное количество слов. Для каждого запроса в большинстве поисковых системах это значение устанавливается равным в 40 слов. Документ или интернет-ресурс будет выдан в качестве конечного результата поисковыми машинами в том случае, если расстояние между всеми найденными словами будет меньше этого целого положительного числа. Это число имеет название ограничение контекста. Размер контекста можно изменять с помощью синтаксической конструкции (ограничение контекста, запрос пользователя). Например, с помощью запроса пользователя «(2, невероятные истории)» (В разных поисковых системах синтаксис может отличаться). Однако, если запрос состоит из имен собственных, то ограничение расстояния между словами вычисляется исходя из количества слов в поисковом запросе по формуле: , где n – количество слов в запросе пользователя. Данное ограничение устанавливается по причине того, что между словами, являющимися частями некоторого имени собственного, не может быть много слов. Например, по запросу «Андреа Сарто» будут найдены документы, в контексте которых найдено имя Андреа дель Сарто, но будут проигнорированы интернет-ресурсы, содержащие имена, совпадающие частично с именем в запросе, но при этом другая часть может иметь другое имя, а, следовательно, найденное имя будет уже относиться к другому человеку.

## Операторы

В данном параграфе также были использованы материалы со страницы:

[4] Операторы поисковых систем «Google» и «Яндекс»

https://1ps.ru/blog/dirs/2016/operatoryi-poiskovyix-sistem-google-i-yandeks

Операторы применяются в запросах, содержащих два и более слова. Основные операторы являются бинарными, т.е. содержат левую и правую части, каждая часть при этом также является запросом, который может содержать также в свою очередь операторы. Рассмотрим три основных видов операторов:

1)And

2) OR

3) NOT

Первый оператор And выполняет логическую операцию “и” и, соответственно, делает обязательным условие выполнения левой и правой частей оператора. Например, при написании запроса пользователем: «Мандарины And Яблоки» пользователем будет получен результат веб-ресурсов, содержащих одновременно слово «Мандарины» и слово «Яблоки».

При использовании логического оператора OR будет произведен поиск интернет-ресурсов, в которых будет удовлетворено хотя бы одно из двух условий оператора OR. Например, при запросе пользователем: «Лампы OR светильники» будут выданы результаты, в которых встречается либо слово «Лампы», либо слово «Светильники». В поисковых системах также обозначается как «|».

Оператор NOT выполняет логическую операцию “НЕ”. Левая часть данного оператора является обязательным условием выполнения при поиске документа, правая же часть оператора является обязательным условием невыполнения при поиске. Например, при запросе пользователем: «Мандарины NOT Яблоки» будут получены результаты, в которых будет присутствовать слово «Мандарины», но при этом отсутствовать слово «Яблоки». В поисковых системах также обозначается как «-».

Однако, основные вышеперечисленные операторы в различных поисковых системах могут иметь иное обозначение. Также существуют другие операторы. В различных поисковых системах одни и те же операторы могут иметь различное обозначение. Например, оператор, обозначаемый как «~» в поисковой системе «Яндекс» осуществляет функцию поиска, в котором слово после оператора не содержится в одном предложении вместе со словом, стоящим до оператора. Однако, в поисковой системе «Google» аналогичное обозначение является обозначением оператора, который осуществляет поиск слова, стоящего после данного оператора и его синонимов.

Таким образом, у каждой поисковой системы свой набор операторов, каждый из которых может иметь аналогичное обозначение в других поисковых системах, или иметь исключительно уникальное обозначение. Приведем некоторые примеры подобных операторов.

Рассмотрим некоторые документные операторы, т.е. операторы, которые уточняют поисковый запрос с помощью указания информации, относящейся к служебной информации страницы. Эта информация обычно содержится в теге head html-кода страницы. Обычно такого рода операторы указываются после поискового запроса, при этом между оператором и поисковым запросом ставится пробел.

Оператор «url:» осуществляет поиск по страницам, которые размещены исключительно по адресу, прописанному после двоеточия.

Оператор «lang:» осуществляет поиск документов на определенном языке.

Оператор «filetype:» осуществляет поиск только в документах с определенным расширением.

Вышеперечисленные операторы используются в поисковых системах «Яндекс» и «Google».

## Понятия полноты и точности

При поиске информации в интернете двумя важнейшими характеристиками поиска являются *полнота* и *точность.* Данные компонентыможно объединить названием релевантность, то есть соответствие запрашиваемых данных и получаемых результатов.

В такой характеристике как полнота выражено отношение количества найденных документов ко всему количеству документов, удовлетворяющих данному запросу. Например, если всего в сети Интернет существует 1000 различных интернет-ресурсов, удовлетворяющих запросу , а поисковая система выдала 800 интернет-ресурсов, которые удовлетворяют запросу , то полнота поиска будет равна = 0,8. Соответственно, из данного определения можно сделать вывод, что чем больше полнота поиска, тем выше вероятность, что пользователь сможет найти необходимый для него интернет-ресурс. Действительно, чем больше удается поисковой машине найти документов, удовлетворяющих некоторому запросу из всех существующих документов, тем больше вероятность, что среди найденных документов будет нужный документ (при условии, что он - нужный пользователю документ вообще существует в интернете)

Рассмотрим также такой параметр поисковой машины как точность. Данная характеристика является отношением количества документов, соответствующих запросу пользователя ко всем найденным документам. Например, при запросе пользователя «Коллекция избранной литературы» результат: «Избранная коллекция бабочек, упоминаемая в русской литературе» выдается поисковой машиной только вследствие наличия в данном интернет-ресурсе этих слов, но при этом не является действительно интересующей пользователя информацией. Таким образом, если среди 1000 найденных документов 700 относятся к документам, которые действительно запрашиваются пользователем, то точность поиска равна: = 0,7.

Важно отметить, что показатели полноты и точности взаимозависимы. Так, увеличивая полноту поиска, одновременно снижается его точность. При этом при увеличении точности поиска необходимым условием данного действия является снижение полноты поиска.

Однако, несмотря на то, что такие характеристики поисковых систем как полнота и точность являются важнейшими характеристиками, существуют и другие немаловажные характеристики такие, как актуальность найденной информации. Данную характеристику можно выразить в виде разницы между временем занесения документа в индексную базу поисковой системы и временем публикации данного документа в сети Интернет. Большинство документов, появляющихся в глобальной сети, не нуждаются в немедленной индексации. При условии, что некоторый пользователь запрашивает некоторую информацию, которая содержится в документе, еще не проиндексированным поисковой системой, наиболее вероятно, что интернет-пользователь будет удовлетворен результатами поисковой системой уже проиндексированных документов. Например, при поисковом запросе пользователя «Редкие растения» пользователь скорее всего не испытает каких-либо затруднений в поиске, если информация некоторого сайта, посвященного редким растениям, не была еще проиндексирована. Однако, иногда существует острая необходимость в быстрой индексации информации каких-либо интернет-ресурсов. Чаще всего подобная информация относится к новостной категории. Такая информация имеет острую необходимость быть проиндексированной в кратчайшие сроки, поэтому она индексируется крупными поисковыми системами с помощью так называемой «быстрой базы», обновляемой несколько раз в день.

Также немаловажной характеристикой поисковой системы является скорость поиска необходимой информации. Безусловно, даже лишняя секунда среднестатистического времени ожидания пользователем результатов запроса может крайне негативно отразиться на популярности поисковой системы.

Также важным фактором хорошей работы поисковой системы является удобная для пользователя форма представления найденной информации. Как уже упоминалось ранее, далеко не все результаты запроса пользователя могут удовлетворить потребности пользователя, поэтому поисковым системам необходимо организовывать выдачу как можно более полной информации в наиболее сжатом виде для осуществления пользователем наиболее быстрого и эффективного поиска необходимых документов среди полученных результатов.

## Структура страницы поисковой выдачи поисковых систем

В данном параграфе были использованы материалы со страницы:

[5] Страница результатов поиска <https://ru.wikipedia.org/wiki/Страница_результатов_поиска>

также были использованы материалы со страницы:

[6] Поисковая и органическая выдача систем «Яндекс» и «Google»

<http://loleknbolek.com/slovar-terminov/poiskovaya-i-organicheskaya-vydacha-sistem-yandeks-google-i-kak-ona-formiruetsya>

а также были использованы материалы со страницы:

[7] Поисковая выдача поисковых систем

<http://pgdv.ru/samostoyatelno/poiskovaya-vydacha>

Рассмотрим структуру страницы поисковой выдачи популярных поисковых систем.

1. Основная часть страницы, содержащая в себе обработанную информацию о части интернет-ресурсов, найденных по запросу. (Органические результаты)
2. Контекстные объявления, размещаемые за определенную плату.
3. Область, называемая шорткатом, которая может содержать ответ на некоторый вопрос пользователя. Данный ответ на запрос содержит сама поисковая система до ввода запроса пользователем, а не некоторый интернет-ресурс, на который поисковая система может дать ссылку вместе с ответом на данный вопрос, взятого с данного интернет-ресурса. Также область шорткат может содержать в себе предложение исправления опечаток в запросе пользователя или некоторую другую полезную для интернет-пользователя информацию, касающеюся введенного им поискового запроса.
4. Элементы управления процессом поиска. Их можно разделить на две категории:
5. Поле, содержащее область для ввода поискового запроса, а также ссылку на страницу выдачи результатов введенного пользователем запроса.
6. Ссылки для перехода на следующую страницу поиска, предыдущую, или переход на следующую или предыдущую страницы с некоторым интервалом. Также эта область имеет ссылки для поиска видео или фотоматериалов по введенному запросу пользователя.
7. Область, содержащую запросы, которые могут быть связаны с запросом пользователя. К ним также могут относиться запросы, имеющие аналогичный смысл со смыслом введенного запроса пользователем, но переформулированные в форму, которая с большей вероятностью сможет удовлетворить требования интернет-пользователя.

Все области страницы поисковой выдачи направлены на более эффективную работу поисковых систем, однако наибольшую зависимость из вышеперечисленных областей оказывает так называемая органическая область т.к. именно в ней находятся непосредственно все результаты поискового запроса интернет-пользователя. Рассмотрим структуру основной части страницы поисковой выдачи:

1. Заголовок найденного интернет-ресурса. Обычно в качестве заголовка используется заголовок, прописанный в теге Title найденной интернет-страницы. Реже используются заголовки, прописанные в тегах h1, h2, h3, h4, h5, h6.
2. Небольшое изображение, относящееся к тематике сайта, на котором был найден интернет-ресурс (иконка). Обычно данное изображение находится в шаблоне сайта с названием favicon и имеет расширение ico.
3. Сниппет, являющийся неким кратким описанием найденной интернет-страницы.
4. Расширенный сниппет, содержащий некоторую более конкретную информацию о найденном интернет-ресурсе. Чаще всего в расширенный сниппет входят номера телефонов или адреса компаний, предлагающих свои услуги, соответствующих поисковому запросу пользователя.

**Расширенный поиск**

В данном разделе были использованы материалы со страницы:

[8] Расширенный поиск на yandex.ru

<https://yandex.ru/search/advanced?&lr=2>

Также были использованы материалы со страницы:

[9] Расширенный поиск на google.ru

<https://www.google.ru/advanced_search?hl=ru&fg=1>

Даже при эффективной работе поисковой системы пользователю нередко достаточно проблематично среди сотен найденных документов с помощью поискового запроса найти действительно нужные. Поэтому крупнейшие поисковые системы для улучшения работы их сервиса работают над повышением эффективности работы поисковой системы с помощью предоставления пользователю возможности задания дополнительных параметров поиска. В каждой поисковой системе, имеющей такую опцию, набор параметров поиска может сильно различаться, однако, безусловно, существует ряд закономерностей у поисковых систем. Рассмотрим основные из них:

* В крупнейших поисковых системах используется возможность выбора языка, на котором написаны необходимые пользователю документы.
* Также поисковыми системами предоставляется возможность выбора некоторого временного диапазона последнего обновления документов, которые стремится найти пользователь. Данная дата последнего обновления документа также может быть одновременно и датой создания данного документа.
* Возможность выбора региона, на котором потенциально находится необходимая пользователю информация, присутствует в крупнейших поисковых системах.
* Формат документов, содержащих необходимую информацию.
* Возможность задания сайта или домена, на котором пользователь желает осуществить поиск.
* Возможность осуществления поиска исключительно в тексте страниц, или, например, в заголовках страниц.

Также, различные поисковые системы имеют и другие параметры поиска, аналоги которых не имеются в других поисковых системах, однако основные параметры, уточняющие поиск, одинаковые. Безусловно, большинство данных параметров можно задать с помощью языка запросов, однако, пользователь может потратить существенно больше времени на осуществление поиска, прописывая поисковые запросы вручную, а для некоторых пользователю, не изучавшему языка запросов, понадобится дополнительное время для его изучения.

Расширенный поиск в большинстве случаев повышает эффективность работы пользователя с поисковыми системами, однако пользователю необходимо при выборе дополнительных критериев поиска быть уверенным в достоверности информации, которую несет каждый дополнительный параметр поиска т.к. документы, не удовлетворяющие данному параметру, не будут выданы поисковыми системами. Слишком подробный поиск существенно сужает область поиска документов, что может негативно сказаться на эффективности поиска.

# Основные положения принципов работы

В данном разделе, содержащем также такие параграфы как: «Модуль индексирования», «базы данных поисковых систем», «системы выдачи результатов» были использованы материалы со страницы:

[10] Состав и принципы работы поисковой системы

<http://www.web-shpargalka.ru/printsip-raboti-poiskovih-sistem.php>

Перед рассмотрением основных принципов работы стоит отметить, что поиск поисковой системой осуществляется исключительно с использованием тех интернет-ресуров, которые были уже занесены в базу поисковой системы. Соответственно, если некоторый интернет-ресурс не был занесен в базу данных некоторой поисковой системы, то результат, ссылающийся на данный интернет-ресурс, не будет выдан данной поисковой системой при любом запросе пользователя.

## Модуль индексирования

Условно, поисковую систему можно разделить на три группы. Первая группа отвечает за сбор необходимой информации в интернете. Также, эту группу можно назвать модулем индексирования. Данную группу можно разделить на три подгруппы:

* Первая подгруппа условно называется spider (паук). Spider – это некоторая программа, скачивающая веб-страницу по текущей ссылке. По принципу действия данная программа имеет сходство с обычным браузером т.к. браузер также производит загрузку текущей интернет-страницы. Однако обычный браузер отображает интернет страницу в графическом виде для удобной работы пользователя с интернет-ресурсом, в то время как данная программа сохраняет текущую интернет-страницу для ее дальнейшей обработки. Помимо html-кода страницы также сохраняется дата скачивания данной интернет-страницы, URL-ссылка интернет-ресурса, а также http-заголовок ответа сервера.
* Следующая подгруппа называется Crawler (переводится как путешествующий паук). Crawler - также некоторая программа, осуществляющая поиск всех ссылок на скаченной с помощью программы Spider интернет-страницы. При поиске ссылок осуществляется поиск тегов, заключающих ссылки, самый распространенный тег, встречаемый при поиске – тег <a>, а также менее распространенные такие как <area>, <frame>. Далее, после обнаружения всех ссылок программа Crawler производит выбор ссылки, по которой данная программа будет далее осуществлять переход, после которого программа Spider проделает то же действие, что и с предыдущей страницей.
* Последним компонентом группы, осуществляющим сбор необходимой информации, является Indexer (индексатор), производящий разделение загруженной интернет-страницы для последующего ее анализа. Индексатор позволяет сохранять загруженные интернет-ресурсы в виде удобном для работы поисковой системы. При этом информация, находящаяся в разных тегах html-страницы, может иметь различный приоритет значимости. Например, информация, находящаяся в теге <p> - абзац, будет иметь меньший приоритет, чем информация, находящаяся в теге <h1> заголовок первого порядка.

## Базы данных поисковых систем

Ко второй группе поисковой системы можно отнести базу данных поисковой системы. Безусловно, база данных требует значительных ресурсов для хранения всей информации, принимающей участие в поиске. Так как поиск пользователем осуществляется исключительно в пределах базы данных некоторой поисковой системы, то актуальность информации на интернет-ресурсах определяется частотой обновления базы данных.

Третьим компонентом, влияющим непосредственно на эффективность и скорость осуществления поиска интернет-пользователем, является система выдачи результатов. Данная система также имеет название Клиент. Клиент одной и той же поисковой системы может находиться на различных компьютерах, не имеющих связь между собой, но данные сервера, безусловно, должны иметь доступ к общей базе данных некоторой поисковой системы.

## Системы выдачи результатов

Работу системы выдачи результатов можно разделить на несколько этапов:

* Вначале, для каждого документа, принимающего участие в поиске, формируется информационное окружение
* Далее модуль ранжирования получает обработанные данные для составления рейтинга каждого документа. Данный рейтинг должен отражать соответствие введенного запроса найденным документам. Например, документы, в которых встречаются только два слова из четырех слов, содержащихся в поисковом запросе пользователя, с наибольшей вероятностью будут иметь меньший рейтинг, чем документы, содержащие три слова. Однако рейтинг интернет-страниц зависит также и от других факторов, например, популярности данного интернет-ресурса.
* Далее, полученный рейтинг может быть изменен в соответствии с дополнительными условиями, задаваемыми, например, в расширенном поиске.
* Далее происходит формирование некоторого краткого описания каждого найденного интернет-ресурса, обычно содержащее максимальное количество ключевых слов. Данное описание также называется сниппетом, содержащем ключевые слова в запросе, которые обычно выделяются на странице выдачи поискового запроса. После чего происходит выдача результатов запроса пользователя на страницу выдачи поисковых запросов.

# Принципы и методы работы краулера BeeBot

В данном разделе, включающим в себя также параграфы: математическая формулировка задачи о многоруком бандите и алгоритмы ее решения, алгоритм Гиттенса, алгоритм UCB1, были использованы материалы научной статьи:

[11] Печников А. А. , Чернобровкин Д. И. Адаптивный краулер для поиска и сбора внешних гиперссылок  // журнал «Управление большими системами» *УБС*,2012, **36**, с 301–315

## Математическая формулировка задачи о многоруком бандите и алгоритмы ее решения

Задача краулера заключается в том, чтобы за ограниченное время найти максимально возможное количество внешних ссылок. Данная задача формулируется как задача о многоруком бандите и имеет несколько вариантов решения.

В процессе работы так называемого блуждающего паука – Краулера с некоторыми тематическими интернет-ресурсами формируется база данных внешних ссылок т.е. ссылок на другие интернет-ресурсы. Краулер BeeBot, предназначенный для работы с тематическими ресурсами, начинает работу с начальной страницы сайта. На начальной странице осуществляется сбор всей необходимой информации, включая сбор внутренних ссылок сайта т.е. ссылок, ведущих на другие интернет-ресурсы, находящиеся на том же домене, что и начальная страница сайта. Далее идет аналогичная обработка других страниц данного сайта в определенном порядке. Данный порядок образуется исходя из количества минимальных гиперссылок, размещенных на начальной странице и имеющих отсылку к данной странице. Чем больше подобных гиперссылок, тем раньше будет обработана интернет-страница.

Рассмотрим более подробно обработку каждой интернет-страницы. Сначала происходит попытка загрузки страницы и попытка преобразования ее в XML-документ. Если интернет-страницу загрузить не удалось, то происходит попытка загрузки страницы альтернативным методом т.е. интернет-страница скачивается в бинарном формате, а затем преобразуется в текстовый формат с кодировкой, которую выберет программа блуждающий паук. Если загрузка страницы по какой-либо причине все-таки не производится (например, ссылка оказалась устаревшей и ссылается на несуществующую интернет-страницу), то данная ссылка помечается как недоступная и больше не используется. Если загрузка начальной страницы удалась, то вначале происходит сбор ссылок, находящихся в теге <frameset>. Ссылки, находящиеся в данном теге, наравне с другими ссылками рассматриваются как внутренние и подлежат дальнейшей обработке. Далее происходит сбор ссылок, находящихся, например, в тегах <a>. Полученные ссылки приводятся к некоторому общему виду, например, приведение букв к нижнему регистру, а также, удаление символа «\» в конце ссылки уже разрешает конфликты одинаковых ссылок в большинстве случаев. Далее, ссылки, в зависимости от их типа помещаются в одну из трех таблиц: **External Links** – таблица внешних ссылок, **Internal Links** –таблица внутренних ссылок, **Bad Links** – таблица нерабочих ссылок. Рассмотрим структуру краулера BeeBot подробнее:

* Данный краулер состоит из базы данных, которая, в свою очередь, состоит из пяти таблиц:

1. Crawler Settings, содержащая настройки данного краулера, например, максимальный уровень страниц, который может быть использован краулером.
2. Таблица Seeds (в переводе с английского означает семена) содержит начальную информацию обо всех интернет ресурсах, с которых данный краулер начинает сканирование. Безусловно, т.к. сканирование некоторого сайта начинается со сканирования его начальной страницы, то данная таблица содержит в первую очередь доменное имя сканируемого сайта. Также данная таблица содержит поля полных и кратких названий сайта, что вполне облегчает работу пользователя, осуществляющего вручную редактирование списка сайтов, которые должны быть просканированы программой.
3. Таблица Bad Links, содержащая ссылки, которые по какой-либо причине не удалось обработать.
4. Таблицы Internal и External Links содержащие, соответственно, внутренние и внешние ссылки. Для определения, к какому из двух типов относится очередная ссылка, краулер проводит сопоставление доменных имен сканируемого сайта и ссылки. При их совпадении ссылка отмечается программой как внутренняя, иначе ссылка считается внешней.

* Также краулер BeeBot состоит из блока управления программой. Данный блок содержит возможность редактирования списка сайтов, подлежащих обработке, просмотр внешних гиперссылок, полученных в результате работы программы, а также возможность запуска программы, а также временной приостановки программы. После завершения приостановки программа продолжит работу с момента, на котором она закончила вследствие остановки.
* Третьим основным компонентом данного краулера является блок сканирования. Блок сканирования вначале использует данные таблицы Seeds, содержащейся в базе данных. Из данной таблицы извлекаются поочередно сайты и сканируются поочередно некоторое заданное количество страниц на данных сайтах.

## Математическая формулировка задачи о многоруком бандите и алгоритмы ее решения

Для начала стоит отметить, что так как по условию задачи время ограничено, а за данное ограниченное время требуется найти максимальное количество внешних ссылок на интернет-ресурсы, то вариант решения данной задачи путем сканирования всех интернет-страниц всех сайтов исключается. Также, так как вероятность нахождения на всех сканируемых сайтах одинакового количества внешних гиперссылок крайне мала, то способ решения задачи, основанный на сканировании на каждом сайте одинакового количества интернет-страниц, практически никогда не сможет привести к нужному результату.

В задаче о многоруком бандите для случайных величин индекс определяет номер так называемого игрового автомата, выбираемого игроком. Индекс определяет номер так называемой игры на некотором автомате. Игрок в процессе игры получает доходы , где - номер автомата, на котором игрок осуществляет игру, а индекс – порядковый номер игры. Данные доходы представляют независимые одинаково распределенные случайные величины. Игрок также может осуществлять игру на разных автоматах, и данные величины также будут являться независимыми. Так называемом игроком в задаче о многоруком бандите является краулер, а независимые случайные величины являются количеством внешних гиперссылок, которые были найдены на сайте на – ой сканируемой странице. Далее предположим, что время, требуемое для сканирования каждой страницы одинаково. Поэтому время, отведенное для решения данной задачи, может быть выражено в виде количества страниц (N), которое может быть просканировано за данное отведенное время.

Так как количество найденных внешних гиперссылок на сайте является положительным числом, то при , , где – количество сайтов, участвующих в данной задаче. Решающее правило – алгоритм, определяющий на каждом шаге выполнения программы на каком сайте будет просканирована следующая страница.

Суммарное количество внешних ссылок на интернет-ресурсы, полученных после того, как были просканированы n любых интернет-страниц сайтов, участвующих в задаче, можно обозначить следующим образом:

, где

## Алгоритм Гиттенса

1. Вначале задаем некоторое число
2. Для каждого сайта просканировать первые страниц.
3. Для каждого сайта задать значение:

Таким образом, для каждого сайта получаем количество всех найденных на нем внешних ссылок на всех его интернет-страницах.

1. Для каждого сайта вычисляем значение индексов Гиттинса: , а также задаем количество просканированных на каждом сайте страниц . Таким образом, индексы Гиттенса обозначают для каждого сайта среднее количество внешних ссылок уже просканированных страниц.
2. Находим номер сайта, содержащий максимальное значение индекса Гиттенса:
3. Просканировать страниц на - ом сайте и изменить значение суммы, соответствующей -ому сайту: = , а также изменить значение количества просканированных страниц для данного сайта: и пересчитать значение Гиттинса для данного сайта:
4. Если сумма количества всех просканированных страниц на всех сайтах меньше , то возвращаемся к пункту 5, иначе завершаем выполнение алгоритма, при этом получив S = внешних ссылок.

**Алгоритм UCB1**

1. Вначале задаем некоторое число
2. Для каждого сайта просканировать первые страниц.
3. Для каждого сайта необходимо задать значение: , (получаем для каждого сайта количество всех найденных на нем внешних ссылок на всех его интернет-страницах), а также необходимо задать количество просканированных на каждом сайте страниц .
4. Вычисляем средние значения количества найденных внешних гиперссылок для каждого сайта:
5. Далее вычисляем количество просканированных интернет-страниц на всех сайтах , а также осуществляем поиск интернет-страницы на всех сайтах с наибольшим количество внешних гиперссылок:
6. Вычисляем индекс сайта , при котором значение + будет максимальным
7. Просканировать следующую страницу сайта : и увеличить количество просканированных страниц для данного сайта на единицу.
8. Увеличиваем переменную, содержащую количество внешних гиперссылок, найденных на сайте : = , t()), затем пересчитываем среднее значение количества найденных внешних гиперссылок для данного сайта: и увеличиваем переменную количества просканированных страниц на единицу:
9. Если , то возвращаемся к пункту 5, иначе количество найденных внешних ссылок на интернет-ресурсы: S =

Как отмечалось ранее, данный краулер предназначен для работы с сайтами, относящимся к определенной тематике. Использование данного алгоритма для сайтов, относящихся к неподходящей для данного алгоритма тематике, может существенно снизить эффективность работы краулера. Однако даже применяя вместо алгоритма Гиттинса алгоритм UCB1, или наоборот, в различных случаях можно получить другое количество внешних ссылок. Например, при сканировании страниц сайтов вузов с помощью алгоритма Гиттинса, краулер находил меньше гиперссылок, чем при сканировании с помощью алгоритма UCB1. Это объясняется тем, что сайты с подобного рода тематикой имеют большой разброс количества внешних ссылок на другие интернет-страницы. То есть на одной странице может быть малое количество внешних ссылок на другие интернет-страницы, а на следующей странице этого же сайта большое количество внешних ссылок. Таким образом, данный фактор усиливает эффект непредсказуемости найденного количества внешних ссылок. Из-за этого интернет-страницы, содержащие большое количество требуемых ссылок, могут быть не просканированы вследствие того, что начальные страницы сайтов, на которых находятся нужные интернет-страницы, содержат достаточно малое количество внешних ссылок, или не содержат их вообще. Алгоритм Гиттинса, сканирующий последовательно сайты с наибольшим средним количеством найденных внешних ссылок, в немалом количестве случаев идет по неверному пути и фактически не добирается до интернет-страниц, содержащих действительно максимальное количество внешних ссылок. В отличии от алгоритма Гиттинса, алгоритм UCB1 достаточно быстро исключает из области сканирования те сайты, на которых при долгом поиске не увеличивается существенно количество внешних гиперссылок. При выборе очередного индекса сайта, который будет выбран для сканирования в слагаемом слишком большое количество страниц отрицательно сказывается в пользу выбора данного сайта.

Однако у алгоритма Гиттинса существует преимущество перед алгоритмом UCB1 в случае сайтов с равномерно распределенным количеством внешних ссылок. Это преимущество заключается в том, что при сканировании сайтов, содержащих интернет-страницы с примерно одинаковым количеством внешних ссылок, алгоритм UCB1 сканирует последовательно практически одинаковое количество внешних ссылок на всех сайтах. Алгоритм Гиттинса может просканировать не одинаковое количество интернет-страниц на всех сайтах, но просканировав то же количество интернет-страниц, алгоритм Гиттинса получит примерно такое же количество внешних ссылок, что и алгоритм UCB1. Преимущество алгоритма Гиттинса заключается в том, что он может просканировать то же количество внешних ссылок, что и алгоритм UCB1, только при этом требуя меньшее количество вычислительных мощностей компьютера, что может сэкономить время для выполнения данной операции.

# Базы данных поисковой системы

В данном разделе были использованы материалы из статьи:

[12] Никитенко Е,В., Р.В. Заровский, М.А. Сдитанов Автоматическая система поиска информации в текстах с использованием алгоритмов нечеткой логики // Технические науки и технологии. Черниговский национальный технологический университет, 2012, №4(61), с 150-157

А также были использованы материалы из учебного пособия:

[13] Яковлев С.А. Корпоративные информационные системы ч.1. 2010. 117 с.

Для начала стоит отметить, что существуют метапоисковые системы. Их отличие от поисковых систем заключается в том, что они не имеют свою собственную базу данных. Данные системы используют базы данных других поисковых систем путем передачи запроса пользователя этим другим поисковым системам. Поисковые системы выдают результаты по некоторому запросу используемой пользователем метапоисковой системе, после чего данная система использует собственные алгоритмы для обработки и ранжирования результатов поиска. Метапоисковые системы могут быть полезны, когда требуется найти все документы в сети, удовлетворяющие запросу пользователя т.к. разработчики большинства крупнейших поисковых систем в последнее время стремятся предоставить как можно более точные результаты поиска за минимальное время, документы, имеющие малое количество посещений, или не удовлетворяющие в полном объеме потребности пользователя, не используется при выдаче результатов поиска.

После того, как программа Crawler осуществляет загрузку страниц, которые сохраняются на некотором так называемом сервере накопления (Store Server), индексатор производит получение всех слов на некоторой загруженной странице или на некотором загруженном документе, после чего компонент Indexer поисковой системы производит индексацию документа. Рассмотрим подробнее данный процесс:

При сохранении документов в виде некоторого вектора слов в базу данных поисковой системы, индексатор сохраняет документы в виде прямого индекса т.е. у каждого документа есть его некоторый порядковый номер, по которому отсортированы все документы в базе данных поисковой системы. Каждый документ представляет собой некоторый отсортированный набор слов, для каждого из которых указано количество вхождений данного слова в документ, номера позиций данного слова в документе, а также формат вхождения данного слова в документе, при этом исключаются так называемые стоп-слова, не имеющие смысловой нагрузки, о которых упоминалось в данной работе ранее. Под номером позиции слова подразумевается порядковый номер некоторого слова в данном документе. При записи формата вхождения слова в документ идет запись тега, в котором было найдено данное слово, если индексация, допустим, html-страницы. Нахождение слов в различных тегах будет оказывать влияние на приоритет выдачи результатов поиска при запросе. Это отражено в понятии вес слова. Для каждого тега присваивается свой вес, который учитывается поисковыми системами при сортировке. Например, информация, находящаяся в теге title, обычно имеет меньший вес, чем слова, находящиеся в теге content. Соответственно, чем больше вес слова, тем большее влияние оно оказывает на повышение рейтинга соответствующей ему интернет-страницы в результатах выдачи.

Стоит отметить, что максимальное количество вхождений слова в документ может быть ограничено, что влечет за собой ограничение максимального количества позиций слова в документе. Данное ограничение устанавливается разработчиками каждой поисковой системы самостоятельно и зависит от методик обработки информации разработчиками поисковой системы.

Для осуществления поиска информации документы сортируются с помощью обратного индекса т.е. по словам. Для этого каждому слову присваивается некоторый порядковый номер, а для каждого слова сохраняется вся информация о вхождении данного слова во все документы, в которых данное слово встречается.

При индексировании документов иногда обрабатываться может только некоторое количество слов, начиная с начала документа. Например, первые пятьсот слов. Однако обрабатываемая поисковыми системами информация может находиться в различных форматах данных, что может существенно затруднить обработку поисковыми системами информации. Поэтому для того, чтобы сделать хранение данных в базах данных поисковых систем более универсальной, было введено понятие *Поискового образа документа (ПОД).* Данное понятие обозначает способы хранения информации об индексируемых документах для осуществления их эффективного поиска. Одна из самых распространенных моделей в поисковых системах – векторная модель. В данной модели каждому документу сопоставляется некоторый фиксированный набор слов, так называемых терминов, которые зафиксированы в поисковых системах. Для всех документов этот список терминов одинаков, однако параметры для этих терминов разные. Каждому документу и каждому термину сопоставлено некоторое число от 0 до 1, характеризующее, насколько данный документ можно охарактеризовать данным термином. В более ранних векторных моделях данное число равнялось либо нулю, либо единице. Стоит отметить, что в поисковых системах проводилось разбиение на системы с фиксированным словарем, в котором добавление новых слов осуществлялось исключительно разработчиками системы, и на системы со свободным словарем, в который новые слова добавлялись автоматически при индексировании новых документов.

Пример векторной модели в поисковой системе, содержащей N документов и M терминов:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Термин №1 | … | Термин № M |
| Документ 1 | 0,121 | … | 0,154 |
| … | … | … |  |
| Документ N | 0,925 | … | 0,713 |

Скорость осуществления поиска в поисковых системах зависит в первую очередь от архитектуры индекса. Различные поисковые системы при индексации документов, сохраняют информацию о них в различных видах, в соответствии с алгоритмами поиска, которые они используют. Например, поисковая система Yahoo в качестве прямого списка сохраняет массив поисковых образов страниц, а в качестве инвертированного списка сохраняет список ключевых слов, которым сопоставляются документы, содержащие данные слова, а также позицию данного слова в данном документе.

# Обработка результатов запроса пользователя

Общая идея обработки поискового запроса состоит в том, что сначала поисковый запрос разбивается на отдельные слова. Далее для каждого слова поисковая система проводит поиск необходимых документов с использованием поисковых алгоритмов системы. В классическом варианте реализации поисковой системы поиск для каждого слова проводится с использованием инвертированного списка т.е. с использованием исключительно базы данных, в которой документы отсортированы по словам. При этом каждому слову присваивается некоторый идентификатор, который соответствует уникальному идентификатору слова в словаре. Затем из списков найденных документов, соответствующих словам создается новый, содержащий только те документы, в которых встречаются все слова из запроса пользователя. (*Как упоминалось в данной работе ранее, в некоторых случаях, когда, например, поисковой системе не удается найти документы, содержащие все слова, в результатах поисковой выдачи могут содержаться документы, содержащие только часть слов из запроса пользователя*). Таким образом,. значение каждого слова, участвующего при поиске одинаково. Однако существует другой метод сортировки, который уже устаревает, но все-таки нередко используется при поиске. Отличие данного метода в том, что данный метод сортирует найденные документы с учетом разности частот встречаемости слов полученных из запроса пользователя, а не с использованием вычисления значимости каждого слова в документе отдельно. Т.е., например, при запросе пользователя «микропроцессор автобус» несмотря на то, что слово «автобус», встречается во всех документах чаще и в более значимых тегах, чем слово «микропроцессор», при сортировке результатов данным методом может первыми выводиться документы, в которых встречается слово микропроцессор из-за того, что этот термин встречается реже в документах.

## Поиск с использованием частоты термина

В данном параграфе, а также в параграфе «Векторный поиск» были использованы материалы статьи, расположенной по адресу:

[14] Основные подходы алгоритмов поисковых систем

<http://blog.aweb.ua/osnovnye-algoritmy-poiskovoj-sistemy/>

а также научной статьи: .

[15] М.А.Демихов Методы нечеткого поиска в информационных системах // научный журнал «Моделирование, оптимизация и информационные технологии» (МОИТ). Воронежский институт высоких технололгий 2015 №2(9)

Для данного метода, так называемого «Частота термина» используется понятие *обратная частота термина* **IDF**, которая выражается в логарифме отношения количества всех документов в поисковой системе и количества документов, в которых встречается данное слово (термин). Таким образом, чем слово чаще используется в документах, тем меньшее значение IDF оно будет иметь. В итоге, часто встречаемые слова будут иметь меньшее значение при поиске, а редкие слова наоборот.

Для каждого документа, умножая количество повторений каждого слова из запроса пользователя в документе на обратную частоту данного термина и складывая эти произведения, можно получить некоторое значение, характеризующее соответствие запроса пользователя документу. По данным значениям и осуществляется сортировка документов, полученных при поиске. Рассмотрим на примере запроса пользователя «Дешевый ремонт»:

Допустим, что в некоторой поисковой системе IDF термина «дешевый» равен 0,45, а IDF термина «ремонт» равен 0,85.

Допустим, что в документе 1 слово дешевый встречается 7 раз, а слово ремонт 5 раз, а в документе 2 слово дешевый встречается 5 раз, а слово ремонт 4 раза.

Располагая данной информацией, вычисляем оценку:

Оценка документа 1:

Оценка документа 2:

Таким образом, документ 1 получил более высокую оценку, а значит больший приоритет при выдаче результата поискового запроса.

## Векторный поиск

Как упоминалось в данной работе ранее, сейчас поисковые системы чаще работают с векторной моделью данных, которая представляется в виде таблицы соответствия терминов документам. Каждому термину сопоставляется вектор вещественных чисел, каждое из которых отражает степень соответствия данного термина некоторому документу. Одновременно для каждого документа в таблице соответствия можно выделить вектор вещественных чисел, каждое из которых отражает степень соответствия данного документа и некоторого термина. Для того, чтобы вычислять данные вещественные числа, необходимо получить начальные данные и некоторый ряд промежуточных компонент, которые используются для вычисления конечных значений. Рассмотрим таблицы, содержащие все вышеперечисленные компоненты для поискового запроса: «Купить ноутбук недорого». При этом документом будет являться интернет-страница

http://qwertyshop.com.ua/noutbuki

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Термины,  содержащиеся  в запросе | Компоненты, связанные со словами из запроса | | | | Компоненты, связанные с документом | | | Конечная оценка |
| Слова из запроса | **tf** | **df** | **idf** | **Wt, q** | **tf** | **Wf** | **Wt, d** |  |
| **Купить** | 1 | 999  .000  .000 | 1,67253237 | 1,67253237 | 5 | 25 | 0,07224 | 0,120830558 |
| **Ноутбук** | 1 | 99  .100  .000 | 2,676024203 | 2,676024203 | 69 | 4761 | 0,99697 | 2,667911206 |
| **Недорого** | 1 | 89  .400  .000 | 2,720760339 | 2,720760339 | 2 | 4 | 0,02890 | 0,078623528 |
| Суммарная оценка для всех слов из запроса:2,867365291а: | | | | | | | | |

Рассмотрим аналогичную страницу для вышерассмотренного запроса «Купить ноутбук недорого» для интернет-страницы: http://centr-noutbukov.ru/gde-kupitj-noutbuk-deshevo

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Термины,  содержащиеся  в запросе | Компоненты, связанные со словами из запроса | | | | Компоненты, связанные с документом | | | Конечная оценка |
| Слова из запроса | **tf** | **df** | **idf** | **Wt, q** | **tf** | **Wf** | **Wt, d** |  |
| **Купить** | 1 | 999  .000  .000 | 1,67253237 | 1,67253237 | 16 | 256 | 0,38760 | 0,648275665 |
| **Ноутбук** | 1 | 99  .100  .000 | 2,676024203 | 2,676024203 | 38 | 1444 | 0,92055 | 2,46342213 |
| **Недорого** | 1 | 89  .400  .000 | 2,720760339 | 2,720760339 | 2 | 4 | 0,04845 | 0,131821269 |
| Суммарная оценка для всех слов из запроса: 3,243519064 | | | | | | | | |

А также рассмотрим аналогичную таблицу для интернет-страницы http://vsebooki.ru

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Термины,  содержащиеся  в запросе | Компоненты, связанные со словами из запроса | | | | Компоненты, связанные с документом | | | Конечная оценка |
| Слова из запроса | **tf** | **df** | **idf** | **Wt, q** | **tf** | **Wf** | **Wt, d** |  |
| **Купить** | 1 | 999  .000  .000 | 1,67253237 | 1,67253237 | 12 | 144 | 0,20902 | 0,349592556 |
| **Ноутбук** | 1 | 99  .100  .000 | 2,676024203 | 2,676024203 | 56 | 3136 | 0,97543 | 2,610264173 |
| **Недорого** | 1 | 89  .400  .000 | 2,720760339 | 2,720760339 | 4 | 16 | 0,06967 | 0,189564355 |
| Суммарная оценка для всех слов из запроса: 3,149421084 | | | | | | | | |

* В данных таблицах первый столбец tf содержит количество данного термина в запросе пользователя. Обычно в данном столбце значения всегда равны единице.
* Столбец df означает количество документов, которые содержат данный термин (называется частотой документа)
* Столбец idf – обратная частота документа. Вычисляется как логарифм отношения общего количества документов, содержащихся в базе данных поисковой системы к количеству документов, содержащих данный термин.
* Столбец tf, то есть данный столбец равен произведению обратной частоты к количеству ввода данного термина пользователем в запросе.
* Далее второй столбец tf уже относится к параметрам, связанным с данным документом. В данном столбце заносится количество повторений каждого термина в документе.
* В следующем столбце Wf содержатся квадраты значений предыдущего столбца.
* В столбце Wt, d содержатся отношения второго столбца tf к корню суммы всех значений столбца Wf: ,
* В последнем столбце содержатся произведения для каждого термина соответствующих ему в столбцах Wt,q и Wt, d значений. Именно данные значения и являются конечными значениями, отражающими соответствия каждого термина каждому документу. Данные значения всех слов, которые содержатся в запросе пользователя, суммируются, после чего каждый документ получает некоторый суммарный рейтинг соответствия документа запросу пользователя.

Мною были проведены вычисления соответствия запроса «Купить ноутбук недорого» трем интернет-страницам. При сравнении итоговых вычислений можно сделать вывод, что первый интернет-ресурс с самым низким рейтингом должен быть выдан после выдачи других двух интернет-ресурсов, а второй интернет-ресурс должен быть выдан раньше остальных.

## Поиск с использованием релевантности страниц

В данном параграфе были использованы материалы презентации:

[16] Введение в поисковые системы. “Современные задачи теоретической информатики”

<http://logic.pdmi.ras.ru/~yura/modern/05.pdf>

также были использованы материалы, расположенные по ссылке:

[17] PageRank в поисковой системе «Google»

<http://wm-help.net/articles/article/25.05.20061167-14.html>

также были использованы материалы со страницы:

[18] Модели информационного поиска. PageRank «Алгоритмы для Интернета»

http://yury.name/internet/03ianote.pdf

При сортировке результатов поиска с использованием релевантности страниц в поисковой системе Google введено такое понятие как PageRank (PR) . PageRank – некоторое число, отражающее рейтинг каждой интернет-страницы в поисковой системе, который учитывает количество, качество ссылающихся на данную интернет-страницу ссылок. PageRank для каждой страницы подсчитывает вероятность нахождения пользователя на данной странице при условии, что пользователь осуществляет путешествие в сети Интернет. В поисковой системе «Яндекс» аналогично PageRank для каждой страницы вычисляется ВИЦ – Взвешенный индекс цитирования.

Рассмотрим ориентированный граф, состоящий из вершин и ребер. Предположим, что пользователь путешествует по сети интернет бесконечно долго, начиная в некоторой вершине. Обозначим с помощью вероятность того, что пользователь окажется в вершине (на некоторой странице через итераций. Т.к. для каждой страницы существует вероятность ее посещения при условии, что количество итераций (количество переходов на новую страницу) будет стремиться к бесконечности, то получаем:

Далее, пусть - вершины рассматриваемого графа, которые имеют ребра, ведущие в вершину ,

Таким образом, получаем основное уравнение PageRank:

- параметр затухания, обозначающий вероятность, что пользователь, загрузивший некоторую страницу, продолжит переходить по ссылкам. В большинстве случаев данный параметр равен 0,85.

- вероятность попадания пользователя в вершину , которая ведет на страницу

- количество ссылок (ребер), ведущих со страницы .

Если предположить, что мы вычисляем вероятность для всех страниц, то есть , то переходя к пределу, получаем:

Стоит отметить, что любая страница, не имеющая других страниц, ссылающихся на нее, в любом случае имеет значение PageRank не равное нулю, т.к. любая страница может быть взята в качестве стартовой.

# Параллельный поиск

В данном разделе были использованы материалы со страницы:

[19] Параллельный поиск в «Яндекс»

<http://tekseo.su/poiskovye-sistemy/koldunshhik-yandeks-kak-rabotaet-parallelnyjj-poisk.php>

У крупнейших поисковых систем множество документов хранится на различных сервисах. Распределение документов по сервисам происходит в соответствии с тематикой документов. При вводе поискового запроса пользователем в крупнейших поисковых системах поисковый запрос классифицируется и в соответствии с тематикой запрос, перенаправляется на некоторый сервис, и результаты запроса данного сервиса также выводятся пользователю при поиске. Обычно алгоритмы перенаправления на определенный сервис начинают свою работу после того, как был произведен поиск документов, но не была произведена еще выдача результатов по запросу. Данные алгоритмы анализируют тематику найденных документов по запросу пользователя, после чего производят передачу запроса пользователя уже на определенный сервер. В поисковой системе «Яндекс» данные системы называются «кубиками параллельного поиска». В данной поисковой системе существуют:

* «Кубик запросов». Его работа происходит следующим образом: в поисковом запросе пользователя производится поиск слов – маркеров. В случае обнаружения данных слов происходит передача запроса пользователя на соответствующий слову – маркеру сервис. Например, при запросе «слушать Битлз» слово «слушать» является словом-маркером, связанным с сервисом «Музыка».
* «Кубик результатов поиска» анализирует найденные по некоторому запросу документы, а затем определяет тематику большинства документов, найденных по заданному запросу, в том числе и благодаря тому, что поисковая система может определять тематику найденных интернет-ресурсов. Определив тематику запроса, происходит передача запроса на соответствующий сервис поисковой системы.
* «Кубик качества» используется в случае, когда пользователь вводит запросы, информация по которым находится в малом количестве документов. Поиск производится по различным сервисам. В случае точного ответа при поиске по сервисам будет выдан соответствующий результат.
* «Кубик переформулировок» использует запросы, схожие с запросом пользователя, однако дополненные в соответствии с наиболее популярными похожими запросам. Дополненные запросы уже могут быть точно классифицированы и отправлены на соответствующий сервер. Например, при запросе «Титаник» будут выведены результаты запроса с сервера «видео».

Если при поиске использовался кубик запросов, то результаты других кубиков не будут использованы. В противном случае для тематик, которые были выданы различными кубиками вычисляются некоторые универсальные коэффициенты. Тематика с наибольшим коэффициентом выбирается в качестве основной тематики, которая и используется при передаче запроса пользователя на определенный сервис.

# Классификация текстовых документов с использованием метода «Наивный Байес»

В данном параграфе были использованы материалысо страницы:

[20] Александр Сизов, Сергей Николенко Наивный Байесовский классификатор.

http://logic.pdmi.ras.ru/~sergey/teaching/mlstc12/sem01-naivebayes.pdf

А также материалы со страницы:

[21] Метод классификации «Наивный Байес»

<http://bazhenov.me/blog/2012/06/11/naive-bayes>

В основе теории положена формула Байеса:

вероятность документа в документах класса

- вероятность встретить документ d во всех документах

– вероятность принадлежности случайного документа классу

Требуется вычислить - вероятность принадлежности d – документа классу . Соответственно, данную вероятность необходимо вычислить для каждого класса из всех классов, которым принадлежат документы.

Далее, необходимо выбрать тот класс, вероятность принадлежности к которому документа будет максимальной. Данный класс и будет наиболее вероятным классом, к которому принадлежит документ. Стоит также отметить, что при подсчете всех вероятностей вероятность встретить документ во всех документах не будет меняться, следовательно, данное значение не будет оказывать влияния на формулу, а, следовательно, знаменатель дроби можно опустить. В итоге получаем формулу:

В данном классификаторе вероятности появления слов в некотором классе условно не зависят друг от друга. Следовательно, вероятность документа, состоящего из некоторого набора слов, можно представить в виде произведения вероятностей появления слов из данного документа в документах, принадлежащих определенному классу.

*(n – количество слов, входящих в документ)*

Таким образом, получаем:

Вероятность принадлежности случайного документа классу вычисляется как отношение количества документов в классе ко всему количеству документов:

- количество документов, относящихся к классу , - общее количество документов.

Соответственно, вероятность появления некоторого слова в классе вычисляется как отношение количества встречаемости этого некоторого слова к сумме частот встречаемости всех слов в классе :

, где – частота встречаемости слова в документах, относящихся к классу .

- список всех слов из класса

Однако если пользователь вводит некоторое слово, которого нет ни в одном из классов документа, то классифицировать данное слово не представляется возможным. Поэтому для избегания подобной ситуации к частоте встречаемости каждого слова стоит прибавить единицу. Таким образом, если нам попадается некоторое слово в запросе пользователя, которое не встречалось ни в одном документе, то частота данного слова будет равна единице:

Таким образом, класс, к которому классификатор относит документ, вычисляется по формуле:

Однако чтобы избежать грубой ошибки округления, стоит взять логарифм от данного выражения:

# «Наивный Байес» при поиске новостей

При решении задачи эффективного поиска документов, относящихся к определенной тематике, выберем новостную тематику. На вход пользователю необходимо будет подавать некоторые ключевые слова и фразы, интересующие пользователя в данной тематике, а на выходе серверу требуется выдавать некоторые новости, которые могут заинтересовать пользователя. При работе поискового сервера поисковой системы Яндекс, относящийся к новостной тематике, поиск осуществляется в основном либо с помощью некоторых конкретных параметров: обязательное вхождение в новости ключевых слов, введенных пользователем, датой публикации новостных ресурсов, регионы новостных ресурсов или же с помощью отношения новостного ресурса к определенной тематике. Однако на различных новостных ресурсах новостные категории разделены по разным тематикам, каждая из которых может охватывать различный от другого новостного ресурса диапазон новостей. Например, на некотором новостном ресурсе есть категория под названием «Политика», в которой публикуются некоторые новостные события, происходящие в стране и мире и относящиеся к теме политика. На некотором другом новостном ресурсе есть категория новостных документов под названием «Мир», в которой публикуются политические события и не только политические, происходящие в мире. А политические события, происходящие в стране находятся в категории «Страна». Кроме того, в категории с названием «Страна» публикуются также новости об открытиях новых месторождений природных ископаемых в стране. Данные новости относятся к тематике «Страна», но их нельзя отнести к категории политика. Моей задачей является поиск способов по некоторым ключевым словам пользователя осуществлять поиск тех свежих событий, которые могут заинтересовать его, при этом используя различные новостные ресурсы.

Идея моего поискового сервиса новостей заключается в том, что ключевые слова пользователя передаются некоторому классификатору, который при запуске программы производит загрузку уже ранее загруженного архива новостей по каждой новостной категории каждого новостного сайта. Для каждого новостного ресурса происходит обучение соответствующему ему классификатора, после чего каждый классификатор каждого новостного ресурса набор слов, введенный пользователем, сможет отнести к определенной тематике, которая наиболее точным образом соответствует ключевым словам пользователя. Таким образом, для каждого новостного ресурса будет произведен отдельно выбор категории, наиболее точно соответствующей ключевым словам. Принцип работы сервера изображен на следующей схеме Рис.1:

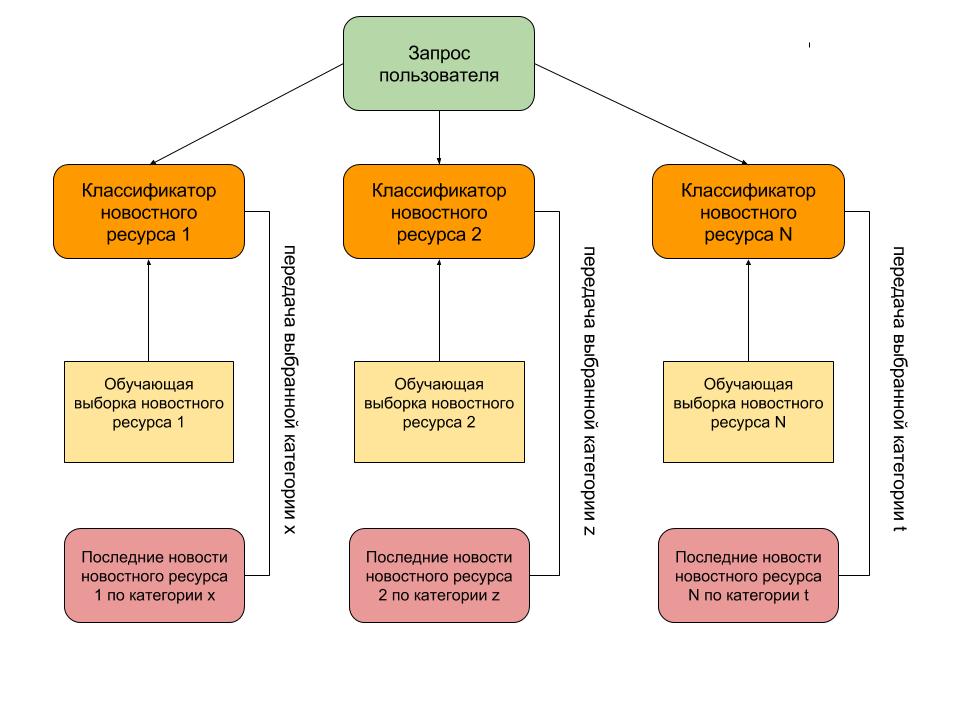


Рис.1

Однако, задача отнесения документа к наиболее вероятному классу, к которому данный документ принадлежит, имеющая решение с помощью классификатора «Наивный Байес», не является совсем той задачей, которую необходимо решить нам. Ведь те слова, которые пишет пользователь, не обязательно являются документом, который обязательно может принадлежать каждому новостному ресурсу или быть написанным авторами, работающими на данных новостных ресурсах. В связи с этим для более точного определения наиболее подходящей категории для пользователя немного изменим формулу, по которой классификатор должен определять категорию:

Таким образом, для каждого новостного ресурса будет выбираться та новостная категория, в которой наиболее часто встречаются введенные пользователем слова.

Мною был реализован подобный сервер для таких новостных ресурсов как: newsru.com и izvestia.ru Результаты:

По запросу: «Зенит, Спартак, арена» были выведены результаты Рис.2:

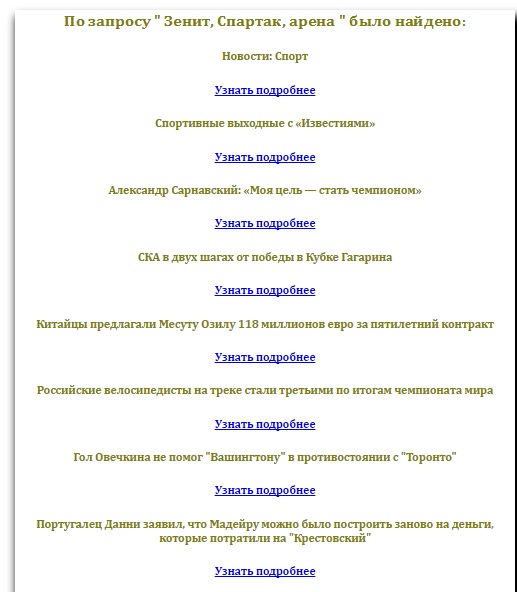


Рис.2

Из данного результата можно сделать вывод, что были выданы результаты о различных спортивных мероприятиях в нашей стране и за рубежом.

По запросу «Доллар акции нефть» были найдены следующие новости Рис.3:



Рис.3

Из результатов запроса можно сделать вывод, что были получены новости, относящиеся к финансовым новостям в стране и в мире, к которым относятся также новости об изменениях цен в стране и в мире.

Рис.4

По запросу «вооружение минобороны Фурсенко танки» было найдено:



Рис.4

Первая половина новостей относится к новостям о военных событиях в мире, в то время как большинство второй половины новостей относится к новостям, происходящим в России, где применялось различное оружие.

Таким образом, большинство новостей, выданных на сервере после запроса, скорее всего заинтересовали бы пользователя. Однако достаточно сложно определить, какие новости удачно бы соответствовали ожиданиям пользователя, а какие нет т.к. данный сервер выдает новости, которые могли бы заинтересовать пользователя исходя их тех слов и словосочетаний, которые его интересуют.

Возможно, для данного сервера следует использовать некоторые другие модификации алгоритмов классификации или же стоит для каждого сервера выбирать несколько наиболее вероятных категорий и из каждой категории для каждого сервера загружать такое количество новостей, которое было бы пропорционально некоторому вероятному процентному соотношению для данной категории.

# Выводы

После проделанной мною работы можно отметить, что были изучены основополагающие компоненты поисковых систем и некоторые принципы их взаимодействия. Были рассмотрены подробно принципы решения проблем сортировки результатов поиска, выдаваемых поисковыми системами. Безусловно, у каждой поисковые системы существуют свои способы поиска новых интернет-страниц, которые не были занесены в базу данных ранее, свои способы индексации т.е. способы занесения найденной информации в базу данных, которые поисковые системы оптимизируют для повышения эффективности работы поисковых систем, свои способы ранжирования результатов поиска. Безусловно, что один и тот же интернет ресурс при одинаковых запросах пользователей на разных поисковых системах будет находиться на разных позициях в результатах поиска. Также были рассмотрены различные алгоритмы сканирования краулера BeeBot интернет-страниц, которые могут быть использованы для различных поисков внешних ссылок на различных интернет ресурсах.

Также была рассмотрена общая структура крупнейших поисковых систем т.к. данные поисковые системы вследствие своего рейтинга заслуживают большего доверия относительно даже расположения элементов взаимодействия пользователя с серверами поисковых систем. Также были изучены и использованы формулы классификации документов с помощью «Наивный Байес» для определения для каждого новостного ресурса наиболее подходящей тематики, соответствующей запросу пользователя. Данный метод был выбран вследствие того, что это один из самых эффективных методов машинного обучения, когда необходимо найти не наиболее близкую оценку вероятности принадлежности некоторого запроса к некоторому классу документов, а когда необходимо определить наиболее вероятный класс, к которому некоторый документ. В моей задаче отнесения некоторого запроса пользователя(документа) к некоторой тематике (классу) как раз-таки требовалось определить единственную наиболее вероятную тематику для каждого новостного ресурса.

# Заключение

Таким образом, были исследованы некоторые основные подходы работы алгоритмов и их взаимодействий между собой в работе поисковых систем. Рассматривать работу алгоритмов некоторых конкретных поисковых систем достаточно проблематично, особенно поисковых систем, известных своим успехом и эффективностью своей работы т.к. точный набор подобных алгоритмов и принципы их взаимодействия крупнейших поисковых систем являются коммерческой тайной и не подлежат всеобщему распространению. При этом некоторые алгоритмы могут быть эффективными только при определенном взаимодействии друг с другом.

Также был создан небольшой сервер по поиску новостей, которые могут заинтересовать пользователя. Данный подход поиска можно использовать далее, оптимизируя результаты поиска, полученные с помощью данного сервера.

# Список литературы

1. Введение в поисковые системы

<http://works.doklad.ru/view/4yqZXQ0Pous/all.html>

1. Поисковые системы: состав, функции, принципы работы: <http://www.studfiles.ru/preview/5866029/page:20/>
2. Поисковые системы Интернет: основные принципы организации и формирование поисковых запросов

<http://e-ikt.uginfo.sfedu.ru/lectures/lecture4/lecture4_6.html>

1. Операторы поисковых систем «Google» и «Яндекс»

https://1ps.ru/blog/dirs/2016/operatoryi-poiskovyix-sistem-google-i-yandeks

1. Страница результатов поиска <https://ru.wikipedia.org/wiki/Страница_результатов_поиска>
2. Поисковая и органическая выдача систем «Яндекс» и «Google»

<http://loleknbolek.com/slovar-terminov/poiskovaya-i-organicheskaya-vydacha-sistem-yandeks-google-i-kak-ona-formiruetsya>

1. Поисковая выдача поисковых систем

<http://pgdv.ru/samostoyatelno/poiskovaya-vydacha>

1. Расширенный поиск на yandex.ru

<https://yandex.ru/search/advanced?&lr=2>

1. Расширенный поиск на google.ru

<https://www.google.ru/advanced_search?hl=ru&fg=1>

1. Состав и принципы работы поисковой системы

<http://www.web-shpargalka.ru/printsip-raboti-poiskovih-sistem.php>

1. Печников А. А. , Чернобровкин Д. И. Адаптивный краулер для поиска и сбора внешних гиперссылок  // журнал «Управление большими системами» *УБС*,2012, **36**, с 301–315
2. Никитенко Е,В., Р.В. Заровский, М.А. Сдитанов Автоматическая система поиска информации в текстах с использованием алгоритмов нечеткой логики // Технические науки и технологии. Черниговский национальный технологический университет, 2012, №4(61), с 150-157
3. Яковлев С.А. Корпоративные информационные системы ч.1. 2010. 117 с.
4. Основные подходы алгоритмов поисковых систем

<http://blog.aweb.ua/osnovnye-algoritmy-poiskovoj-sistemy/>

1. . М.А.Демихов Методы нечеткого поиска в информационных системах // научный журнал «Моделирование, оптимизация и информационные технологии» (МОИТ). Воронежский институт высоких технололгий 2015 №2(9)
2. Введение в поисковые системы. “Современные задачи теоретической информатики”

<http://logic.pdmi.ras.ru/~yura/modern/05.pdf>

1. PageRank в поисковой системе «Google»

<http://wm-help.net/articles/article/25.05.20061167-14.html>

1. Модели информационного поиска. PageRank «Алгоритмы для Интернета»

http://yury.name/internet/03ianote.pdf

1. Параллельный поиск в «Яндекс»

<http://tekseo.su/poiskovye-sistemy/koldunshhik-yandeks-kak-rabotaet-parallelnyjj-poisk.php>

1. Александр Сизов, Сергей Николенко Наивный Байесовский классификатор.

http://logic.pdmi.ras.ru/~sergey/teaching/mlstc12/sem01-naivebayes.pdf

1. Метод классификации «Наивный Байес»

<http://bazhenov.me/blog/2012/06/11/naive-bayes>

# Приложение

(Полностью программный код вместе с текстовыми данными, необходимыми для работы сервера, можно найти в размещенной мною репозитории, находящейся по ссылке: https://github.com/Valera14/Search\_news\_server)

# coding: utf-8

# In[ ]:

import re

import numpy as np

from bottle import route, request, run

import lxml.html

import datetime

import requests

import lxml.etree

#Известия

categories = list()

categories.append('15')

categories.append('19')

categories.append('16')

categories.append('28')

categories.append('17')

categories.append('24')

categories.append('25')

categories.append('18')

categories.append('21')

categories.append('87')

categories.append('97')

today = datetime.date.today()

year\_current = today.year

month\_current = today.month

day\_current = today.day

for category in categories:

f\_category = open('izvestia\_' + category + '.txt', 'r', encoding='utf-8')

category\_t = f\_category.read()

result = re.findall(r'\w\w\w+', category\_t)

all\_words\_counter = np.arange(len(result))

rating\_list = np.zeros(len(result))

final\_list\_words = list()

for counter\_all in all\_words\_counter:

word = result[counter\_all]

list\_counter = len(final\_list\_words)

list\_counter = np.arange(list\_counter)

i = 0

for counter in list\_counter:

if (final\_list\_words[counter] == word):

rating\_list[counter] = rating\_list[counter] + 1

i = i+1

if (i==0):

final\_list\_words.append(word)

if (category == '15'):

final\_words\_15 = final\_list\_words

rating\_list\_15 = rating\_list

print(category + 'completed')

elif (category == '19'):

final\_words\_19 = final\_list\_words

rating\_list\_19 = rating\_list

print(category + 'completed')

elif (category == '16'):

final\_words\_16 = final\_list\_words

rating\_list\_16 = rating\_list

print(category + 'completed')

elif (category == '28'):

final\_words\_28 = final\_list\_words

rating\_list\_28 = rating\_list

print(category + 'completed')

elif (category == '17'):

final\_words\_17 = final\_list\_words

rating\_list\_17 = rating\_list

print(category + 'completed')

elif (category == '24'):

final\_words\_24 = final\_list\_words

rating\_list\_24 = rating\_list

print(category + 'completed')

elif (category == '25'):

final\_words\_25 = final\_list\_words

rating\_list\_25 = rating\_list

print(category + 'completed')

elif (category == '18'):

final\_words\_18 = final\_list\_words

rating\_list\_18 = rating\_list

print(category + 'completed')

elif (category == '21'):

final\_words\_21 = final\_list\_words

rating\_list\_21 = rating\_list

print(category + 'completed')

elif (category == '87'):

final\_words\_87 = final\_list\_words

rating\_list\_87 = rating\_list

print(category + 'completed')

elif (category == '97'):

final\_words\_97 = final\_list\_words

rating\_list\_97 = rating\_list

print(category + 'completed')

#newsru.com

categories\_newsru = list()

categories\_newsru.append('russia')

categories\_newsru.append('world')

categories\_newsru.append('finance')

categories\_newsru.append('religy')

categories\_newsru.append('crime')

categories\_newsru.append('sport')

categories\_newsru.append('cinema')

for category in categories\_newsru:

f\_category = open('newsru.com\_' + category + '.txt', 'r', encoding='utf-8')

category\_t = f\_category.read()

result = re.findall(r'\w\w\w+', category\_t)

all\_words\_counter = np.arange(len(result))

rating\_list = np.zeros(len(result))

final\_list\_words = list()

for counter\_all in all\_words\_counter:

word = result[counter\_all]

list\_counter = len(final\_list\_words)

list\_counter = np.arange(list\_counter)

i = 0

for counter in list\_counter:

if (final\_list\_words[counter] == word):

rating\_list[counter] = rating\_list[counter] + 1

i = i+1

if (i==0):

final\_list\_words.append(word)

if (category == 'russia'):

final\_words\_russia = final\_list\_words

rating\_list\_russia = rating\_list

print(category + 'completed')

elif (category == 'world'):

final\_words\_world = final\_list\_words

rating\_list\_world = rating\_list

print(category + 'completed')

elif (category == 'finance'):

final\_words\_finance = final\_list\_words

rating\_list\_finance = rating\_list

print(category + 'completed')

elif (category == 'religy'):

final\_words\_religy = final\_list\_words

rating\_list\_religy = rating\_list

print(category + 'completed')

elif (category == 'crime'):

final\_words\_crime = final\_list\_words

rating\_list\_crime = rating\_list

print(category + 'completed')

elif (category == 'sport'):

final\_words\_sport = final\_list\_words

rating\_list\_sport = rating\_list

print(category + 'completed')

elif (category == 'cinema'):

final\_words\_cinema = final\_list\_words

rating\_list\_cinema = rating\_list

print(category + 'completed')

@route('/')

def login():

return '''

<head>

<meta name="viewport" content="width=device-width" />

<title>Поиск интересующих новостей</title>

<style type="text/css">

input{

box-shadow:8px 13px 13px 7px black;

color:olive;

align-content:center;

font-family:Cambria;

width:500px;

height:100px;

margin-left:120px;

margin-top:75px;

text-align:center;

font-size:30px;

}

form{

font-size:30px;

color:teal;

margin-left:120px;

margin-top:75px;

}

</style>

</head>

<body>

<form action="/login" method="post">

<p style="padding-left:200px; padding-top:100px;"> Введите ключевые слова</p>

<input name="our\_sentence" type="text" />

<p><input value="Начать поиск" type="submit" /></p>

</form>

</body>

'''

@route('/login', method='POST')

def do\_login():

user\_request = (request.forms.get('our\_sentence')).encode('latin1').decode('utf8')

words\_news = re.findall(r'\w\w\w+', user\_request)

#Известия

category\_point\_15, category\_point\_19, category\_point\_16 = 0, 0, 0

category\_point\_28, category\_point\_17, category\_point\_24 = 0, 0, 0

category\_point\_25, category\_point\_18, category\_point\_21 = 0, 0, 0

category\_point\_87, category\_point\_97 = 0, 0

#newsru.com

category\_point\_russia, category\_point\_world, category\_point\_finance = 0, 0, 0

category\_point\_religy, category\_point\_crime, category\_point\_sport = 0, 0, 0

category\_point\_cinema = 0

#Известия

len\_15 = len(final\_words\_15)

len\_15\_all = np.arange(len\_15)

print(len\_15)

len\_19 = len(final\_words\_19)

len\_19\_all = np.arange(len\_19)

print(len\_19)

len\_16 = len(final\_words\_16)

len\_16\_all = np.arange(len\_16)

print(len\_16)

len\_28 = len(final\_words\_28)

len\_28\_all = np.arange(len\_28)

print(len\_28)

len\_17 = len(final\_words\_17)

len\_17\_all = np.arange(len\_17)

len\_24 = len(final\_words\_24)

len\_24\_all = np.arange(len\_24)

len\_25 = len(final\_words\_25)

len\_25\_all = np.arange(len\_25)

len\_18 = len(final\_words\_18)

len\_18\_all = np.arange(len\_18)

len\_21 = len(final\_words\_21)

len\_21\_all = np.arange(len\_21)

len\_87 = len(final\_words\_87)

len\_87\_all = np.arange(len\_87)

len\_97 = len(final\_words\_97)

len\_97\_all = np.arange(len\_97)

print(len\_17)

print(len\_24)

print(len\_25)

print(len\_18)

print(len\_21)

print(len\_87)

print(len\_97)

#newru.com

len\_russia = len(final\_words\_russia)

len\_russia\_all = np.arange(len\_russia)

print(len\_russia)

len\_world = len(final\_words\_world)

len\_world\_all = np.arange(len\_world)

print(len\_world)

len\_finance = len(final\_words\_finance)

len\_finance\_all = np.arange(len\_finance)

print(len\_finance)

len\_religy = len(final\_words\_religy)

len\_religy\_all = np.arange(len\_religy)

print(len\_religy)

len\_crime = len(final\_words\_crime)

len\_crime\_all = np.arange(len\_crime)

print(len\_crime)

len\_sport = len(final\_words\_sport)

len\_sport\_all = np.arange(len\_sport)

print(len\_sport)

len\_cinema = len(final\_words\_cinema)

len\_cinema\_all = np.arange(len\_cinema)

print(len\_cinema)

#Известия

for word in words\_news:

for counter\_15\_category in len\_15\_all:

if (word == final\_words\_15[counter\_15\_category]):

category\_point\_15 = category\_point\_15 + rating\_list\_15[counter\_15\_category]

for counter\_19\_category in len\_19\_all:

if (word == final\_words\_19[counter\_19\_category]):

category\_point\_19 = category\_point\_19 + rating\_list\_19[counter\_19\_category]

for counter\_16\_category in len\_16\_all:

if (word == final\_words\_16[counter\_16\_category]):

category\_point\_16 = category\_point\_16 + rating\_list\_16[counter\_16\_category]

for counter\_28\_category in len\_28\_all:

if (word == final\_words\_28[counter\_28\_category]):

category\_point\_28 = category\_point\_28 + rating\_list\_28[counter\_28\_category]

for counter\_17\_category in len\_17\_all:

if (word == final\_words\_17[counter\_17\_category]):

category\_point\_17 = category\_point\_17 + rating\_list\_17[counter\_17\_category]

for counter\_24\_category in len\_24\_all:

if (word == final\_words\_24[counter\_24\_category]):

category\_point\_24 = category\_point\_24 + rating\_list\_24[counter\_24\_category]

for counter\_25\_category in len\_25\_all:

if (word == final\_words\_25[counter\_25\_category]):

category\_point\_25 = category\_point\_25 + rating\_list\_25[counter\_25\_category]

for counter\_18\_category in len\_18\_all:

if (word == final\_words\_18[counter\_18\_category]):

category\_point\_18 = category\_point\_18 + rating\_list\_18[counter\_18\_category]

for counter\_21\_category in len\_21\_all:

if (word == final\_words\_21[counter\_21\_category]):

category\_point\_21 = category\_point\_21 + rating\_list\_21[counter\_21\_category]

for counter\_87\_category in len\_87\_all:

if (word == final\_words\_87[counter\_87\_category]):

category\_point\_87 = category\_point\_87 + rating\_list\_87[counter\_87\_category]

for counter\_97\_category in len\_97\_all:

if (word == final\_words\_97[counter\_97\_category]):

category\_point\_97 = category\_point\_97 + rating\_list\_97[counter\_97\_category]

#newsru.com

for word in words\_news:

for counter\_russia\_category in len\_russia\_all:

if (word == final\_words\_russia[counter\_russia\_category]):

category\_point\_russia = category\_point\_russia + rating\_list\_russia[counter\_russia\_category]

for counter\_world\_category in len\_world\_all:

if (word == final\_words\_world[counter\_world\_category]):

category\_point\_world = category\_point\_world + rating\_list\_world[counter\_world\_category]

for counter\_finance\_category in len\_finance\_all:

if (word == final\_words\_finance[counter\_finance\_category]):

category\_point\_finance = category\_point\_finance + rating\_list\_finance[counter\_finance\_category]

for counter\_religy\_category in len\_religy\_all:

if (word == final\_words\_religy[counter\_religy\_category]):

category\_point\_religy = category\_point\_religy + rating\_list\_religy[counter\_religy\_category]

for counter\_crime\_category in len\_crime\_all:

if (word == final\_words\_crime[counter\_crime\_category]):

category\_point\_crime = category\_point\_crime + rating\_list\_crime[counter\_crime\_category]

for counter\_sport\_category in len\_sport\_all:

if (word == final\_words\_sport[counter\_sport\_category]):

category\_point\_sport = category\_point\_sport + rating\_list\_sport[counter\_sport\_category]

for counter\_cinema\_category in len\_cinema\_all:

if (word == final\_words\_cinema[counter\_cinema\_category]):

category\_point\_cinema = category\_point\_cinema + rating\_list\_cinema[counter\_cinema\_category]

#Известия

rating\_list\_final = list()

rating\_list\_final.append(category\_point\_15/len\_15)

rating\_list\_final.append(category\_point\_19/len\_19)

rating\_list\_final.append(category\_point\_16/len\_16)

rating\_list\_final.append(category\_point\_28/len\_28)

rating\_list\_final.append(category\_point\_17/len\_17)

rating\_list\_final.append(category\_point\_24/len\_24)

rating\_list\_final.append(category\_point\_25/len\_25)

rating\_list\_final.append(category\_point\_18/len\_18)

rating\_list\_final.append(category\_point\_21/len\_21)

rating\_list\_final.append(category\_point\_87/len\_87)

rating\_list\_final.append(category\_point\_97/len\_97)

selected\_category = np.argmax(rating\_list\_final)

category = str(categories[selected\_category])

selected\_html = 'http://izvestia.ru/rubric/' + category

response = requests.get(selected\_html)

tree = lxml.html.fromstring(response.text)

f13 = tree.xpath('//h2/a[@href]/text()')

f5 = tree.xpath('//h2/a[@href]/@href')

len\_pages = len(f5)

len\_pages5 = np.arange(len\_pages)

for page in len\_pages5:

f5[page] = 'http://izvestia.ru' + str(f5[page])

#newsru

rating\_list\_final\_newsru = list()

rating\_list\_final\_newsru.append(category\_point\_russia/len\_russia)

rating\_list\_final\_newsru.append(category\_point\_world/len\_world)

rating\_list\_final\_newsru.append(category\_point\_finance/len\_finance)

rating\_list\_final\_newsru.append(category\_point\_religy/len\_religy)

rating\_list\_final\_newsru.append(category\_point\_crime/len\_crime)

rating\_list\_final\_newsru.append(category\_point\_sport/len\_sport)

rating\_list\_final\_newsru.append(category\_point\_cinema/len\_cinema)

selected\_category = np.argmax(rating\_list\_final\_newsru)

category = str(categories\_newsru[selected\_category])

selected\_html = 'http://www.newsru.com/' + category

response = requests.get(selected\_html)

tree = lxml.html.fromstring(response.text)

f13\_newsru = tree.xpath('//div/a[@class="index-news-title"]/text()')

f5\_newsru = tree.xpath('//div/a[@class="index-news-title"]/@href')

len\_pages = len(f5\_newsru)

len\_pages5 = np.arange(len\_pages)

for page in len\_pages5:

f5\_newsru[page] = 'http://www.newsru.com' + str(f5\_newsru[page])

user\_request = (request.forms.get('our\_sentence')).encode('latin1').decode('utf8')

return '''

<head>

<meta name="viewport" content="width=device-width" />

<title>Поиск интересующих новостей</title>

<style type="text/css">

div{

box-shadow:8px 13px 13px 7px black;

color:olive;

align-content:center;

font-family:Cambria;

width:500px;

height:1000px;

margin-left:120px;

margin-top:75px;

text-align:center;

font-size:14px;

}

</style>

</head>

<body>

<div>

<h3> По запросу "

''' + user\_request + '''

" было найдено:

</h3>

<h5>

''' + str(f13[0]) + '''

</h5>

<h5><a href="''' + str(f5[0]) + '''">Узнать подробнее</a>

</h5>

<h5>

''' + str(f13[1]) + '''

</h5>

<h5><a href="''' + str(f5[1]) + '''">Узнать подробнее</a>

</h5>

<h5>

''' + str(f13[2]) + '''

</h5>

<h5><a href="''' + str(f5[2]) + '''">Узнать подробнее</a>

</h5>

<h5>

''' + str(f13[3]) + '''

</h5>

<h5><a href="''' + str(f5[3]) + '''">Узнать подробнее</a>

</h5>

<h5>

''' + str(f13\_newsru[0]) + '''

</h5>

<h5><a href="''' + str(f5\_newsru[0]) + '''">Узнать подробнее</a>

</h5>

<h5>

''' + str(f13\_newsru[1]) + '''

</h5>

<h5><a href="''' + str(f5\_newsru[1]) + '''">Узнать подробнее</a>

</h5>

<h5>

''' + str(f13\_newsru[2]) + '''

</h5>

<h5><a href="''' + str(f5\_newsru[2]) + '''">Узнать подробнее</a>

</h5>

<h5>

''' + str(f13\_newsru[3]) + '''

</h5>

<h5><a href="''' + str(f5\_newsru[3]) + '''">Узнать подробнее</a>

</h5>

</div>

</body>

'''

# In[ ]:

run(host='localhost', port=5000)