Алгоритм текстового ранжирования Яндекса на РОМИП-2006

© Андрей Гулин, Михаил Маслов, Илья Сегалович

Яндекс
{gulin, maslov, iseg}@yandex-team.ru

Аннотация

В статье описан алгоритм ранжирования документов по релевантности запросу, использованный компанией Яндекс для участия в дорожках информационного поиска семинара РОМИП'2006. Кроме описания самого алгоритма рассматриваются полученные результаты, и обсуждается применимость описанного алгоритма для поиска по большому вебу.

1. Введение

Поисковые системы Интернета учитывают много факторов для определения релевантности страницы запросу пользователя. Одним из важнейших факторов является содержимое документа. Для улучшения этого аспекта ранжирования документов в Яндексе мы создали экспериментальную поисковую систему под кодовым названием ***atr***, описанную в данной статье. В результате проведения дорожек по поиску РОМИП в 2004 и 2005 годах накопилось значительное количество оцененных с точки зрения релевантности пар "запрос-документ". На основании этих данных мы составили набор факторов для вычисления оценки релевантности, которые реализовали в системе ***atr***.

2. Факторы, использованные в алгоритме ранжирования *atr*

Цель поисковой системы, рассматриваемой в данной статье – ранжирование документов (html страниц) коллекции по релевантности запросу. Для ранжирования использовался текст запроса, текст документа и некоторые элементы html-разметки документа.

Для каждого запроса мы вычисляем значение Score документа – показатель релевантности документа запросу, на основании которого и производится ранжирование. Сортировка выдачи поисковой системы по Score увеличивает average precision, precision at N и другие метрики качества поиска, основанные на оценке релевантности каждого документа по отдельности.

Для расчета Score была выбрана аддитивная модель. Интерес представляет вопрос какие слагаемые надо добавить в формулу расчета Score. В результате экспериментов были отобраны слагаемые за встречаемость слов из запроса в документе (W_{single}), за встречаемость пар слов из запроса в докумументе (W_{pair}) и за встречаемость текста запроса целиком (W_{Phrase}). Помимо этого есть два слагаемых, дающих преимущество за наличие всех слов запроса в документе ($W_{AllWords}$) и за наличие многих слов запроса в одном предложении ($W_{HalfPhrase}$).Итоговая формула выглядит следующим образом:

$$Score = W_{\sin gle} + W_{pair} + k_1 * W_{AllWords} + k_2 * W_{Phrase} + k_3 * W_{HalfPhrase} + W_{PRF} \quad (1)$$

Для улучшения результатов поиска был также использован подход “Pseudo-relevance feedback” [6]. Суть подхода заключается в том, что мы проводим поиск в два этапа. На первом этапе мы используем простой метод, описанный выше. После этого документы, найденные на первых позициях, мы объявляем релевантными, и ищем «похожие». Можно использовать любую меру похожести, мы использовали 2 разные меры, которые можно реализовать с достаточной для реальных применений производительностью.

Рассмотрим слагаемые более подробно.

2.1 Встречаемость слов в документе

Есть множество вариаций стандартного слагаемого за встречаемость слова в тексте, известного как подход TF*IDF. Мы опробовали на данных РОМИПа множество различных вариантов, начиная с классического TF * IDF, и заканчивая реже используемыми pl2, bm3 [7] и другими. Среди всех опробованных вариантов оптимальной оказалась модификация bm25 [5]:

$$W_{single} = \log(p) * \frac{TF}{TF + k_1 + k_2 * DocLength}, \quad (2)$$

$$k_1 = 1, k_2 = 1/350$$

В этой формуле TF – число вхождений леммы в документ, DocLength – длина документа в словах. При подсчете количества вхождений слова в документ мы проводим предварительную лемматизацию слов запроса и слов документа. Результат поиска без лемматизации существенно уступает варианту с лемматизацией.

Использование в качестве меры длины документа максимальной TF среди всех лемм документа ухудшает результат. Попытки нелинейных преобразований DocLength и поиска наилучших поддокументов (фрагментов) в большом документе, как описано в [1], не привели к улучшению результата.

В качестве p были опробованы:

$$IDF = \frac{D}{DF},$$

$$ICF = \frac{TotalLemms}{CF},$$

$$rIDF = -\log(\frac{DF}{D}) + \log(1 - \exp(-\frac{CF}{D})) \quad \text{//-- residual idf.}$$

Здесь D – число документов в коллекции, DF – количество документов, в которых встречается лемма, CF – число вхождений леммы в коллекцию, TotalLemms – общее число вхождений всех лемм в коллекции. Из этих вариантов лучший результат в наших эксперимента показал ICF.

Но этот результат нам удалось улучшить.

Улучшение связано с гипотезой [2] о том, что распределение количества слов в документах хорошо апроксимируется смесью двух распределений Пуассона. Согласно этой гипотезе для каждого слова есть 2 вида документов – те, которые “про это слово”, и остальные, в которых слово встречается случайно. Для каждого слова мы таким образом получаем 3 параметра распределения: f_1 – вероятность встретить слово случайно, f_2 – вероятность встретить слово в документе “про это слово” и $p_{OnTopic}$ – вероятность того, что документ “про это слово”.

Нами была проведена экспериментальная проверка этой гипотезы на коллекции РОМИПа. Для всех слов из запросов были вычислены их оптимальные f_1, f_2 и $p_{OnTopic}$. Проверка показала, что использование этой модели описывает распределение слов по документам значительно лучше, чем простой Пуассон или равномерное распределение. Оптимальные f_1, f_2 и $p_{OnTopic}$ для разных слов, естественно, различаются. Они могут быть достаточно точно аппроксимированы так:

$$f_1 \approx 0.003$$
$$f_2 \approx 0.5\frac{CF}{TotalLemms} \qquad (3)$$
$$p_{OnTopic} \approx 1 - \exp(-1.5 * \frac{CF}{D})$$

Здесь CF – число вхождений леммы в коллекцию, TotalLemms – сколько всего лемм в коллекции, D – число документов в коллекции. Формулы оценки релевантности, основанные на этой модели напрямую, использующие вычисление вероятности того, что "документ про это слово" не дали хороших результатов. Однако, использование $p_{OnTopic}$ в нашей модификакции bm25 в качестве p заметно улучшает качество и использовалось в финальной версии алгоритма.

Помимо учета количества слов в документе можно учитывать html-форматирование и позицию слова в документе. Мы учитываем это в виде отдельного слагаемого. Учитывается наличие слова в первом предложении, во втором предложении, внутри выделяющих html тегов. Итоговая формула выглядит так:

$$W_{single} = \log(p) * (TF_1 + 0.2 * TF_2)$$
$$TF_1 = \frac{TF}{TF + k_1 + k_2 * DocLength}, k_1 = 1, k_2 = 1/350 \qquad (4)$$
$$TF_2 = \frac{Hdr}{1 + Hdr}$$
$$p = 1 - \exp(-1.5 * \frac{CF}{D})$$

Здесь Hdr – сумма весов слова за форматирование.

2.2 Учет пар слов

Помимо вхождения в текст одиночных слов запроса в описываемом алгоритме учитывается вхождение пар слов запроса в документ. После экспериментов с различными способами учета мы выделили четыре варианта учета пар и определили соответствующие веса. За разные случаи дается разный вес. Пара учитывается, когда слова запроса встречаются в тексте подряд (+1), через слово (+0.5) или в обратном порядке (+0.5). Плюс еще специальный случай, когда слова, идущие в запросе через одно, в тексте встречаются подряд (+0.1). Вес за пары вычисляется по такой формуле:

$$W_{pair} = 0.3 * (\log(p_1) + \log(p_2)) * \frac{TF}{1 + TF} \qquad (5)$$

p_1 и p_2 здесь - p для первого и второго слова пары из слагаемого $W_{\sin gle}$. TF – количество вхождений пары в текст с учетом весов вхождений. Учет длины документа и форматирования для пар слов не дали выигрыша, поэтому они не учитываются.

Учет встречаемости трех и более слов запроса в документе уличшений в наших экспериментах не дал.

2.3 Учет всех слов запроса в документ, учет фраз

Важным фактором помимо перечисленных является наличие в документе всех слов запроса. За наличие всех слов запроса мы добавляем дополнительный «бонус» $W_{AllWords}$, пропорциональный сумме idf слов запроса:

$$W_{AllWords} = 0.2 * \sum \log(p_i)$$

Если в документе присутствуют не все слова, то за каждое отсутствующее слово $W_{AllWords}$ домножается на коэффициэнт 0.03. Т.е. полная формула выглядит так:

$$W_{AllWords} = 0.2 * \sum \log(p_i) * 0.03^{N_{miss}}$$

где N_{miss} – количество отсутствующих в документе слов запроса.

Если в документе отсутствует много слов запроса, то возникает проблема. С точки зрения эффективности поиска выгодно удалять документы, в которых нет хотя бы половины (считая по сумме idf) слов запроса. С точки зрения полноты поиска мы должны вернуть как можно больше возможно релевантных документов. Следовательно, удалять документы нельзя. Для РОМИПа мы подготовили два разных прогона с удалением документов и без такового. Вариант

с удалением документов достигает слегка большего average precision, возможно, за счет того, что наличие половины слов запроса в документе никак не учитывается в варианте без удаления.

Помимо наличия слов запроса в документе мы можем учесть наличие в документе текста запроса целиком, это слагаемое называется W_{Phrase} TF в данном случае – количество вхождений запроса в текст документа

$$W_{Phrase} = 0.1 * \sum \log(p_i) * \frac{TF}{1+TF}$$

Плюс к этому еще небольшой «бонус» $W_{HalfPhrase}$ дается за наличие в тексте предложений, содержащих значительное количество слов запроса. «Значительное» в данном случае означает, что сумма idf слов запроса в предложении больше половины суммы idf всех слов запроса. TF здесь – количество таких предложений в тексте.

$$W_{Phrase} = 0.02 * \sum \log(p_i) * \frac{TF}{1+TF}$$

2.4 Pseudo-relevance feedback

Для дальнейшего улучшения результатов поиска нами был использован метод PRF (pseudo-relevance feedback) [6].

Метод relevance feedback без приставки «псевдо-» заключается в том, что поиск ведется в две стадии. На первой находятся какие-то документы, которые предъявляются пользователю. Пользователь помечает документы, релевантные запросу. Второй этап поиска использует эти оценки.

Традиционный метод использования relevance feedback – добавить в запрос новые слова, встречающиеся в документах, помеченных как релевантные. Такой способ сложен в практическом использовании, так как требует повторного исполнения всего запроса, причем более сложного, чем изначальный. Мы использовали другой способ – мы даем бонус документам, похожим на помеченные экспертом. Мера похожести может быть любой. Нами была использована мера похожести, основанная на тегах, которые мы присваивали каждому документу.

Метод relevance feedback можно применять и без участия пользователя, если предположить, что наша система достаточно хороша и на первых позициях находит релевантные документы. Тогда мы просто объявляем первые N документов релевантными и повышаем

ранг документов, похожих на них. Мы объявляем, что степень релевантности зависит от позиции документа в выдаче:

$$R_i = \exp(k_1 * i)$$

Для расчета похожести нам нужны какие-то теги – признаки документов, по которым мы будем определять похожесть. Для РОМИПа мы использовали два набора тегов.

Первый набор тегов – автоматическая классификация документов по темам Яндекс.Каталога. Для классификации мы использовали алгоритм Байеса в интерпретации Пола Грэма [3] Коллекция документов, на которой настраивался автомат, сформирована по принципам, описанным в [4]. Каждому документу автомат приписывает одну тему. Точность алгоритма – 63%, полнота – 46%, F1 – 54% (величины – микроусредненные, измерены по рубрикам 2-го уровня Яндекс.Каталога) . В результате PRF дополнительный бонус получают документы той же темы, что и первые документы выдачи.

Второй набор тегов использует слова, встречающиеся в документе. Идея метода заключается в том, что некоторые группы слов часто встречаются вместе. Найдя такие группы, мы можем назначить им теги. После этого каждому документу мы можем назначить тег, если в документе встречается много слов из группы этого тега. Для построения таких групп был использован принцип минимальной длины описания (Minimal Description Length, MDL).

Возьмем матрицу, по строкам которой расположены документы, по столбцам слова. Запишем 1 в пересечение, если в документе встречается это слово и 0 в обратном случае. Дальше нам нужно построить максимально компактное описание этой матрицы с помощью тегов (групп слов). Используем описание в следующем виде. Для каждого тега имеем список слов, для каждого документа – список тегов этого документа и список “поправочных” слов. Объединение слов тегов документа дает нам множество “предсказанных” слов для документа. “Поправочные” слова – это слова, которые есть в документе, но их нет в “предсказанных” или, наоборот, слова, которые есть в “предсказанных”, но отсутствуют в документе. Подберем оптимальное с точки зрения количества информации описание исходной матрицы. В результате получим группы слов и списки документов, в которых используются эти группы слов. Для РОМИПа мы взяли ~50 тысяч документов и ~15 тысяч самых частотных слов. В этих данных были найдены ~800 групп слов. Эти группы и были использованы в качестве второго набора тегов. В результате PRF со вторым набором тегов бонус получают документы, использующие сходную с лидерами лексику.

Для определения схожести двух документов по тегам используется взвешенное по idf тега скалярное произведение:

$$tagW_i = \sum R_{pos[k]} * tagged_{k,i}$$

$$Similar_k = \frac{\sum tagW_i * tagged_{k,i} * (idf_i)^2}{\sqrt{\left[\sum (tagW_i)^2 * (idf_i)^2\right] * \left[\sum tagged_{k,i} * (idf_i)^2\right]}}$$

$$idf_i = \frac{D_i}{D}$$

Pos[k] – позиция k-го документа в выдаче первого прохода, $tagged_{k,i}$ = 1, если у k-го документа есть тег i и = 0 в противном случае, D_i - количество документов с i-м тегом, D – количество документов в коллекции.

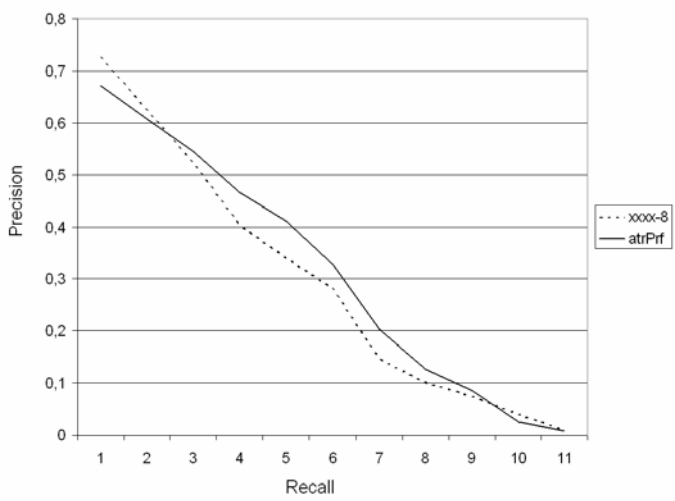
В качестве бонуса для k-го документа добавляется $Similar_k * k_2 * \sum \log(p_i)$. Для первого прохода с тегами – классификацией по каталогу $k_1 = -0.3, k_2 = 0.25$ Для второго прохода с тегами по наборам слов $k_1 = -0.1, k_2 = 0.1$.

3. Описание экспериментов

Мы приняли участие в двух дорожках – поиску по веб-коллекции и по коллекции нормативных документов.

Для прогона по веб-коллекции мы заявили 6 прогонов, три с использованием стандартного алгоритма Яндекса и три с использованием ***atr***, описанного в этой статье. 3 прогона алгоритма ***atr*** отличались использованием prf и способом отсечения вероятно нерелевантных документов. Прогоны: простой (***altTextRef***), с PRF (***altTextRefPRF***) и с PRF и отсечением документов без половины слов запроса (***altTextRefPRF2***).

Был получен результат, говорящий, видимо, о том, что описанная система способна отличать релевантные документы от нерелевантных, но существенно хуже различает сильно релевантные документы от слаборелевантных.



На графике представлено сравнение прогона ***atrPrf*** (сплошная линия) с прогоном №8 (пунктиром) на дорожке web-adhoc по запросам только 2006-го года с релевантностью, считаемой по Or. График – 11-точечный Precision-Recall, построенный по методике РОМИПа, по горизонтали – уровни recall, по вертикали – достигнутый precision на этом уровне. Видно, что участник номер 8 опережает нас в области самых релевантных документов, но алгоритм ***atr*** заметно обгоняет в средней области. Остальные графики демонстрируют аналогичное поведение сравниваемых систем.

Для более детального сравнения систем мы можем разделить запросы на группы. Для дальнейших сравнений мы будем использовать данные по 50 запросам 2006 года с релевантностью, считаемой по and. Первое очевидное разделение – по количеству слов в запросе:

	atrPrf2	участник №8
2 слова	21,1%	20,8%
3 слова	27,5%	27,5%
>3 слов	27,3%	30,0%

В таблице указано среднее значение average precision для групп запросов. Видно, что участник номер 8 в среднем обрабатывает длинные запросы лучше, чем ***atr***..

Второй интересный способ разделения – по размеру пула. Подразумевается, что запросы с большим пулом сложнее для систем, чем запросы, в которых ответы разных систем сильно пересекаются.

Размер пула	***atrPrf2***	участник №8
Простые (<200 документов)	22,6%	20,8%
Средние	29,5%	26,5%
Сложные(>275 документов)	21,9%	29,4%

Как видно из таблицы, atr отрабатывает “сложные” запросы заметно хуже системы №8. Существенную лепту в цифры в этих таблицах вносит единственный запрос “Коммунистическая партия СССР”, отработанный системой №8 особенно удачно по сравнению с другими участниками. Но и без учета этого запроса наблюдаемые закономерности сохраняются.

Специальной настройки под legal-коллекцию мы не проводили и PRF для legal-коллекции не использовали. Тем не менее, достигнутый результат по legal-коллекции averge precision 33.2%, первое место из прнимавших участие, позволяет говорить о достаточно широкой применимости найденных закономерностей.

4. Применимость atr для поиска по веб

В этом разделе мы обсудим результат попытки использования ***atr*** для поиска по всему вебу и какие проблемы были встречены в ходе этой попытки.

4.1 Объем данных

Объем данных в Интернете в тысячи раз больше, чем в коллекции РОМИП, поэтому, даже при незначительных изменениях способа расчета Score, находится масса новых неоцененных документов. Неоцененные документы плохи тем, что при любом способе их учета (как нерелевантных, релевантных как соседи и т.п.) они резко усложняют подбор формы и коэффициентов формулы для расчета Score в силу появляющейся неопределенности достигнутого результата.

4.2 Тип данных

Как известно, коллекция документов РОМИПа сформирована на основе сайтов с бесплатного хостинга Narod.ru.

На содержимое Narod.ru влияют следующие обстоятельства:

1) Narod.ru не поддерживает динамические страницы; например, в поисковой базе Яндекса в момент написания этой статьи таких страниц более половины.
2) Пользовательское соглашение существенно ограничивает размещение коммерческой информации на Narod.ru.
3) Администрация Narod.ru модерирует сайты, в частности, активно удаляет поисковый спам.
4) Администрация Narod.ru, так же как и других «больших» хостингов, существенно ограничивает возможность для обхода роботам поисковых систем.

Поэтому, если сравнивать с «рунетом», то в коллекции РОМИПа плохо представлены ресурсы многих типов. В частности:

1) Практически нет интернет-магазинов, которые, как правило, динамические.
2) Нет больших справочных сайтов и сайтов-«баз данных», которые, также, как правило, динамические.
3) Нет досок объявлений, форумов и блогов, которые, практически всегда являются динамическими; к тому же блогов в момент формирования коллекции не было вообще.
4) Мало сайтов организаций (корпоративных сайтов и т.п.), а те, что есть – как правило маленькие сайты-«визитки».

4.3 Отсутствие спама и оптимизированных сайтов в тестовой коллекции

Владельцы коммерческих сайтов - корпоративных, интернет-магазинов и т.п. активно адаптируют тексты своих страниц так, чтобы поисковые системы находили их в вершине списка

Многие форумы и каталоги ресурсов в интернете слабо или вообще не модерируются. Пользуясь этим, владельцы сайтов размещают в них рекламные объявления и ссылки на свои ресурсы.

Многие владельцы бесплатных хостингов, в отличие от Narod.ru, не удаляют у себя поисковый спам.

Все вышеперечисленные продукты деятельности поисковых оптимизаторов практически не представлены в коллекции РОМИПа.

4.4 Запросы

Небольшое сравнение коллекции оцененных запросов РОМИП с запросами, задаваемыми поисковой системе Яндекс, показывает, что их характеристики довольно заметно различаются:

Источник	1-словн	2-х-словн	3-х-словн	>3-словн	“Нерусскоязычные” запросы
Веб-запросы	22.2%	27.0%	21.7%	29.1%	16.8%
РОМИП-2006	0%	41.4%	37.1%	21.4%	0%

Колонки таблицы соответствуют доле запросов со следующими характеристиками: однословные запросы, двухсловные, трехсловные, запросы с количеством слов более трех и “нерусскоязычные”, т.е. запросы, в которых нет русских букв.

Как видно, наибольшие отличия веб-запросов от запросов РОМИП состоят в том, что в последней полностью отсутствуют однословные и “нерусскоязычные” запросы.

Литература

[1] *Федоровский А.Н, Костин М.Ю.* Mail.ru на РОМИП-2005. // в сб. "Труды РОМИП'2005" Труды третьего российского семинара по оценке методов информационного поиска. Под ред. И.С. Некрестьянова, стр. 106-124, Санкт-Петербург: НИИ Химии СПбГУ, 2005

[2] *Kenneth W. Church, William A. Gale.* Poisson Mixtures, 1995. http://citeseer.ist.psu.edu/church95poisson.html

[3] *P. Graham.* Better Bayesian Filtering, Spam Conference, 2003. http://www.paulgraham.com/better.html

[4] *И. Сегалович, М. Маслов, Ю. Зеленков.* Цели и результаты программы научных стипендий Яндекса, пункт "Набор данных "Классификация сайтов" ", 2005. http://company.yandex.ru/grant/2005/00_YANDEX.pdf

[5] *S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford.* Okapi at trec-3. // In Proc. of the TREC-3, 1994

[6] *Jinxi Xu and W. Bruce Croft.* Query expansion using local and global document analysis. In Proc. of the SIGIR’96, pp. 4-11, 1996.

[7] *Ben He and Iadh Ounis.* A study of the dirichlet priors for term frequency normalisation. In Proc. of the SIGIR’2005, pp. 465-471, 2005