

Как работают поисковые системы

Илья Сегалович,
iseg@yandex-team.ru

В мире написаны сотни *поисковых систем*, а если считать функции поиска, реализованные в самых разных программах, то счет надо вести на тысячи. И как бы ни был реализован процесс поиска, на какой бы математической модели он не основывался, идеи и программы, реализующих поиск, достаточно просты. Хотя эта простота, относится, по-видимому, к той категории, про которую говорят «просто, но работает». Так или иначе, но именно поисковые системы стали одним из двух новых чудес света, предоставив Homo Sapiens неограниченный и мгновенный доступ к информации. Первым чудом, очевидно, можно считать Интернет как таковой, с его возможностями всеобщей коммуникации.

ПОИСКОВЫЕ СИСТЕМЫ В ИСТОРИЧЕСКОЙ ПЕРСПЕКТИВЕ

Существует распространенное убеждение, что каждое новое поколение программ совершенней предыдущего. Дескать, раньше все было несовершенно, зато теперь повсюду царит чуть ли не *искусственный интеллект*. Иная крайняя точка зрения состоит в том, что «все новое - это хорошо забытое старое». Думаю, что применительно к поисковым системам истина лежит где-то посередине.

Но что же поменялось в действительности за последние годы? Не алгоритмы и не структуры данных, не математические модели. Хотя и они тоже. Поменялась парадигма использования систем. Проще говоря, к экрану со строчкой поиска подсади домохозяйка, ищущая уютг подешевле, и выпускник вспомогательного интерната в надежде найти работу автомеханика. Кроме появления фактора, невозможного в доинтернетовскую эру – фактора тотальной востребованности поисковых систем – стала очевидна еще пара изменений. Во-первых, стало ясно, что люди не только «думают словами», но и «ищут словами». В ответе системы они ожидают увидеть слово, набранное в строке запроса. И второе: «человека ищущего» трудно «переучить искать», так же как трудно переучить говорить или писать. Мечты 60-х – 80-х об итеративном уточнении запросов, о понимании естественного языка, о поиске по смыслу, о генерации связного ответа на вопрос с трудом выдерживают сейчас жестокое испытание реальностью.

АЛГОРИТМ + СТРУКТУРА ДАННЫХ = ПОИСКОВАЯ СИСТЕМА

Как и любая программа, поисковая система оперирует со структурами данных и исполняет алгоритм. Разнообразие алгоритмов не очень велико, но оно есть. Не считая квантовых компьютеров, которые обещают нам волшебный прорыв в «алгоритмической сложности» поиска, и про которые автору почти ничего не известно, есть четыре класса поисковых алгоритмов. Три алгоритма из четырех требуют «индексирования», предварительной обработки документов, при котором создаются вспомогательный файл, сиречь «индекс», призванный упростить и ускорить сам поиск. Это алгоритмы *инвертированных файлов*, *суффиксных*

деревьев, сигнатур. В вырожденном случае предварительный этап индексирования отсутствует, а поиск происходит при помощи последовательного просмотра документов. Такой поиск называется *прямым*.

прямой поиск

Простейшая его версия знакома многим, и нет программиста, который бы не написал хотя бы раз в своей жизни подобный код:

```
char* strstr(char *big, char *little)
{
    char *x, *y, *z;
    for (x = big; *x; x++)
    {
        for (y = little, z = x; *y;
             ++y, ++z)
        {
            if (*y != *z)
                break;
        }
        if (!*y)
            return x;
    }
    return 0;
}
```

В этой функции языка C текст строки **big** просматривают слева направо и для каждой позиции **x** запускают последовательное сравнение с искомой подстрокой **little**. Для этого, двигая одновременно два указателя **y** и **z**, попарно сравнивают все символы. Если мы успешно дошли до конца искомой подстроки, значит она найдена.

Несмотря на кажущуюся простоту, последние 30 лет прямой поиск интенсивно развивается. Было выдвинуто немалое число идей, сокращающих время поиска в разы. Эти алгоритмы подробно описаны в разнообразной литературе, есть их сводки и сопоставления. Неплохие обзоры прямых методов поиска можно найти в учебниках, например [**sedgewick**] или [**кормен**]. При этом надо учесть, что новые алгоритмы и их улучшенные варианты появляются постоянно.

Хотя прямой просмотр всех текстов – довольно медленное занятие, не следует думать, что алгоритмы прямого поиска не применяются в интернете. Норвежская поисковая система Fast (www.fastsearch.com) использовала чип, реализующий логику прямого поиска упрощенных регулярных выражений [**fastpmc**], и разместила 256 таких чипов на одной плате. Это позволяло Fast-у обслуживать довольно большое количество запросов в единицу времени.

Кроме того, есть масса программ, комбинирующих индексный поиск для нахождения блока текста с дальнейшим прямым поиском внутри блока. Например, весьма популярный, в том числе и в Рунете, **glimpse** [**glimpse**].

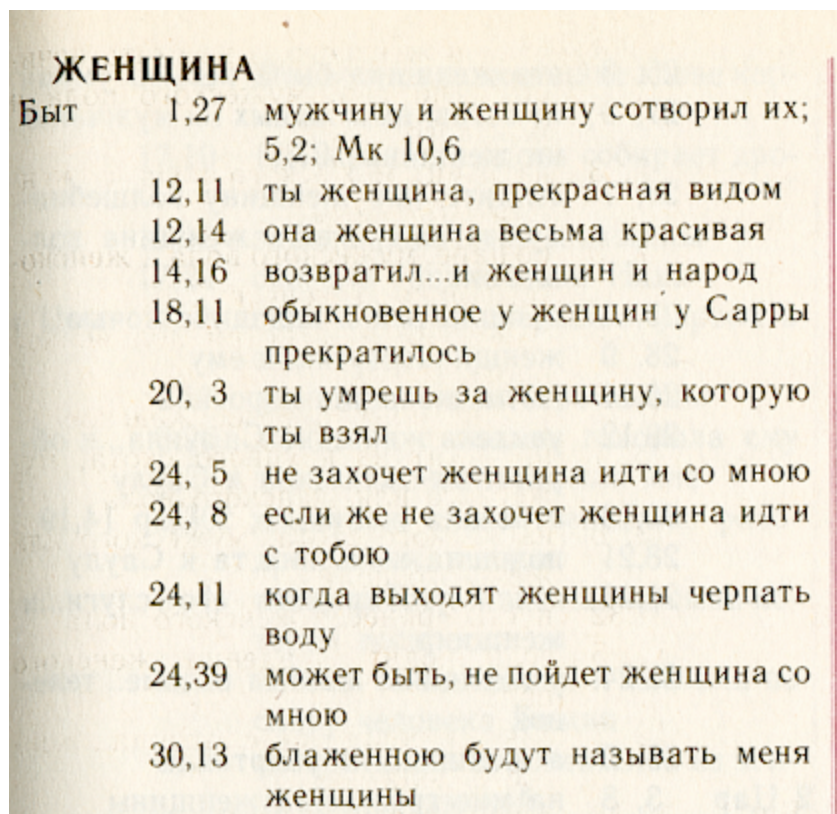
Вообще, у прямых алгоритмов есть принципиально беспроегрывные отличительные черты. Например, неограниченные возможности по приближенному и нечеткому поиску. Ведь любое индексирование всегда сопряжено с упрощением и нормализацией терминов, а, следовательно, с потерей

информации. Прямой же поиск работает непосредственно по оригинальным документам безо всяких искажений.

инвертированный файл

Эта простейшая структура данных, несмотря на свое загадочное иностранное название, интуитивно знакома любому грамотному человеку, так и любому программисту баз данных, даже не имевшему дело с полнотекстовым поиском. Первая категория людей знает, что это такое, по «конкордансам» - алфавитно упорядоченным исчерпывающим спискам слов из одного текста или принадлежащих одному автору (например «Конкорданс к стихам А. С. Пушкина», «Словарь-конкорданс публицистики Ф. М. Достоевского»). Вторые имеют дело с той или иной формой инвертированного списка всякий раз, когда строят или используют «индекс БД по ключевому полю».

Проиллюстрируем эту структуру при помощи замечательного русского конкорданса - «Симфонии», выпущенной московской патриархией по тексту синодального перевода Библии [симфония].



ЖЕНЩИНА		
Быт	1,27	мужчину и женщину сотворил их; 5,2; Мк 10,6
	12,11	ты женщина, прекрасная видом
	12,14	она женщина весьма красивая
	14,16	возвратил...и женщин и народ
	18,11	обыкновенное у женщин у Сарры прекратилось
	20, 3	ты умрешь за женщину, которую ты взял
	24, 5	не захочет женщина идти со мною
	24, 8	если же не захочет женщина идти с тобою
	24,11	когда выходят женщины черпать воду
	24,39	может быть, не пойдет женщина со мною
	30,13	блаженною будут называть меня женщины

Рис. 1

Перед нами упорядоченный по алфавиту список слов. Для каждого слова перечислены все «позиции», в которых это слово встретилось. Поисковый

алгоритм состоит в отыскании нужного слова и загрузке в память уже развернутого списка позиций.

Чтобы сэкономить на дисковом пространстве и ускорить поиск, обычно прибегают к двум приемам. Во-первых, можно сэкономить на подробности самой позиции. Ведь чем подробнее задана такая позиция, например, в случае с «Симофонией» это «книга+глава+стих», тем больше места потребуется для хранения инвертированного файла.

В наиподробнеешем варианте в инвертированном файле можно хранить и номер слова, и смещение в байтах от начала текста, и цвет и размер шрифта, да много чего еще. Чаще же просто указывают только номер документа, скажем, книгу Библии, и число употреблений этого слова в нем. Именно такая упрощенная структура считается основной в классической теории информационного поиска – Information Retrieval (IR).

Второй (никак не связанный с первым) способ сжатия: упорядочить позиции для каждого слова по возрастанию адресов и для каждой позиции хранить не полный ее адрес, а разницу от предыдущего. Вот как будет выглядеть такой список для нашей странички в предположении, что мы запоминаем позицию вплоть до номера главы:

ЖЕНЩИНА : [Быт . 1] , [+11] , [0] , [+2] , [+4] , [+2] , [+4] , . .
--

Дополнительно на разностный способ хранения адресов накладывают какой-нибудь простенький способ упаковки: зачем отводить небольшому целому числу фиксированное «огромное» количество байт, ведь можно отвести ему почти столько байт, сколько оно заслуживает. Здесь уместно упомянуть коды Голомба или встроенную функцию популярного языка Perl: `pack ("w")`.

В литературе встречается и более тяжелая артиллерия упаковочных алгоритмов самого широкого спектра: арифметический, Хафман, LZW, и т.д. Прогресс в этой области идет непрерывно. На практике в поисковых системах они используются редко: выигрыш невелик, а мощности процессора расходуются неэффективно.

В результате всех описанных ухищрений размер инвертированного файла, как правило, составляет от 7 до 30 процентов от размера исходного текста, в зависимости от подробности адресации.

занесены в «Красную книгу»

Неоднократно предлагались другие, отличные от инвертированного и прямого поиска алгоритмы и структуры данных. Это, прежде всего, суффиксные деревья [**manber**], [**gonnet**], а также сигнатуры [**faloutsos**].

Первый из них функционировал и в интернете, будучи запатентованным алгоритмом поисковой системы OpenText [**opentext**]. Мне доводилось встречать суффиксные индексы в отечественных поисковых системах.

Второй - метод сигнатур - представляет собой преобразование документа к поблочным таблицам *хеш-значений* его слов - "сигнатуре" и последовательному просмотру "сигнатур" во время поиска.

Широкого распространения ни тот ни другой метод не получили, а, следовательно, не заслужили и подробного обсуждения в этой небольшой статье.

МАТЕМАТИЧЕСКИЕ МОДЕЛИ

Приблизительно 3 из 5 поисковых систем и модулей функционируют безо всяких математических моделей. Точнее сказать, их разработчики не ставят перед собой задачу реализовывать абстрактную модель и/или не подозревают о существовании оной. Принцип здесь прост: лишь бы программа хоть что-нибудь находила. Абы как. А дальше сам пользователь разберется.

Однако, как только речь заходит о повышении качества поиска, о большом объеме информации, о потоке пользовательских запросов, кроме эмпирически предоставленных коэффициентов полезным оказывается оперировать каким-нибудь пусть и несложным теоретическим аппаратом. *Модель поиска* – это некоторое упрощение реальности, на основании которого получается формула (сама по себе никому не нужная), позволяющая программе принять решение: какой документ считать найденным и как его ранжировать. После принятия модели коэффициенты часто приобретают физический смысл и становятся понятней самому разработчику, да и подбирать их становится интересней.

Все многообразие моделей традиционного информационного поиска (IR) принято делить на три вида: теоретико-множественные (булевская, нечетких множеств, расширенная булевская), алгебраические¹ (векторная, обобщенная векторная, латентно-семантическая, нейросетевая) и вероятностные.

Булевское семейство моделей, по сути, – первое, приходящее на ум программисту, реализующему полнотекстовый поиск. Есть слово - документ считается найденным, нет – не найденным. Собственно, классическая булевская модель – это мостик, связывающий теорию информационного поиска с теорией поиска и манипулирования данными.

Критика булевской модели, вполне справедливая, состоит в ее крайней жесткости и непригодности для ранжирования. Поэтому еще в 1957 году Joysе и Needham (Джойс и Нидхэм) предложили учитывать частотные характеристики слов, чтобы «... операция сравнения была бы отношением расстояния между векторами...» [joysе_1957*]. *Векторная модель* и была с успехом реализована в 1968 году отцом-основателем науки об информационном поиске Джерардом Солтоном (Gerard Salton)² в поисковой системе SMART (Salton's Magical Automatic Retriever of Text).

¹ В отечественной литературе алгебраические модели часто называют линейными

² Gerard Salton (Sahlman) 1927-1995. Он же Селтон, он же Залтон и даже Залман, он же Жерар, Герард, Жерард или даже Джеральд в зависимости от вкуса переводчика и допущенных опечаток <http://www.cs.cornell.edu/Info/Department/Annual95/Faculty/Salton.html>

Ранжирование в этой модели основано на естественном статистическом наблюдении, что чем больше локальная частота термина в документе (TF) и больше «редкость» (т.е. *обратная встречаемость в документах*) термина в коллекции (IDF), тем выше вес данного документа по отношению к термину. Обозначение IDF ввела Karen Sparck-Jones (Карен Спарк-Джоунз) в 1972 в статье **[spark-jones]** про *различительную силу (term specificity)*. С этого момента обозначение $TF*IDF$ широко используется как синоним векторной модели.

Наконец, в 1977 году Robertson и Sparck-Jones (Робертсон и Спарк-Джоунз) **[robertson]** обосновали и реализовали *вероятностную модель* (предложенную еще в 1960 **[maron]**), также положившую начало целому семейству. *Релевантность* в этой модели рассматривается как вероятность того, что данный документ может оказаться интересным пользователю. При этом подразумевается наличие уже существующего первоначального набора релевантных документов, выбранных пользователем или полученных автоматически при каком-нибудь упрощенном предположении. Вероятность оказаться релевантным для каждого следующего документа рассчитывается на основании соотношения встречаемости терминов в релевантном наборе и в остальной, «нерелевантной» части коллекции. Хотя вероятностные модели обладают некоторым теоретическим преимуществом, ведь они располагают документы в порядке убывания "вероятности оказаться релевантным", на практике они так и не получили большого распространения.

Я не собираюсь вдаваться в подробности и выписывать громоздкие формулы для каждой модели. Их сводка вместе с обсуждением занимает в сжатом виде 35 страниц в книжке «Современный информационный поиск» **[baezo-yates]**. Важно только заметить, что в каждом из семейств простейшая модель исходит из предположения о взаимонезависимости слов и обладает простым условием фильтрации: документы, не содержащие слова запроса, никогда не бывают найденными. Продвинутые («альтернативные») модели каждого из семейств не считают слова запроса взаимонезависимыми, а, кроме того, позволяют находить документы, не содержащие ни одного слова из запроса.

Поиск «по смыслу»

Способность находить и ранжировать документы, не содержащие слов из запроса, часто считают признаком искусственного интеллекта или *поиска по смыслу* и относят априори к преимуществам модели. Вопросы, о том, так ли это или нет мы оставим за рамками данной статьи.

Для примера опишу лишь одну, пожалуй, самую популярную модель, работающую по смыслу. В теории информационного поиска данную модель принято называть *латентно-семантически индексированием* (иными словами, выявлением скрытых смыслов). Эта алгебраическая модель основана на сингулярном разложении прямоугольной матрицы, ассоциирующей слова с документами. Элементом

матрицы является частотная характеристика, отражающая степень связи слова и документа, например, $TF*IDF$. Вместо исходной миллионно-размерной матрицы авторы метода [furnas], [deerwester] предложили использовать 50-150 «скрытых смыслов»³, соответствующих первым *главным компонентам* ее *сингулярного разложения*.

Сингулярным разложением действительной матрицы A размеров $m*n$ называется всякое ее разложение вида $A = USV$, где U - ортогональная матрица размеров $m*m$, V - ортогональная матрица размеров $n*n$, S - диагональная матрица размеров $m*n$, элементы которой $s_{ij} = 0$, если i не равно j , и $s_{ii} = s_i \geq 0$. Величины s_i называются сингулярными числами матрицы и равны арифметическим значениям квадратных корней из соответствующих собственных значений матрицы AA^T . В англоязычной литературе сингулярное разложение принято называть SVD-разложением.

Давным-давно доказано [eckart], что если оставить в рассмотрении первые k сингулярных чисел (остальные приравнять нулю), мы получим ближайшую из всех возможных аппроксимацию исходной матрицы ранга k (в некотором смысле ее «ближайшую семантическую интерпретацию ранга k »). Уменьшая ранг, мы отфильтровываем нерелевантные детали; увеличивая, пытаемся отразить все нюансы структуры реальных данных.

Операции поиска или нахождения *похожих документов* резко упрощаются, так как каждому слову и каждому документу сопоставляется относительно короткий вектор из k смыслов (строки и столбцы соответствующих матриц). Однако по причине малой ли осмысленности «смыслов», или по какой иной⁴, но использование LSI в лоб для поиска так и не получило распространения. Хотя во вспомогательных целях (автоматическая фильтрация, классификация, разделение коллекций, предварительное понижение размерности для других моделей) этот метод, по-видимому, находит применение.

ОЦЕНКА КАЧЕСТВА

*Consistency checking has shown that the overlap of relevant documents between any two assessors is on the order of 40% on average... cross-assessor recall and precision of about 65% ... This implies a practical upper bound on retrieval system performance of 65% ...*⁵

Donna Harman

What we have learned, and not learned, from TREC [harman]

³ для больших коллекций число «смыслов» увеличивают до 300

⁴ После наших экспериментов с LSI получилось, что «смысл номер 1» в Рунете - все англоязычные документы, «смысл номер 3» – все форумы и т.п.

⁵ «...проверка устойчивости показала, что перекрытие релевантных документов между любыми двумя ассессорами примерно 40% в среднем ... точность и полнота измеренная между ассессорами около 65% ... Это накладывает практическую верхнюю границу на качество поиска в районе 65%...»

Какова бы ни была модель, поисковая система нуждается в «тюнинге» - оценке качества поиска и настройке параметров. Оценка качества – идея, фундаментальная для теории поиска. Ибо именно благодаря оценке качества можно говорить о применимости или не применимости той или иной модели и даже обсуждать их теоретические аспекты.

В частности, одним из естественных ограничений качества поиска служит наблюдение, вынесенное в эпиграф: мнения двух «асессоров» (специалистов, выносящих вердикт о релевантности) в среднем не совпадают друг с другом в очень большой степени! Отсюда вытекает и естественная верхняя граница качества поиска, ведь качество измеряется по итогам сопоставления с мнением асессора.

Обычно⁶ для оценки качества поиска меряют два параметра:

- *точность* (precision) – доля релевантного материала в ответе поисковой системы
- *полнота* (recall) – доля найденных релевантных документов в общем числе релевантных документов коллекции

Именно эти параметры использовались и используются на регулярной основе для выбора моделей и их параметров в рамках созданной Американским Институтутом Стандартов (NIST) конференции по оценке систем текстового поиска (TREC - text retrieval evaluation conference)⁷. Начавшаяся в 1992 году консорциумом из 25 групп, к 12-му году своего существования конференция накопила значительный материал, на котором до сих пор оттачиваются поисковые системы. К каждой очередной конференции готовится новый материал (т.н. «дорожка») по каждому из интересующих направлений. «Дорожка» включает коллекцию документов и запросов. Приведу примеры:

- Дорожка произвольных запросов (*ad hoc*) – присутствует на всех конференциях
- Многоязычный поиск
- Маршрутизация и фильтрации
- Высокоточный поиск (с единственным ответом, выполняемый на время)
- Взаимодействие с пользователем
- Естественно-языковая «дорожка»
- Ответы на «вопросы»
- Поиск в «грязных» (только что отсканированных) текстах
- Голосовой поиск
- Поиск в очень большом корпусе (20GB, 100GB и т.д.)
- WEB корпус (на последних конференциях он представлен выборкой по домену .gov)
- Распределенный поиск и слияние результатов поиска из разных систем

⁶ но не обязательно – есть и «альтернативные» метрики!

⁷ материалы конференции публично доступны по адресу trec.nist.gov/pubs.html

НЕ ТОЛЬКО ПОИСК

Как видно из «дорожек» TREC, к самому поиску тесно примыкает ряд задач, либо разделяющих с ним общую идеологию (классификация, маршрутизация, фильтрация, аннотирование), либо являющихся неотъемлемой частью поискового процесса (кластеризация результатов, расширение и сужение запросов, обратная связь, «запросо-зависимое» аннотирование, поисковый интерфейс и языки запросов). Нет ни одной поисковой системы, которой бы не приходилось решать на практике хотя бы одну из этих задач.

Зачастую наличие того или иного дополнительного свойства является решающим доводом в конкурентной борьбе поисковых систем. Например, краткие аннотации состоящие из информативных цитат документа, которыми некоторые поисковые системы сопровождают результаты своей работы, помогают им оставаться на полступеньки впереди конкурентов.

Обо всех задачах и способах их решения рассказать невозможно. Для примера рассмотрим «расширение запроса», которое обычно производится через привлечение к поиску ассоциированных терминов. Решение этой задачи возможно в двух видах – локальном (динамическом) и глобальном (статическом). Локальные техники опираются на текст запроса и анализируют только документы, найденные по нему. Глобальные же «расширения» могут оперировать тезаурусами, как априорными (лингвистическими), так и построенными автоматически по всей коллекции документов. По общепринятому мнению, глобальные модификации запросов через тезаурусы работают неэффективно, понижая точность поиска. Более успешный глобальный подход основан на построенных вручную статических классификациях, например, ВЕБ-директориях. Этот подход широко используется в интернет-поисковиках в операциях сужения или расширения запроса.

Нередко реализация дополнительных возможностей основана на тех же самых или очень похожих принципах и моделях, что и сам поиск. Сравните, например, нейросетевую поисковую модель, в которой используется идея передачи затухающих колебаний от слов к документам и обратно к словам (амплитуда первого колебания – все тот же $TF*IDF$), с техникой локального расширения запроса. Последняя основана на *обратной связи* (relevance feedback), в которой берутся наиболее *смыслоразличительные* (контрастные) слова из документов, принадлежащих верхушке списка найденного.

К сожалению, локальные методы расширения запроса, несмотря на эффективные технические идеи типа «Term Vector Database» [**stata**] и очевидную пользу, все еще остаются крайне «дорогим»⁸ удовольствием.

⁸ в смысле вычислительных ресурсов

ЛИНГВИСТИКА

Немного в стороне от статистических моделей и структур данных стоит класс алгоритмов, традиционно относимых к лингвистическим. Точно границы между статистическим и лингвистическими методами провести трудно. Условно можно считать лингвистическими методы, опирающиеся на словари (морфологические, синтаксические, семантические), созданные человеком. Хотя считается доказанным, что для некоторых языков лингвистические алгоритмы не вносят существенного прироста точности и полноты (например, английский) [strzalkowski], все же основная масса языков требует хотя бы минимального уровня лингвистической обработки. Не вдаваясь в подробности, приведу только список задач, решаемый лингвистическими или околотингвистическими приемами:

- автоматическое определение языка документа
- *токенизация* (графематический анализ): выделение слов, границ предложений
- исключение неинформативных слов (*стоп-слов*)
- *лемматизация* (нормализация, *стемминг*): приведение *словоизменительных* форм к «словарной». В том числе и для слов, не входящих в словарь системы
- разделение сложных слов (компаундов) для некоторых языков (например, немецкого)
- *дизамбигуация*: полное или частичное снятие *омонимии*
- выделение *именных групп*

Еще реже в исследованиях и на практике можно встретить алгоритмы *словообразовательного, синтаксического* и даже *семантического* анализа. При этом под семантическим анализом чаще подразумевают какой-нибудь статистический алгоритм (LSI, нейронные сети), а если толково-комбинаторные или семантические словари и используются, то в крайне узких предметных областях.

ПОИСК В ВЕБЕ

“Things that work well on TREC often do not produce good results on the web ... Some argue that on the web, users should specify more accurately what they want and add more words to their query. We disagree vehemently with this position. If a user issues a query like "Bill Clinton" they should get reasonable results since there is a enormous amount of high quality information available on this topic”⁹

Sergei Brin, Larry Page

The Anatomy of a Large-Scale Hypertextual Web Search Engine [*]

<http://www7.scu.edu.au/programme/fullpapers/1921/com1921.htm>

“I was struck when a Google person told me at SIGIR that the most recent Google ranking algorithm completely ignores anything discovered at

⁹ ... то, что хорошо работает в TREC часто не срабатывает в вебе ... некоторые утверждают, что в вебе пользователи обязаны более точно специфицировать то, что им нужно, писать побольше слов в запросах. Мы категорически не согласны с такой точкой зрения. Если люди спрашивают «Билл Клинтон» они должны получать осмысленные результаты, так как в вебе полным полно качественной информации на эту тему.... Сергей Брин, Ларри Пейдж

*TREC, because all the good Ad Hoc ranking algorithms developed over the 10 years of TREC get trashed by spam*¹⁰

Mark Sanderson [*]

<http://groups.yahoo.com/group/webir/message/710>

Пора вернуться к теме, с которой началась эта статья: что же изменилось поисковых системах за последнее время?

Прежде всего, стало очевидно, что поиск в вебе, не может быть сколько-нибудь корректно выполнен, будучи основан на анализе (пусть даже сколь угодно глубоко, семантическом и т.п.) одного лишь текста документа. Ведь *внетекстовые* (off-page) факторы играют не меньшую, а порой и большую роль, чем текст самой страницы. Положение на сайте, посещаемость, авторитетность источника, частота обновления, цитируемость страницы и ее авторов – все эти факторы невозможно сбрасывать со счета.

Став основным источником получения справочной информации для человеческого вида, поисковые системы стали основным источником трафика для интернет-сайтов. Как следствие, они немедленно подверглись «атакам» недобросовестных авторов, желающих любой ценой оказаться в первых страницах результатов поиска. Искусственная генерация *входных страниц*, насыщенных популярными словами, техника *клоакинга*, «слепого текста» и многие другие приемы, предназначенные для обмана поисковых систем, мгновенно заполнили Интернет.

Кроме проблемы корректного ранжирования, создателям поисковых систем в Интернете пришлось решать задачу обновления и синхронизации колоссальной по размеру коллекции с гетерогенными форматами, способами доставки, языками, кодировками, массой бессодержательных и дублирующихся текстов. Необходимо поддерживать базу в состоянии максимальной свежести (на самом деле достаточно создавать *иллюзию свежести* - но это тема отдельного разговора), может быть учитывать индивидуальные и коллективные предпочтения пользователей. Многие из этих задач никогда прежде не рассматривались в традиционной науке информационного поиска.

Для примера рассмотрим пару таких задач и практических способов их решения в поисковых системах для интернета.

Качество ранжирования

Не все внетекстовые критерии полезны в равной мере. Именно ссылочная популярность и производные от нее оказались решающим фактором, поменявшим в 1999-2000 мир поисковых систем и вернувший им преданность пользователей. Так как именно с ее помощью поисковые системы научились прилично и самостоятельно (без подпорок из вручную отредактированных результатов)

¹⁰ ... я был потрясен, когда кто-то из Гугля сказал мне, что они вообще не используют ничего наработанного в TREC, потому что все алгоритмы заточенные на дорожке «произвольных запросов» спам рашибает вдребезги... Марк Сандерсон

ранжировать ответы на короткие частотные запросы, составляющие значительную часть поискового потока.

Простейшая идея глобального (т.е. статического) учета ссылочной популярности состоит в подсчете числа ссылок, указывающих на страницы. Примерно то, что в традиционном библиотековедении называют *индексом цитирования*. Этот критерий использовался в поисковых системах еще до 1998 года. Однако он легко подвергается *накрутке*, кроме того, он не учитывает вес самих источников. Естественным развитием этой идеи можно считать предложенный Брином и Лейджем в 1998 году алгоритм *PageRank [brin]* – итеративный алгоритм, подобный тому, что используется в задаче определения победителя в шахматной турнире по швейцарской системе. В сочетании с поиском по лексике ссылок, указывающих на страницу (старая, весьма продуктивная идея, которая использовалась в гипертекстовых поисковых системах еще в 80-е годы), эта мера позволила резко повысить качество поиска.

Немного раньше, чем PageRank, был предложен локальный (т.е. динамический, основанный на запросе) алгоритм учета популярности – *HITS [kleinberg]*, который¹¹ не используется на практике в основном из-за вычислительной дороговизны. Примерно по той же причине, что и локальные (т.е. динамические) методы, оперирующие словами.

Оба алгоритма, их формулы, условия сходимости подробно описаны, в том числе и в русскоязычной литературе. Отмечу только, что расчет *статической популярности* не является самоценной задачей, он используется в многочисленных вспомогательных целях: определение порядка обхода документов, ранжирование поиска по тексту ссылок и т.д. Формулы расчета популярности постоянно улучшают, в них вносят учет дополнительных факторов: тематической близости документов (например, популярная поисковая система www.teoma.com), их структуры, и т.п., позволяющие понизить влияние *непотизма*. Интересной отдельной темой является эффективная реализация соответствующих структур данных **[bharat]**

Качество индекса

Хотя размер базы в интернете на поверхностный взгляд не кажется критическим фактором, это не так. Недаром рост посещаемости таких машин, как Google и Fast хорошо коррелируют именно с ростом их баз. Основная причины: «редкие» запросы, то есть те, по которым находится менее 100 документов, составляют в сумме около 30% от всей массы поисков – весьма значительную часть. Этот факт делает размер базы одним из самых критичных параметров системы.

Однако рост базы, кроме технических проблем с дисками и серверами, ограничивается логическими: необходимостью адекватно реагировать на мусор, повторы и т.п. Не могу удержаться, чтобы не описать остроумный алгоритм,

¹¹ точнее производные от него, так как сам алгоритм оказался не очень устойчив

применяемый в современных поисковых системах для того, чтобы исключить «очень похожие документы».

Происхождение копий документов в Интернете может быть различным. Один и тот же документ на одном и том же сервере может отличаться по техническим причинам: быть представлен в разных кодировках и форматах; может содержать переменные вставки – рекламу или текущую дату.

Широкий класс документов в вебе активно копируется и редактируется – ленты новостных агентств, документация и юридические документы, прейскуранты магазинов, ответы на часто задаваемые вопросы и т.д. Популярные типы изменений: корректура, реорганизация, ревизия, реферирование, раскрытие темы и т.д. Наконец, публикации могут быть скопированы с нарушением авторских прав и изменены злонамеренно с целью затруднить их обнаружение.

Кроме того, индексация поисковыми машинами страниц, генерируемых из баз данных, порождает еще один распространенных класс внешне мало отличающихся документов: анкеты, форумы, страницы товаров в электронных магазинах

Очевидно, что с полными повторами проблем особых нет, достаточно сохранять в индексе контрольную сумму текста и игнорировать все остальные тексты с такой же контрольной суммой. Однако этот метод не работает для выявления хотя бы чуть-чуть измененных документов.

Для решения этой задачи Udi Manber (Уди Манбер) (автор известной программы приближенного прямого поиска *agrep*) в 1994 году предложил идею **[manber1994]**, а Andrei Broder (Андрей Бродер) в 1997 **[broder]** придумал название и довел до ума алгоритм «шинглов» (от слова *shingles*, «черепички, чешуйки»). Вот его примерное описание.

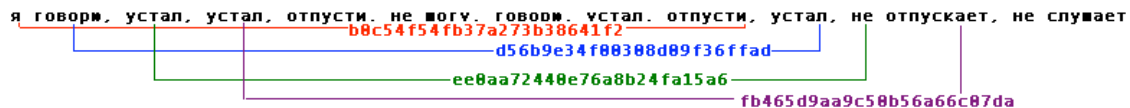


Рис. 2

Для каждого десятисливия текста рассчитывается контрольная сумма (*шингл*). Десятисливия идут внахлест, с перекрытием, так, чтобы ни одно не пропало. А затем из всего множества контрольных сумм (очевидно, что их столько же, сколько слов в документе минус 9) отбираются только те, которые делятся на, скажем, 25. Поскольку значения контрольных сумм распределены равномерно, критерий выборки никак не привязан к особенностям текста. Ясно, что повтор даже одного десятисливия – весомый признак дублирования, если же их много, скажем, больше половины, то с определенной (несложно оценить вероятность) уверенностью можно утверждать: копия найдена! Ведь один совпавший шингл в выборке соответствует примерно 25 совпавшим десятисливиям в полном тексте!

Очевидно, что так можно определять процент перекрытия текстов, выявлять все его источники и т.п. Этот изящный алгоритм воплотил давнюю мечту доцентов: отныне мучительный вопрос «у кого студент списывал этот курсовик» можно считать решенным! Легко оценить долю плагиата в любой статье¹².

Чтобы у читателя не создалось впечатление, что информационный поиск исключительно *западная наука*, упомяну про альтернативный алгоритм определения *почти-дубликатов*, придуманный и воплощенный у нас в Яндексе [ilyinsky]. В нем используется тот факт, что большинство поисковых систем уже обладают индексом в виде инвертированного файла (или *инвертированным индексом*) и этот факт удобно использовать в процедуре нахождения почти-дубликатов.

ЦЕНА ОДНОГО ПРОЦЕНТА

Архитектурно современные поисковые системы представляют собой сложные многокомпьютерные комплексы. Начиная с некоторого момента по мере роста системы основная нагрузка ложится вовсе не на работа, а на поиск. Ведь в течении секунды приходит десятки и сотни запросов.

Для того, чтобы справиться с этой проблемой, индекс разбивают на части и раскладывают по десяткам, сотням и даже тысячам компьютеров. Сами компьютеры, начиная с 1997 года (поисковая система Inktomi) представляют собой обычные 32-битные машины (Linux, Solaris, FreeBSD, Win32) с соответствующими ограничениями по цене и производительности. Исключением из общего правила осталась лишь AltaVista, которая с самого начала использовала относительно «большие» 64-битные компьютеры Alpha.

Поисковые системы для Интернета (и, вообще, все большие поисковые системы) могут ускорять свою работу при помощи техник *эшелонирования* и *прюнинга*. Первая техника состоит в разделении индекса на заведомо более релевантную и менее релевантную части. Поиск сначала выполняется в первой, а затем, если ничего не найдено, или найдено мало, поисковая система обращается ко второй части индекса. Pruning (от англ. отсечение, сокращение), состоит в том, чтобы динамически прекращать обработку запроса после накопления достаточного количества релевантной информации. Бывает еще статический pruning, когда на основании некоторых допущений индекс сокращается за счет таких документов, которые заведомо никогда не будут найдены.

Отдельная проблема – организовать бесперебойную работу многокомпьютерных комплексов, бесшовное обновление индекса, устойчивость к сбоям и задержкам с ответами отдельных компонент. Для общения между поисковыми серверами и серверам, собирающими отклики и формирующими страницу выдачи разрабатываются специальные протоколы.

¹² в т.ч. и в данной; надеюсь, что 0%; можете проверить

Заметьте, что один процент производительности (скажем неудачно написанный оператор в каком-нибудь цикле) для десятидесятикомпьютерной¹³ системы стоит примерно ста компьютеров. Поэтому, можно себе представить, как вычищается код, отвечающий за поиск и ранжирование результатов, как оптимизируется использование всех возможных ресурсов: каждого байта памяти, каждого обращения к диску.

Решающее значение приобретает продумывание архитектуры всего комплекса с самого начала, так как любые изменения, например добавление необычного фактора при ранжировании или сложного источника данных, становится исключительно болезненной и сложной процедурой. Очевидно, системы стартующие позже, имеют в это ситуации преимущество. Но инертность пользователей весьма высока, так, например, требуется 2-4 года, чтобы сформированная многомиллионная аудитория сама, пусть и медленно, но перешла на непривычную поисковую систему, даже при наличии у нее неоспоримых преимуществ. В условиях жесткой конкуренции, это порой неосуществимо.

¹³ размер кластера Google в конце 2001 – начале 2002 года

Список литературы

- **[baezo-yates]**
Modern Information Retrieval
Baezo-Yates R. and Ribeiro-Neto B.
ACM Press Addison Wesley, 1999
- **[bharat]**
The Connectivity Server: fast access to linkage information on the Web
K. Bharat, A. Broder, M. Henzinger, P. Kumara, and S. Venkatasubramanian
WWW7, 1998
<http://www7.scu.edu.au/programme/fullpapers/1938/com1938.htm>
- **[brin]**
The Anatomy of a Large-Scale Hypertextual Web Search Engine
S. Brin and L. Page
WWW7, 1998
<http://www7.scu.edu.au/programme/fullpapers/1921/com1921.htm>
- **[broder]**
Syntactic Clustering of the Web
Andrei Z. Broder, Steven C. Glassman, Mark S. Manasse
WWW6, 1997
- **[deerwester]**
Indexing by Latent Semantic Analysis
S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Landauer, R. Harshman
JASIS, 1990
<http://citeseer.nj.nec.com/deerwester90indexing.html>
- **[eckart]**
The approximation of one matrix by another of lower rank
C. Eckart, G. Young
Psychometrika, 1936
- **[faloutsos]**
Description and performance analysis of signature file methods
C. Faloutsos, S. Christodoulakis
ACM TOIS 1987
- **[fastpmc]**
FAST PMC - The Pattern Matching Chip
<http://www.fast.no/product/fastpmc.html>
<http://www.idi.ntnu.no/grupper/KS-grp/microarray/slides/heggebo.pdf>
- **[furnas]**
Information retrieval using a Singular Value Decomposition Model of Latent Semantic Structure
G.W. Furnas, S. Deerwester, S.T. Dumais, T.K. Landauer, R. A. Harshman, L.A. Streeter, and K.E. Lochbaum
ACM SIGIR, 1988
- **[glimpse]**
Glimpse, Webglimpse, Unix-based search software...
<http://webglimpse.org>

- **[gonnet]**
Examples of PAT applied to the Oxford English Dictionary
Gonnet G.
University of Waterloo, 1987
- **[harman]**
What we have learned, and not learned, from TREC
Donna Harman
<http://irsg.eu.org/irsg2000online/papers/harman.htm>
- **[joyce]**
The Thesaurus Approach to Information Retrieval
T. Joyce and R.M. Needham
American Documentation, 1958
- **[kleinberg]**
Authoritative Sources in a Hyperlinked Environment
Jon M. Kleinberg
JACM, 1998
<http://citeseer.nj.nec.com/87928.html>
- **[ilyinsky]**
An efficient method to detect duplicates of Web documents with the use of inverted index
S. Ilyinsky, M. Kuzmin, A. Melkov, I. Segalovich
WWW2002, 2002
- **[manber1990]**
Suffix Arrays: A New Method for On-line String Searches
U. Manber, G. Myers
1st ACM-SIAM Symposium on Discrete Algorithms, 1990
- **[manber1994]**
Finding similar files in a large file system
U. Manber
USENIX Conference, 1994
- **[maron]**
M.E. Maron and J.L. Kuhns
On relevance, probabilistic indexing and information retrieval
Journal of the ACM, 1960
- **[opentext]**
Open Text Corporation
<http://www.opentext.com>
- **[robertson]**
S.E. Robertson and Sparck Jones K.
Relevance Weighting of Search Terms
JASIS, 1976
- **[sedgewick]**
Algorithms in C++, Robert Sedgewick
Addison-Wesley, 1992
- **[spark-jones]**
A Statistical Interpretation of Term Specificity and Its Application in Retrieval

- Karen Sparck Jones
Journal of Documentation, 1972
- **[stata]**
The Term Vector Database: fast access to indexing terms for Web pages
R. Stata, K. Bharat, F. Maghoul
WWW9, 2000
<http://www9.org/w9cdrom/159/159.html>
 - **[strzalkowski]**
Natural Language Information Retrieval
Tomek Strzalkowski (ed.)
Kluwer Academic Publishers, 1999
 - **[кормен]**
Алгоритмы: построение и анализ, Т. Кормен, Ч. Лейзерсон, Р.Ривест
МЦНМО, 2000
<http://www.ozon.ru/?context=detail&id=114200>
 - **[симфония]**
Симфония или словарь-указатель к священному писанию ветхого и нового завета. Составители М.А. Бондарев, М.С.Косьян, С.Ю.Косьян
Изд-во Московской патриархии, 1995

Глоссарий

++ **ассессор** (assesor, эксперт) – специалист в предметной области, выносящий заключение о *релевантности* документа, найденного поисковой системой

++ **булевская модель** (boolean, булева, булевая, двоичная) – модель поиска, опирающаяся на операции пересечения, объединения и вычитания множеств

++ **векторная модель** – модель информационного поиска, рассматривающая документы и запросы как векторы в пространстве слов, а *релевантность* как расстояние между ними

++ **вероятностная модель** – модель информационного поиска, рассматривающая релевантность как вероятность соответствия данного документа запросу на основании вероятностей соответствия слов данного документа идеальному ответу

++ **внетекстовые критерии** (off-page, вне-страничные) – критерии ранжирования документов в поисковых системах, учитывающие факторы, не содержащиеся в тексте самого документа и не извлекаемые оттуда никаким образом

++ **входные страницы** (doorways, hallways) – страницы, созданные для искусственного повышения ранга в поисковых системах (*поискового спама*). При попадании на них пользователя перенаправляют на целевую страницу

++ **дизамбигуация** (tagging, part of speech disambiguation, таггинг) – выбор одного из нескольких *омонимов* с помощью контекста; в английском языке часто сводится к автоматическому назначению грамматической категории «часть речи»

++ **дубликаты** (duplicates) – разные документы с идентичным, с точки зрения пользователя, содержанием; **приблизительные дубликаты** (near duplicates, почти-дубликаты), в отличие от точных дубликатов, содержат незначительные отличия

++ **иллюзия свежести** – эффект кажущейся свежести, достигаемый поисковыми системами в интернете за счет более регулярного обхода тех документов, которые чаще находятся пользователями

++ **инвертированный файл** (inverted file, инверсный файл, инвертированный индекс, инвертированный список) – индекс поисковой системы, в котором перечислены слова коллекции документов, а для каждого слова перечислены все места, в которых оно встретилось

++ **индекс** (index, указатель) – см. *индексирование*

++ **индекс цитирования** (citation index) – число упоминаний (цитирований) научной статьи, в традиционной библиографической науке рассчитывается за промежуток времени, например, за год

++ **индексирование** (indexing, индексация) – процесс составления или приписывания указателя (*индекса*) – служебной структуры данных, необходимой для последующего поиска

++ **информационный поиск** (Information Retrieval, IR) – поиск неструктурированной информации, единицей представления которой является документ произвольных форматов. Предметом поиска выступает информационная потребность пользователя, неформально выраженная в поисковом запросе. И критерий поиска, и его результаты недетерминированы. Этими признаками информационный поиск отличается от «поиска данных», который оперирует набором формально заданных предикатов, имеет дело со структурированной информацией и чей результат всегда детерминирован. Теория *информационного поиска* изучает все составляющие процесса поиска, а именно, предварительную обработку текста (индексирование), обработку и исполнение запроса, ранжирование, пользовательский интерфейс и обратную связь.

++ **клоакинг** (cloaking) – техника *поискового спама*, состоящая в распознавании авторами документов робота (индексирующего агента) поисковой системы и генерации для него специального содержания, принципиально отличающегося от содержания, выдаваемого пользователю

++ **контрастность** термина – см. *различительная сила*

++ **латентно-семантическое индексирование** – запатентованный алгоритм *поиска по смыслу*, идентичный факторному анализу. Основан на сингулярном разложении матрицы связи слов с документами

++ **лемматизация** (lemmatization, нормализация) – приведение формы слова к словарному виду, то есть лемме

++ **накрутка** поисковых систем – см. *спам поисковых систем*

++ **непотизм** – вид *спама поисковых систем*, установка авторами документов взаимных ссылок с единственной целью поднять свой ранг в результатах поиска

++ **обратная встречаемость в документах** (inverted document frequency, **IDF**, обратная частота в документах, обратная документная частота) – показатель поисковой ценности слова (его *различительной силы*); *обратная* говорят, потому что при вычислении этого показателя в знаменателе дроби обычно стоит число документов, содержащих данное слово

++ **обратная связь** – отклик пользователей на результат поиска, их суждения о *релевантности* найденных документов, зафиксированные *поисковой системой* и использующиеся, например, для итеративной модификации запроса. Следует отличать от *псевдо-обратной связи* – техники модификации запроса, в которой несколько первых найденных документов автоматически считаются релевантными

++ **омонимия** – см. *полисемия*

- ++ **основа** – часть слова, общая для набора его *словообразовательных* и *словоизменительных* (чаще) форм
- ++ **поиск по смыслу** – алгоритм *информационного поиска*, способный находить документы, не содержащие слов запроса
- ++ **поиск похожих документов** (similar document search) – задача *информационного поиска*, в которой в качестве *запроса* выступает сам документ и необходимо найти документы, максимально напоминающие данный
- ++ **поисковая система** (search engine, SE, информационно-поисковая система, ИПС, поисковая машина, машина поиска, «поисковик», «искалка») – программа, предназначенная для поиска информации, обычно текстовых документов
- ++ **поисковое предписание** (query, запрос) – обычно строка текста
- ++ **полисемия** (polysemy, homography, многозначность, омография, *омонимия*) - наличие нескольких значений у одного и того же слова
- ++ **полнота** (recall, охват) – доля релевантного материала, заключенного в ответе поисковой системы, по отношению ко всему релевантному материалу в коллекции
- ++ **почти-дубликаты** (near-duplicates, приблизительные дубликаты) – см. *дубликаты*
- ++ **прюнинг** (pruning) – отсечение заведомо нерелевантных документов при поиске с целью ускорения выполнения *запроса*
- ++ **прямой поиск** – поиск непосредственно по тексту документов, без предварительной обработки (без *индексирования*)
- ++ **псевдо-обратная связь** – см. *обратная связь*
- ++ **различительная сила слова** (term specificity, term discriminating power, контрастность, различительная сила) – степень ширины или узости слова. Слишком широкие термины в поиске приносят слишком много информации, при это существенная часть ее бесполезна. Слишком узкие термины помогают найти слишком мало документов, хотя и более точных.
- ++ **регулярное выражение** (regular expression, pattern, «шаблон», реже «трафарет», «маска») – способ записи *поискового предписания*, позволяющий определять пожелания к искомому слову, его возможные написания, ошибки и т.д. В широком смысле – язык, позволяющий задавать *запросы* неограниченной сложности
- ++ **релевантность** (relevance, relevancy) – соответствие документа запросу
- ++ **сигнатура** (signature, подпись) – множество *хеш-значений* слов некоторого блока текста. При поиске по **методу сигнатур** все сигнатуры всех блоков

коллекции просматриваются последовательно в поисках совпадений с *хеш-значениями* слов запроса

++ **словоизменение** (inflection) – образование формы определенного грамматического значения, обычно обязательного в данном грамматическом контексте, принадлежащей к фиксированному набору форм (парадигме), характерного для слов данного типа. В отличие от *словообразования* никогда не приводит к смене типа и порождает предсказуемое значение. *Словоизменение* имен называют склонением (declension), а глаголов – спряжением (conjugation)

++ **словообразование** (derivation) – образование слова или основы из другого слова или основы. Чаще приводит к смене типа и к образованию слов, имеющих идеосинкразическое значение

++ **смыслоразличительный** – см. *различительная сила*

++ **спам поисковых систем** (spam, спамдексинг, накрутка поисковых систем) – попытка воздействовать на результат *информационного поиска* со стороны авторов документов

++ **статическая популярность** – см. *PageRank*

++ **стемминг** – процесс выделения основы слова

++ **стоп-слова (stop-words)** – те союзы, предлоги и другие частотные слова, которые данная поисковая система исключила из процесса индексирования и поиска для повышения своей производительности и/или *точности* поиска

++ **суффиксные деревья, суффиксные массивы (suffix trees, suffix arrays, PAT-arrays)** – *индекс*, основанный на представлении всех значимых *суффиксов* текста в структуре данных, известной как **бор** (trie). **Суффиксом** в этом индексе называю любую «подстроку», начинающуюся с некоторой позиции текста (текст рассматривается как одна непрерывная строка) и продолжающуюся до его конца. В реальных приложениях длина *суффиксов* ограничена, а индексируются только значимые позиции – например, начала слов. Этот индекс позволяет выполнять более сложные запросы, чем индекс, построенный на инвертированных файлах

++ **токенизация** (tokenization, lexical analysis, графематический анализ, лексический анализ) – выделение в тексте слов, чисел, и иных *токенов*, в том числе, например, нахождение границ предложений

++ **точность** (precision) - доля *релевантного* материала в ответе поисковой системы

++ **хеш-значение** (hash-value) – значение **хеш-функции** (hash-function), преобразующей данные произвольной длины (обычно, строку) в число фиксированного порядка

++ **частота (слова) в документах** (document frequency, встречаемость в документах, документная частота) – число документов в коллекции, содержащих данное слово

++ **частота термина** (term frequency, **TF**) – частота употреблений слова в документе

++ **шингл** – (shingle) – *хеш-значение* непрерывной последовательности слов текста фиксированной длины

++ **PageRank** – алгоритм расчета статической (глобальной) популярности страницы в интернете, назван в честь одного из авторов - Лоуренса Пейджа. Соответствует вероятности попадания пользователя на страницу в модели случайного блуждания

++ **TF*IDF** – численная мера соответствия слова и документа в *векторной модели*; тем больше, чем **относительно чаще** слово встретилось в документе и **относительно реже** в коллекции