Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет Компьютерных сетей и систем

Обработка больших объемов информации

РЕФЕРАТ

на тему «Автоматизированное извлечение знаний из гипертекста»

Студент
гр. 758601
Лимонтов А. С.

Проверил
Захаров В. В.

1.Проблемы понимания текста на естественном языке

Автоматизированное извлечение знаний из текста становится одной из центральных задач искусственного интеллекта. Этому способствует исключительно быстрое развитие Internet и электронных библиотек, в которых знания представляются, в основном, в текстовом виде. Создание методов автоматизированного извлечения знаний из текста сопряжено с фундаментальной проблемой искусственного интеллекта, связанной с пониманием текста на естественном языке.

1.1.Схема анализа текста на естественном языке

Общепризнанная схема анализа монологического текста на естественном языке изображена на рисунке:

Предредактор выделяет в исходном тексте слова и фразы и проверяет выполнение принятых ограничений. Обычно недопустимыми являются сложноподчиненные предложения,

включающие рекурсивно вложенные определительные предложения.

Блок морфологического анализа выделяет в словах неизменные части (основы) и приписывает словам ряд грамматических характеристик (часть речи, род, число, падеж, склонение, вид и т. п.).

Программная реализация предредактора и блока морфологического анализа не вызывает трудностей за исключением отмеченных выше ограничений для предредакторов и немногих случаев морфологической омонимии. Последняя проблема разрешается в блоке синтаксического анализа. Он строит дерево синтаксического разбора, используя базу синтаксических правил.

Цель семантического анализа состоит в определении для каждого слова и фразы в целом некоторых смысловых характеристик. Сложности возникают из-за семантической неоднозначности. Для ее снятия используются тезаурусные статьи, связанные друг с другом в рамках семантической сети. Анализ отношений в ней позволяет получить информацию, в явном виде отсутствующую во фразе, но без которой адекватное понимание фразы невозможно. Трудности реализации этого этапа связаны с большим размером требуемых семантических сетей и многовариантностью анализа.

После семантического анализа выполняется перевод анализируемого текста во внутреннее представление. Обычно для этого также используются семантические сети. Содержание текста на естественном языке отображается во фрагмент семантической сети, связанный дугами соответствующего типа с той семантической сетью, которая уже хранится в системе. Воплощение этого этапа не вызывает трудностей.

Внутреннее представление служит основой для реализации феномена понимания естественно-языкового текста. Именно с этим процессом связаны основные теоретические проблемы. Во многом они обусловлены отсутствием точного определения термина «понимание».

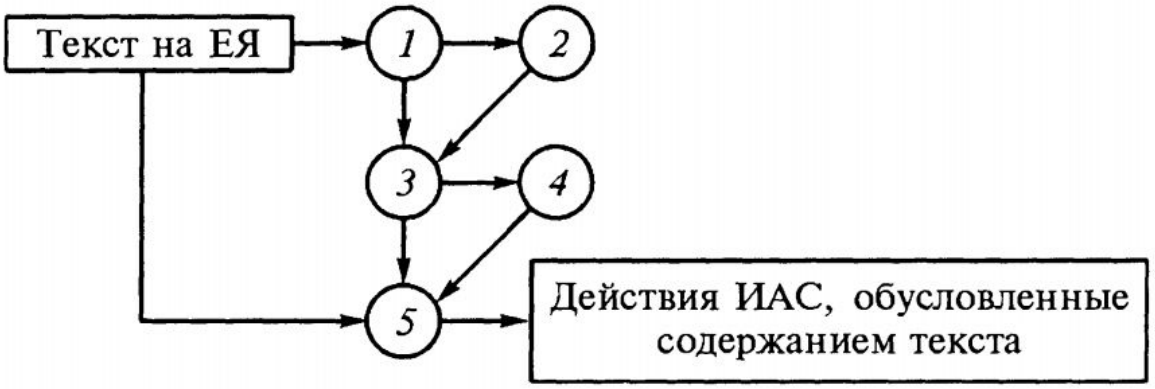
1.2.Понимание текста

Понимание — многоуровневый процесс. На первом, простейшем уровне все сведения о содержании рассматриваемого текста извлекаются в результате его анализа без привлечения дополнительных знаний, известных системе. На втором уровне с помощью процедур логического пополнения информации осуществляется доопределение временной, пространственной и причинно-следственной структур событий.

На третьем уровне к сформированному представлению содержания текста добавляется информация, релевантная этому содержанию и известная системе. На четвертом уровне к нему присоединяются сведения, извлеченные из базы знаний и связанные с анализируемым текстом только отношениями ассоциации.

На пятом уровне понимания из анализируемого текста извлекается его прагматическое содержание. При этом система выполняет все обусловленные им действия, например, решает задачу, для которой есть готовая или генерируемая программа, а в исходном тексте выражены исходные данные для нее. Ясно, что наибольший практический интерес представляют системы, реализующие пятый уровень понимания, и именно они называются интеллектуальными. Вместо модели «текст — смысл — текст» такие системы реализуют модель «текст — действительность — текст».

Взаимосвязь уровней понимания естественно-языковых текстов показана на следующей схеме. Уровни не образуют строгой иерархии, и порядок их прохождения может быть разным.



До реализации в полном объеме такой схемы понимания еще далеко.

2.Компьютерные методы поиска в тексте

2.1.Актуальность проблемы

Проблема поиска и сбора информации является одной из важнейших задач информатики. Компьютерные методы информационного поиска – активно развивающаяся, актуальная в научном и практическом аспекте тема современных публикаций. Развитие компьютерной техники влечет существенный рост объема информации, представляемой в электронном виде, влияние этого процесса на современные информационные технологии, включая поиск, отмечается в большинстве публикаций в периодических изданиях. Приобрели актуальность вопросы, связанные с проблемой поиска информации на электронных носителях, функция поиска значительно упрощает пользователям навигацию в неограниченном множестве массивов документов, хранящихся на Web-серверах, включая электронные библиотеки. В частности, функция поиска данных строкового типа существенно облегчает редактирование документов и поиск требуемой информации.

В настоящее время функция поиска является составляющей многих программных продуктов, редакторов языков программирования. Актуальны задачи поиска объектов, хранящихся в оцифрованном виде по нескольким признакам одновременно, однако поиск изображений сводится к поиску в тексте по названию

изображения. Отметим, что с этой целью можно предложить поиск изображений по фрагментом без текстовой надписи.

Можно выделить следующие классические разновидности поиска:

- поиск в массиве записей,
- поиск подстроки в строке или поиск по образцу,
- алгоритмы информационного поиска,
- алгоритмы поисковой системы.

2.2. Классические схемы поиска в массиве записей

В классических методах поиск рассматривается как нахождение данных, удовлетворяющих определенному свойству. В некоторых алгоритмах предполагается, что информация содержится в записях, которые представляют собой массив данных в программе, и каждая запись содержит поле, которое называется ключом. Записи идут в массиве последовательно, номера записей в списке идут от 1 до N — полного числа записей. Массив записей может быть неотсортированным или отсортированным по значению ключевого поля. В неотсортированном массиве порядок записей случаен, в отсортированном они идут в порядке возрастания ключа. При этом известные схемы поиска реализуют поиск только однотипных данных, в этих схемах поиск ведется только по одному ключу (признаку поиска).

Существует следующая классификация методов поиска:

1. Внутренний и внешний поиск с разделением используемых сортировок.
2. Статические и динамические методы поиска. При статическом поиске массив значений не меняется во время работы алгоритма. Во время динамического поиска массив может перестраиваться или изменять размерность.
3. Методы, основанные на сравнении ключей и на цифровых свойствах ключей.
4. Методы, использующие истинные ключи и преобразованные ключи.

Простейший поиск записи в неотсортированном массиве сводится к просмотру всего списка до того, как запись будет найдена. Этот алгоритм не всегда эффективен, однако работает на произвольном списке.

2.2.1. Линейный поиск

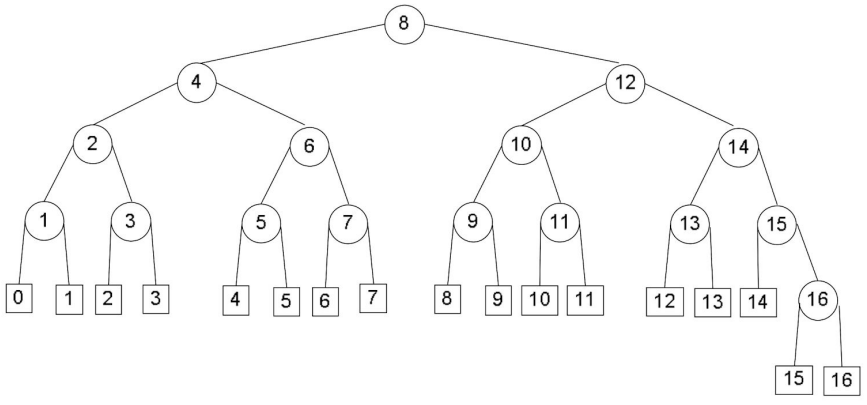
Линейный поиск заключается в простом последовательном просмотре всех элементов массива и сравнении с эталоном (ключом) X. Эта процедура выдает либо значение индекса для найденного элемента массива, либо нулевое значение, когда требуемый элемент не найден. При прямом последовательном поиске в среднем проверяются $N/2$ элементов. В лучшем случае будет проверяться один элемент, в худшем – N элементов. Если данные не отсортированы, то линейный поиск является единственно возможным.

Классические способы поиска в отсортированном массиве:

- бинарный поиск,
- поиск по “дереву Фибоначчи”,
- интерполяционный поиск.

2.2.2. Бинарный поиск

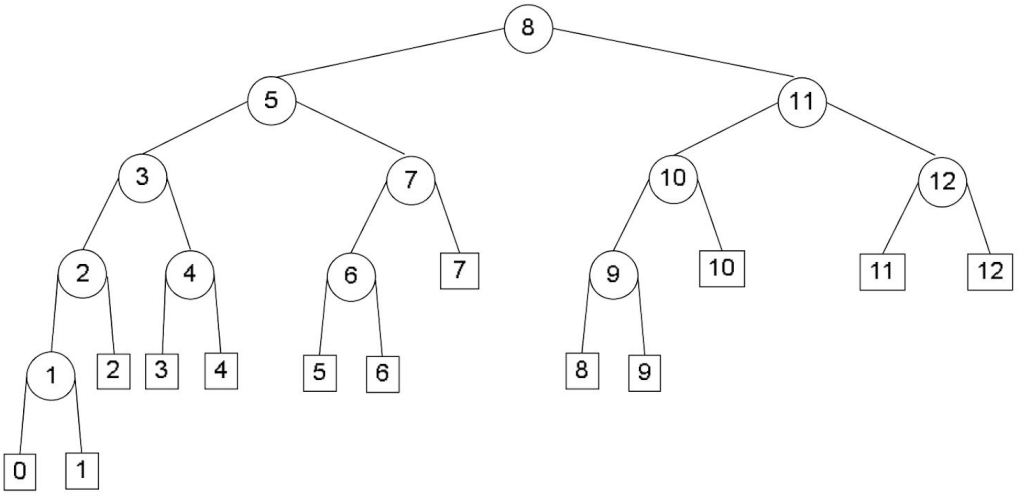
В процессе бинарного поиска сравнивается эталон (ключ) с элементом в середине массива и в зависимости от результата сравнения (больше или меньше) дальнейший поиск проводится в левой или в правой половине массива. Самый простой визуальный пример бинарного поиска (здесь узлы представлены ключами, $N = 16$):



Асимптотическая оценка времени поиска $O(logN)$.

2.2.3. Поиск по “дереву Фибоначчи”

В дереве Фибоначчи числа в дочерних узлах отличаются от числа в родительском узле на одну и ту же величину, а именно на число Фибоначчи. Суть метода в том, что в ходе сравнения искомого значения с очередным значением в массиве новая зона поиска не делится пополам, как в бинарном поиске, а происходит смещение от предыдущего значения, с которым сравнивали, в нужную сторону на число Фибоначчи. На рисунке представлен пример дерева Фибоначчи с порядком 6:



Этот способ считается более эффективным, чем предыдущий, потому что метод Фибоначчи включает в себя только такие арифметические операции, как сложение и вычитание: нет необходимости в делении, тем самым экономится процессорное время.

2.2.4. Интерполяционный поиск

Если известно, что K лежит между K_L и K_R, то следующую пробу делаем на расстоянии $\frac{(R-L)(K-K_L)}{(K_R-K_L)}$ от L, предполагая, что ключи являются числами, возрастающими приблизительно в арифметической прогрессии. Интерполяционный поиск асимптотически предпочтительнее бинарного, так как один шаг

бинарного поиска уменьшает количество записей, среди которых находится искомая, с N до $N/2$, а один шаг интерполяционного поиска с N до $\sqrt{N}$. Интерполяционный поиск требует в среднем около $log_2 log_2 N$ шагов. Скорость интерполяционного метода начинает существенно превышать скорость метода половинного деления при больших значениях N.

Кроме перечисленных можно также назвать метод цифрового поиска, который вместо непосредственного сравнения ключей использует их представление в виде последовательности цифр и букв – поиск по дереву. Еще одна группа методов поиска позволяет произвести над ключом K арифметические вычисления и получить функцию $f(K)$, указывающую адрес в таблице, где хранится K и ассоциированная с ним информация. В идеале для задач поиска хеш-адрес должен быть уникальным, чтобы за одно обращение получить доступ к элементу, характеризуемому заданным ключом.

2.3. Классические схемы поиска подстроки в строках

Среди известных методов поиска подстроки можно выделить те методы, когда ключ является составным объектом, например, массив символов, называемый строкой или словом. Для того чтобы установить факт вхождения подстроки в строку, необходимо убедиться, что все символы сравниваемых строк соответственно равны один другому. Поиск подстроки в тексте – важный элемент текстовых редакторов.

2.3.1. Алгоритм последовательного поиска в тексте

Идея метода заключается в следующем: проверяется, совпадают ли M символов текста (начиная с выбранного) с символами строки. Стандартный алгоритм начинается со сравнения первого символа текста с первым символом подстроки. Если они совпадают, то происходит переход ко второму символу текста и подстроки. При совпадении сравниваются следующие символы. Так продолжается до тех пор, пока не окажется, что подстрока целиком совпала с отрезком текста, или пока не встретились несовпадающие символы. В первом случае задача решена, во

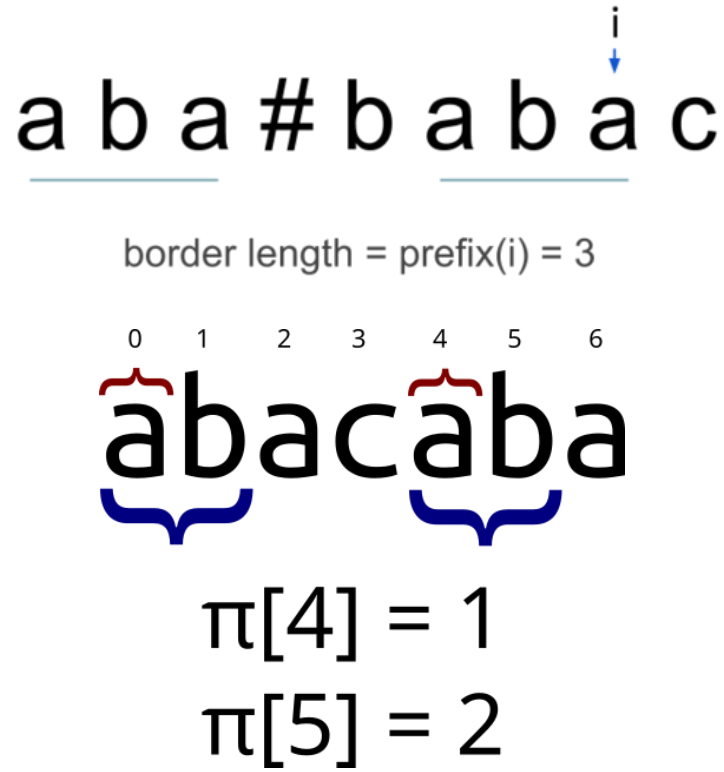
втором – указатель текущего положения в тексте перемещается на один символ и сравнение начинается заново. Время работы алгоритма оценивается как $O((N-M+1)M)$.

2.3.2. Алгоритм Рабина-Карпа

В алгоритме предлагается поставить в соответствие каждой строке некоторое уникальное число и вместо сравнения строк сравнивать числа. Недостаток метода в том, что искомая строка может быть длинной и строк в тексте может быть много. Так как в каждой строке нужно сопоставить уникальное число, то чисел должно быть много, числа будут большими и работать с ними будет неудобно. Проблема больших чисел может быть решена, если производить все арифметические действия по модулю какого-то простого числа (брать остаток от деления на это число). В этом случае находится не число, характеризующее строку, а его остаток от деления на простое число. Число ставится в соответствие не одной строке, а целому классу, но так как классов много (столько, сколько различных остатков от деления на это простое число), то дополнительная проверка производится редко. Время работы $O(M+N+\frac{MN}{P})$, где P - выбранное простое число, так что сложность алгоритма почти линейная.

2.3.3. Алгоритм Кнута-Морриса-Пратта

КМП основывается на том, что после частичного совпадения начальной части образа с соответствующими символами образ сдвигается на все пройденное расстояние, так как меньший сдвиг не может привести к полному совпадению. Центральным элементом работы этого алгоритма является понятие префикс-функции. Она определена для каждого индекса исходного текста и говорит о длине максимального собственного суффикса, совпадающего с собственным префиксом. Например:



Этот алгоритм хорош тем, что легко реализуется, а также быстро работает (за $O(N+M)$).

2.3.4. Алгоритм Бойера-Мура

Алгоритм Бойера-Мура считается более быстрым среди алгоритмов, предназначенных для поиска подстроки в строке. Отличием алгоритма БМ от алгоритма КМП является то, что для поиска выполняется сравнение символов с конца образа, а не с начала. На первом шаге строится таблица смещений для искомого образца. Совмещая начало строки и образца, выполняется проверка с последнего символа образца. Если последний символ образца и соответствующий ему при наложении символ строки не совпадают, образец сдвигается относительно строки на величину, полученную из таблицы смещений, и снова проводится сравнение, начиная с последнего символа образца. Если же символы совпадают, производится сравнение предпоследнего символа образца и т. д. Если все символы образца совпали с наложенными символами строки, значит, найдена подстрока и поиск окончен. Если же какой-то (не последний) символ образца не совпадает с соответствующим символом строки, то сдвигается образец на один

символ вправо и снова начинается проверка с последнего символа. Весь алгоритм выполняется до тех пор, пока либо не будет найдено вхождение искомого образца, либо не будет достигнут конец строки. Величина сдвига в случае несовпадения последнего символа вычисляется следующим образом: сдвиг образца должен быть минимальным, таким, чтобы не пропустить вхождение образца в строке. Если данный символ строки встречается в образце, то образец смещается таким образом, чтобы символ строки совпал с самым правым вхождением этого символа в образце. Если образец вообще не содержит этого символа, то образец сдвигается на величину, равную его длине, так что первый символ образца накладывается на следующий за проверявшимся символом строки. Величина смещения для каждого символа образца зависит только от порядка символов в образце, поэтому смещения удобно вычислить заранее и хранить в виде одномерного массива, где каждому символу алфавита соответствует смещение относительно последнего символа образца.

Число сравнений и асимптотика работы алгоритма равна $O(N + rM)$, где r - количество вхождений.

2.4. Алгоритмы информационного поиска

Информационный поиск текстов – одна из самых востребованных задач обработки текстов. Центральная проблема – помочь пользователю найти ту информацию, в которой он заинтересован. Задача поиска состоит в определении по запросу пользователя множества текстов из некоторой фиксированной базы, релевантных запросу. Запрос представляется набором ключевых слов. Основным вопросом при этом является определение релевантностей текстов запросу и сортировки документов по этим значениям. Классическая задача информационного поиска, с которой началось развитие этой области, – это поиск документов, удовлетворяющих запросу в рамках некоторой статической (на момент выполнения поиска) коллекции документов. Эта задача решается в рамках большинства современных справочных систем, включая Windows. Модели

информационного поиска делятся на три класса:

1. Теоретико-множественные модели, опирающиеся на аппарат теории множеств. Классический пример – булева модель. В рамках этой модели документы и запросы представляются в виде множеств термов.
2. Вероятностные модели, опирающиеся на теорию вероятностей. В качестве оценки релевантности документа запросу пользователя используется вероятность того, что пользователь признает документ истинно релевантным.
3. Алгебраические модели, в которых документы и запросы описываются в виде векторов в многомерном пространстве; аппарат опирается на алгебраические методы, которые широко применяются в современных информационно-поисковых системах.

2.4.1.Булева модель

Булев поиск опирается на использование инвертированного индекса ключевых слов, то есть таблицы, в которой для каждого ключевого слова перечисляются все документы, где оно встречается:

Terms	Docs
Book	Doc1 , Doc3 , Doc4
Teach	Doc4 , Doc61 , Doc103
.	.
.	.
.	.

Главным достоинством этого алгоритма является возможность связывания слов запроса логическими операциями и получения в

результате объединения множеств документов, содержащих искомые слова. К недостаткам следует отнести невозможность определения релевантности запросу полученной выборки документов. При поиске из инвертированного индекса извлекаются списки документов, соответствующие каждому слову запроса. Над полученными множествами проводятся операции, соответствующие логическим операциям, связывающим слова запроса, в результате чего образуется список найденных документов. Как правило, данный алгоритм используется совместно с другими алгоритмами.

2.4.2. Вероятностная модель

Данная модель основана на теории вероятности и использует статистические показатели, характеризующие вероятность соответствия проиндексированных текстовых ресурсов запросу пользователя. Преимущество в том, что модель располагает документы в порядке убывания “вероятности оказаться релевантным”. На практике эти модели не получили большого распространения. В рамках моделей вычисляется условная вероятность события, что документ соответствует данному запросу, то есть $P(d \mid q)$. Для расчета используется формула Байеса и то, что вероятность $P(q)$ постоянна на протяжении всего поиска. Таким образом, $P(d \mid q) = \alpha P(d)P(q \mid d),\ \alpha = const$. В качестве факторов, влияющих на безусловную релевантность документа $P(d)$, можно рассматривать его размер, источник, дату публикации. Вероятность запроса q при условии релевантности документа d зависит главным образом от веса ключевых слов запроса в документе d. Для ее расчета обычно принимают гипотезу независимости слов документа и запроса, что приводит к следующей формуле релевантностей:

$R(d \mid q) = logP(d) + \sum_k logP(w_k \mid d)$, где $P(w_k \mid d)$ – вероятность появления k-го слова запроса в документе d.

2.4.3. Векторная модель

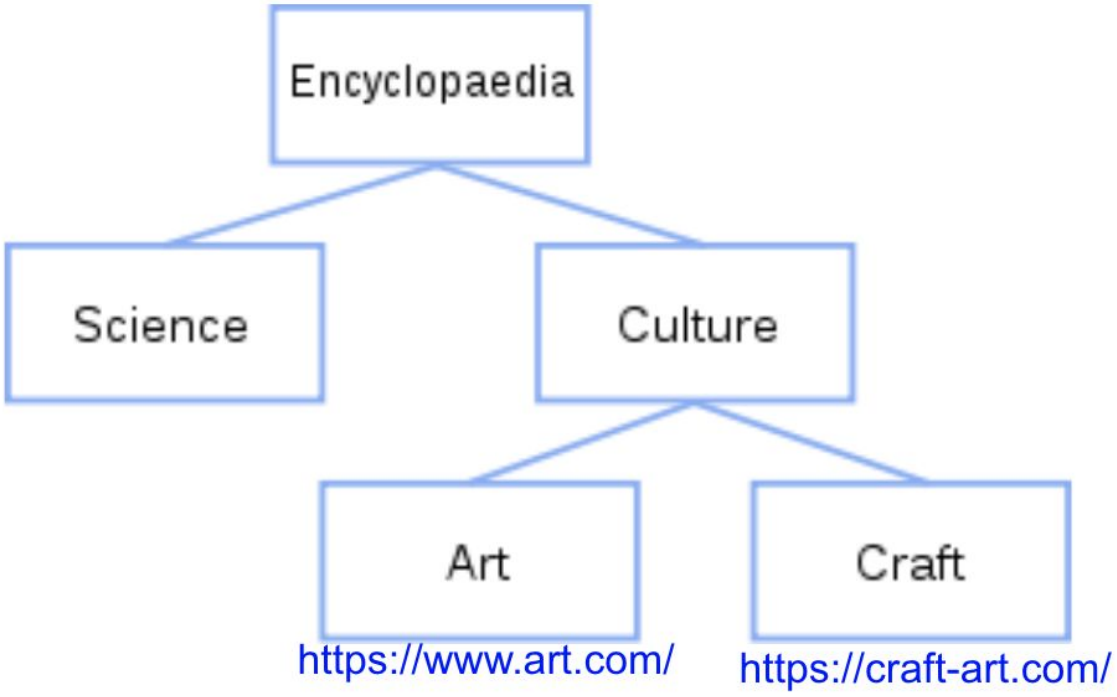
Векторная модель является классическим представителем класса алгебраических моделей, реализована в 1968 г . Джерардом Солтоном в поисковой системе SMART (Salton's Magical Automatic Retriever of Text). Сокращенное обозначение TF*IDF – синоним наиболее распространенной современной векторной модели, основанной на математическом аппарате геометрии, в которой индексируемые текстовые ресурсы и запросы пользователей рассматриваются как векторы в пространстве слов, а релевантность – как расстояние между ними. Векторные модели, в отличие от булевых, позволяют ранжировать результирующее множество документов запроса. В векторной модели каждому документу ставится в соответствие вектор $D_i = \{w_{i1}, w_{i2}, \dots, w_{in}\}$, где w_{ij} - вес j - го ключевого слова в i - ом документе, обычно вычисляемый по формуле нормированного представления TF*IDF: $w_{ij} = a_{ij} log \frac{N}{d_j}$, где a_{ij} - частота появления j - го слова в i - ом документе, d_j - количество документов, в которых встречается j - ое ключевое слово, N - общее количество документов. Аналогично для запроса Q вводится вектор $Q = \{q_1, q_2, \dots, q_n\}$, где $q_j = 1$, если j - ое ключевое слово встречается присутствует в запросе Q, иначе $q_j = 0.$ Мера схожести документа D_i и запроса Q вычисляется как косинус угла между соответствующими векторами: $r(D_i, Q) = (D_i, Q)/(||D_i|| * ||Q||)$, где (D_i, Q) - скалярное произведение, $||D_i||, ||Q||$ - нормы векторов.

В реальных поисковых системах, как правило, используется комбинация рассмотренных методов. При этом булев поиск используется для выделения из всего массива тех документов, которые содержат все слова запроса. Для определения

релевантности документов и сортировки полученной выборки используются алгоритмы векторного и вероятностного поиска. Булева составляющая индексирования, сильно ускоряющая процесс поиска, – неотъемлемая часть поисковых систем, что говорит о необходимости создания и поддержки инвертированного индекса.

3.Каталог ресурсов Интернет

Каталог ресурсов Internet — постоянно обновляемая и пополняемая система ссылок на ресурсы, распределенные по иерархической структуре категорий. На верхнем уровне каталога представлены самые общие категории (рубрики), например, “наука”, “бизнес”, “развлечения” и т. д. На ниже лежащих уровнях эти рубрики декомпозируются на подчиненные рубрики, имеющие более частный характер. На нижнем уровне каталога указываются ссылки на конкретные ресурсы Internet (сайты и web-страницы), снабженные краткими описаниями их содержимого.



Каталоги ресурсов Internet незаменимы, когда человек имеет недостаточно точное представление о цели поиска. Некоторые из них позволяют выполнять поиск по ключевым словам в кратком описании содержимого ресурсов.

Каталоги облегчают поиск за счет упорядоченности ссылок на ресурсы. Все интеллектуальные функции остаются за человеком. То же можно сказать о подборках ссылок на информационные ресурсы Internet. Такие подборки представляют собой отсортированные по темам адреса ресурсов.

Формирование и актуализация каталогов и подборок ссылок выполняются вручную персоналом соответствующих информационных систем. Подобная работа требует высокой квалификации и достаточно трудоемка.

Пример универсальных каталогов:

- https://www.google.com/
- https://yandex.by/
- https://www.search.com/
- https://www.wikipedia.org/

Специализированные каталоги:

- https://www.kinopoisk.ru/
- https://www.youtube.com/
- https://www.tut.by/

4. Типовая поисковая машина

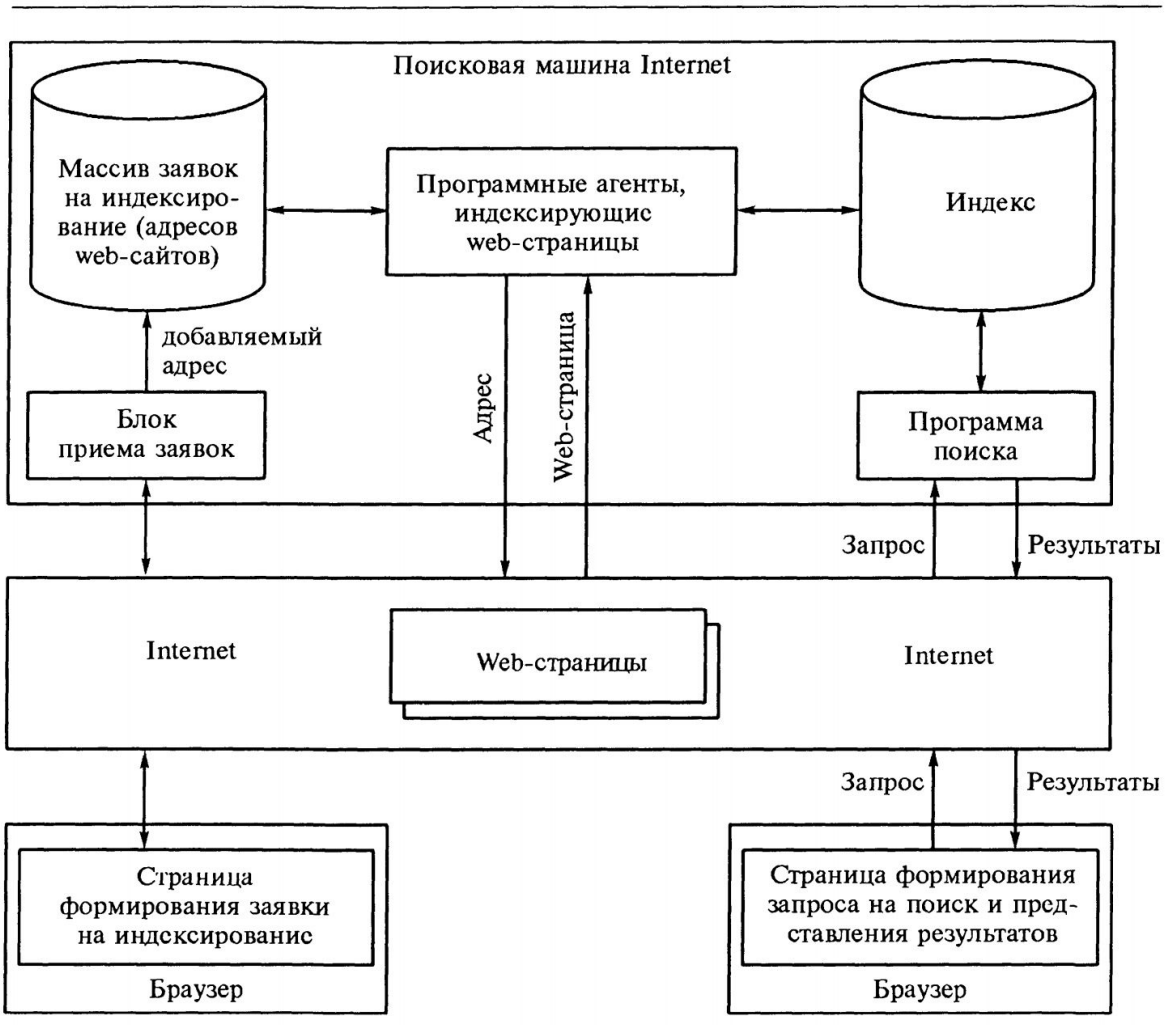
Поисковые машины (или поисковые системы) позволяют находить ресурсы Internet непосредственно по их текстовому содержимому. Функционирование поисковой машины включает два базовых процесса:

1. индексирование ресурсов Internet (автоматическое построение и обновление индекса);
2. поиск по индексу по запросам пользователей.

В Международном каталоге поисковых машин (Search Engine Colossus — http://www.searchenginecolossus.com) зарегистрировано свыше 2300 систем из 232 стран. По данным этого каталога более 80% пользователей Internet находят информационные ресурсы с помощью поисковых машин, 57% пользователей ежедневно применяют поисковые машины, каждый день выполняется до 450 млн поисковых запросов, поисковые машины служат источником сведений для 55% всех покупок в on-line.

Упрощенная структура типовой поисковой машины показана ниже. Ее главными компонентами являются:

- программный агент, “перемещающийся” по сети и индексирующий ресурсы (web-страницы),
- БД (индекс), содержащая информацию, собираемую агентом,
- программа поиска, применяемая пользователями для поиска информации в БД.



4.1.Алгоритм работы поисковой машины

Стандартный алгоритм работа поисковой машины можно описать следующими шагами:

1. Адреса web-узлов, включаемые в обрабатываемую область, определяются по гиперссылкам, ведущим из страниц данного web-узла.

2. Агент переходит к индексированию очередного web-узла из списка либо выполняет дублирование его содержимого на свой web-узел.
3. Индексирование.
4. Данные сохраняются в БД.
5. Дубликаты (в случае их создания в п.2) стираются.
6. Повторять пп. 2-5 для каждого адреса из п.1.

Изложенный алгоритм соответствует некоторой канонической структуре поисковой машины. Конкретные их реализации различаются по многим параметрам:

- поддержке простого и сложного поиска;
- учету различий строчных и прописных символов;
- возможности поиска по частям слов и словосочетаниям;
- поддержке обработки запросов, содержащих логические операторы И, ИЛИ, НЕ;
- использованию специальных языков поиска информации, значительно сокращающих его время (к сожалению, такие языки не стандартизованы, поэтому в разных поисковых машинах реализуются разные поисковые языки).

Применение поисковых машин для поиска в Internet эффективно, если пользователь представляет, какие ключевые слова характеризуют нужные ресурсы.

Дополнительные возможности предоставляет режим расширенного поиска, в котором можно задавать правила поиска. Часто это значительно увеличивает вероятность нахождения требуемой информации.

4.2. Поисковые агенты

Агент - самый интеллектуальный из компонентов поисковой машины. Он обладает автономностью, имеет блоки навигации, управляющие “перемещением” по сети, и механизмы индексации, основанные на некоторой базе правил. Агенты реализуются как простые программные системы, запрашивающие информацию с узлов Internet. Физически по сети агенты не перемещаются. Они индексируют полученные страницы и заносят результаты в БД.

Поисковые механизмы отличаются разнообразием. Некоторые агенты следуют по каждой ссылке на каждой найденной странице и затем, в свою очередь, исследуют каждую ссылку на новой странице и т. д. Как правило, агенты игнорируют ссылки к графическим и мультимедийным файлам, файлам с данными (например, архивам), БД и др. Ряд агентов просматривают страницы с учетом их популярности.

Одной из проблем является реализация алгоритма перемещения (на вигации) по сети. Учитывая, что большинство web-серверов организовано иерархически, перемещение вширь по ссылкам от исходной вершины при ограниченной глубине вложенности с большей вероятностью приводит к нахождению документов с высоким уровнем релевантности, чем при перемещении в глубину. Поскольку это подтверждается статистикой работы поисковых машин, данный метод (сначала вширь, затем вглубь) принят как предпочтительный для индексирования web-ресурсов.

Системой правил для всего этого сообщества автономных программ управляют администраторы поисковых машин. Они же устанавливают параметры алгоритмов определения степени релевантности документа и запроса. Обычно в этих алгоритмах учитываются:

- количество слов запроса в текстовом содержимом документа (т. е. в HTML-коде);
- теги, в которых эти слова встречаются;
- местоположение искомых слов в документе;
- удельный вес слов, относительно которых определяется релевантность, в общем количестве слов документа;
- время существования web-сайта;
- индекс цитируемости web-сайта и др.

Литература

1. Машинное понимание текстов естественного языка: онтологическая парадигма: http://dspace.nbuv.gov.ua/bitstream/handle/123456789/56278/29-Sviatogor.pdf?sequence=1
2. Проблема понимания в системах искусственного интеллекта: https://cyberleninka.ru/article/n/problema-ponimaniya-v-sistemah-iskusstvennogo-intellekta
3. Интеллектуальные информационные технологии: https://studfiles.net/preview/6828962/
4. Обработка текстов на естественном языке: https://www.osp.ru/os/2003/12/183694/
5. Обзор методов информационного поиска: https://cyberleninka.ru/article/v/obzor-metodov-informatsionnogo-poiska