

Текстовое ранжирование. Часть II. Сводим всё воедино

(основано на слайдах Андрея Калинина, Hinrich Schütze,
Christina Lioma)

Содержание занятия

1. Ускоряем векторное ранжирование
2. Строим полноценную поисковую систему
 - а. Потребуется некоторое количество ухищрений и эвристик

Вычисление весов-косинусов



COSINESCORE(q)

```
1  float Scores[ $N$ ] = 0
2  float Length[ $N$ ]
3  for each query term  $t$ 
4  do calculate  $w_{t,q}$  and fetch postings list for  $t$ 
5      for each pair( $d, tf_{t,d}$ ) in postings list
6      do Scores[ $d$ ] +=  $w_{t,d} \times w_{t,q}$ 
7  Read the array Length
8  for each  $d$ 
9  do Scores[ $d$ ] = Scores[ $d$ ]/Length[ $d$ ]
10 return Top  $K$  components of Scores[]
```



- Нужно найти K «ближайших» к запросу документов $\Rightarrow k$ самых больших косинусов между запросом и документом.
- Оптимизация ранжирования:
 - Быстрое вычисление одного косинуса.
 - Быстрый выбор k наибольших косинусов.
 - Можно ли при этом не вычислять все N косинусов?



- Что мы делаем: решаем задачу поиска k -ближайших соседей для вектора запроса
- Вообще, мы не знаем, как эффективно решать эту задачу для пространств большой размерности.
- Но есть методы для коротких запросов, совместимые с обычными индексами.

Особый случай – запросы без весов



- Не будем взвешивать термины запроса
 - Предположим, что каждый термин в запросе встречается только один раз
- Тогда не требуется нормализация запроса
 - Небольшое упрощение алгоритма из 7-й лекции

Ускоряем: запрос без взвешивания



FASTCOSINESCORE(q)

- 1 float $Scores[N] = 0$
- 2 **for each** d
- 3 **do** Initialize $Length[d]$ to the length of doc d
- 4 **for each** query term t
- 5 **do** calculate $w_{t,q}$ and fetch postings list for t
- 6 **for each** pair($d, tf_{t,d}$) in postings list
- 7 **do** add $wf_{t,d}$ to $Scores[d]$
- 8 Read the array $Length[d]$
- 9 **for each** d
- 10 **do** Divide $Scores[d]$ by $Length[d]$
- 11 **return** Top K components of $Scores[]$

Figure 7.1 A faster algorithm for vector space scores.

Вычисление k наибольших косинусов: выбор или сортировка?

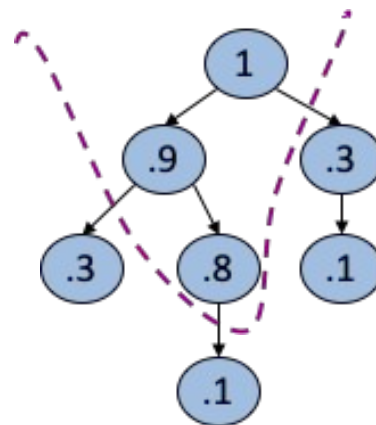


- Обычно мы хотим получить только k верхних документов
 - а не отсортировать все документы в корпусе
- Можем ли мы выбрать K документов с наибольшими значениями косинусов?
- Пусть J = количество документов с ненулевыми косинусами
 - Мы ищем k лучших из J документов

Используем кучу для выбора K максимальных



- Бинарное дерево, где значение в каждой вершине больше значений детей
- Требуется $2J$ операций для создания, затем каждый документ из K -верхушки добывается за $2\log J$ шагов.
- Для $J = 1\text{M}$, $K = 100$, это где-то 10% от стоимости сортировки.





- Главная вычислительная проблема во взвешивании – вычисление косинусов.
- Можно ли избежать этих вычислений?
- Да, но иногда недостаточно точно
 - документ не из К-верхушки может пролезть в окончательный список К документов
 - Действительно ли это так страшно?

Косинусы лишь приближение



- У пользователя есть задача и он сформулировал запрос
- Косинусы позволяют найти документы для этого запроса
- Косинус – приближение удовлетворённости пользователя
- Если мы получим список K документов, достаточно близкий к K -верхушке «по косинусам», то всё должно быть в порядке



- Найдём множество A соискателей такое, что $K < |A| \ll N$
 - A может не содержать все документы из верхушки K , но должно иметь достаточное количество
 - Вернём верхние K документов из A
- **Английский термин – pruning**
- Аналогичный подход используется и для других функций взвешивания
- **Рассмотрим несколько реализаций этого подхода**

Убираем ненужные документы



- Базовый алгоритм FastCosineScore рассматривает только документы, содержащие хотя бы один термин запроса
- Пойдём дальше:
 - Будем рассматривать только термины с высокими значениями idf
 - Отсечём документы с небольшим количеством терминов запроса

Термины с большим IDF



- Для запроса [catcher in the rye]
- Будем учитывать веса только для catcher и rye
- Идея: in и the не дают большого вклада в итоговый вес, т.е. не могут сильно повлиять на итоговый результат.
- Выгода:
 - Координатные блоки терминов с небольшим IDF будут иметь много документов → они пропадут из списка A

Документы с большим количеством терминов из запроса



- Любой документ, содержащий хотя бы один термин из запроса, является кандидатом для результирующего списка
- Для длинных запросов будем отбирать документы, содержащие несколько терминов запроса
 - Например, 3 из 4-х
 - «Мягкая конъюнкция» веб-поиска
- Просто реализовать при проходе по координатным блокам

3 из 4-х терминов



Antony	→	3	4	8	16	32	64	128	
Brutus	→	2	4	8	16	32	64	128	
Caesar	→	1	2	3	5	8	13	21	34
Calpurnia	→	13	16	32					

- Веса вычисляются для документов 8, 16 и 32.



- Предвычислим для каждого термина t r документов с самыми высокими весами среди координат t
 - Английское название: champion list
 - (или fancy list)
- Значение r выбирается при построении индекса
 - Можно выбрать $r < K$
- При выполнении запроса вычисляем веса только для лучших документов какого-то из терминов запроса



- Можно ли использовать списки лучших документов с уже рассмотренными способами? Есть ли в этом необходимость?
- Как эти списки могут быть реализованы в обратном индексе?
 - Списки лучших никак не соотносятся с docID



- Мы хотим, чтобы документы наверху выдачи были бы релевантными и авторитетными
- Релевантность сейчас оценивается косинусами
- Авторитетность не зависит от запроса
- Примеры авторитетных документов:
 - Википедия среди веб-сайтов
 - Статьи в некоторых газетах или информационных агентствах
 - Научная статья с большим количеством ссылок на неё
 - Много социальных «рекомендаций», упоминаний в твиттерах и т.п.
 - Pagerank

Модель авторитетности



- Назначим каждому документу d запросо-независимый вес качества из диапазона $[0,1]$
 - Обозначим его как $g(d)$
- Например, количество ссылок нормированное в $[0,1]$
 - Вопрос: по какой формуле?



- Рассмотрим простой вес, учитывающий релевантности «по косинусам» и авторитетность
- $\text{net-score}(q,d) = g(d) + \text{cosine}(q,d)$
 - Можем использовать другую линейную комбинацию
 - Или любую функцию от двух факторов (сигналов), моделирующую удовлетворённость или счастье пользователя
- Теперь мы ищем K-верхушку по общему весу

Быстрый способ получить K-верхушку по общему весу



- Идея: упорядочить координаты по $g(d)$
- Часто так и делают для всех координат
- Можем одновременно проходить термины запроса для
 - Пересечения координат
 - Вычислению веса «по косинусам»
- Упражнение: Напишите псевдокод

Зачем упорядочивать по $g(d)$?



- В этом случае документы с большими весами имеют высокую вероятность оказаться в начале списков
- При ограничениях на время выполнения мы можем остановить время выполнения по достижению лимита времени или количества найденных документов
 - Тем самым, не считаем косинусы для всех документов из координатных блоков

Списки лучших документов и упорядочение по авторитетности



- Можно объединить два подхода
- Подготовим для каждого термина список лучших документов с высшими значениями $g(d) + \text{tf-idf}_{td}$
- Найдём верхние K результатов среди документов из этих списков



- Для каждого термина построим два координатных блока, «золотой» и «серебряный»
 - Золотой – список лучших документов
- При обработке терминов запроса сначала проходим только по золотым блокам
 - Если в результате получим больше K документов, то заканчиваем выполнение запроса
 - Иначе переходим к серебряным блокам
- Можно использовать и без $g(d)$
- Тем самым, делим индекс на два эшелона

Упорядочение координат по весам



- Мы можем вычислить веса только для тех документов, где $wf_{t,d}$ достаточно высок
- Сортируем все координаты по $wf_{t,d}$
- Но теперь: все координаты в расположены в разном порядке!
- Как вычислить веса для выбора K-верхушки?
 - Есть пара идей

1. Раннее прекращение запроса



- При обходе координат t , остановим выполнение запроса как только
 - Обработали или получили r документов
 - $wf_{t,d}$ стал меньше некоторого порога
- Объединим все документы для каждого термина
- Вычислим веса только для этого набора документов

2. Термины в IDF-порядке



- При обработке координат для терминов...
- ... будем загружать их в порядке убывания IDF
 - Термины с высоким IDF дадут больший вклад в итоговый вес
- При обновлении весов для терминов остановимся тогда, когда веса перестают значимо меняться
- Применимо не только для «косинусов»

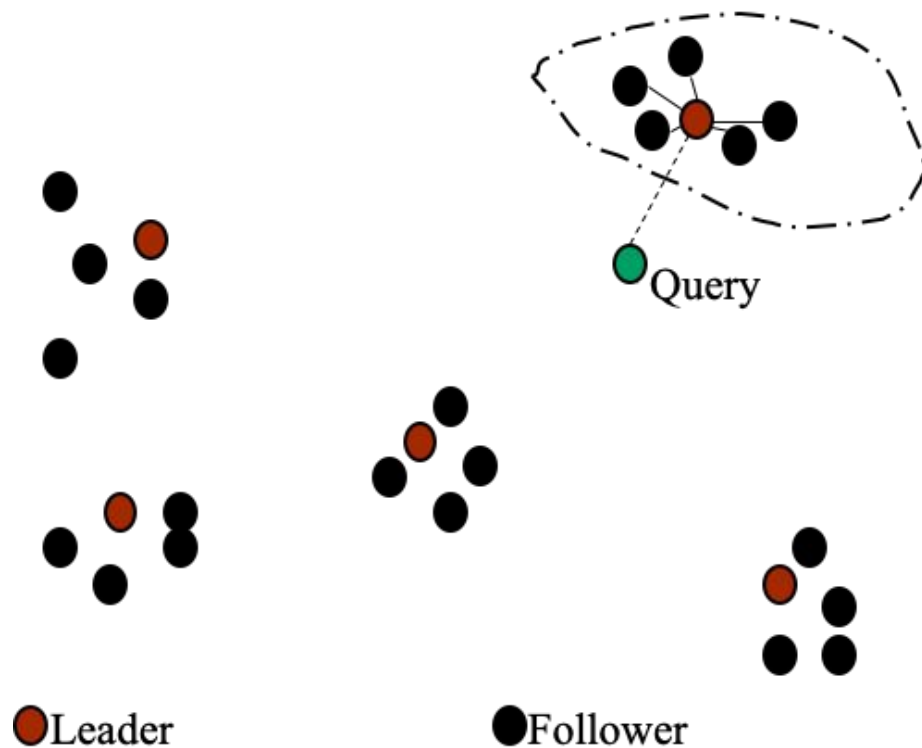


- Выберем случайны \sqrt{N} документов: *лидеры*
- Для остальных документов найдём ближайшего лидера
 - Документы, отнесённые к лидеру – его *последователи*;
 - Вероятно: каждый лидер будет иметь $\sim \sqrt{N}$ последователей.



- Выполняем запрос так:
 - Для запроса Q найдём ближайшего *лидера* L .
 - Затем найдём K ближайших документов среди последователей L .

Примерно так:



Почему случайный выбор?



- Быстро
- Лидеры отражают распределение данных в корпусе



- Каждый последователь присоединяется к, например, $b_1=3$ ближайшим лидерам.
- Для запроса найдём, например, $b_2=4$ ближайших лидеров и их последователей.



- Сколько нужно вычислить косинусов, чтобы найти ближайшего лидера на шаге 1?
 - Откуда взялось число \sqrt{N} ?
- Каков эффект от констант b_1, b_2 с предыдущего слайда?
- Приведите пример, когда этот подход скорее всего будет неудачным – т.е., мы пропустим один из K документов.
 - Скорее всего – при условии случайного выбора лидеров.



- До сих пор документ представлялся последовательностью терминов
- Обычно документы состоят из нескольких частей, с определённой семантикой:
 - Автор
 - Заголовок
 - Дата публикации
 - Язык
 - Формат
 - И т.п.
- Всё это – метаданные

Поля (числовые зоны)

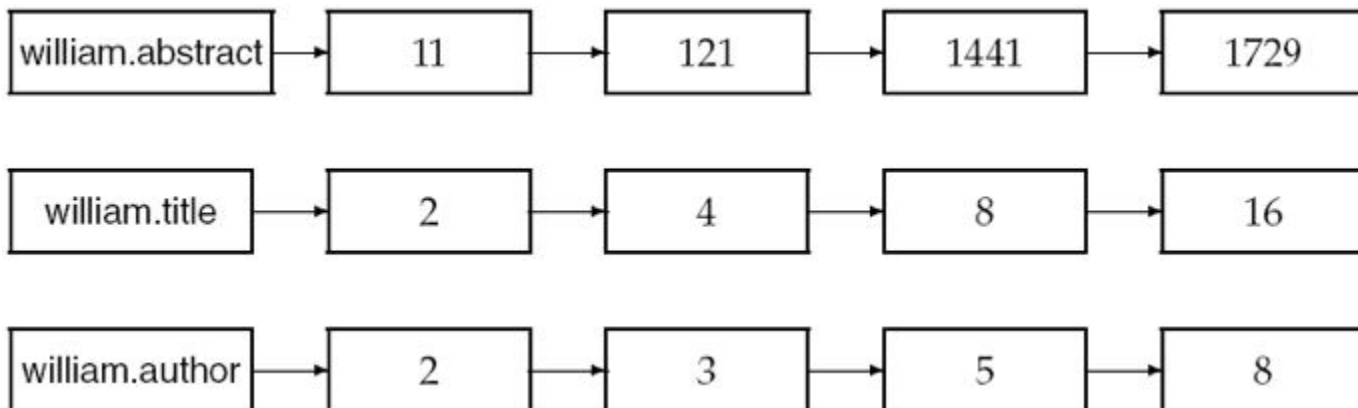


- Иногда мы хотим искать в метаданных
 - Например, найти документа за авторством Шекспира, написанные в 1601-м году и содержащие цитату `alas poor Yorick`
- **Year = 1601 это пример поля**
- Так же, `author last name = shakespeare` и т.п.
- **Для хранения этих данных строится специальный индекс (параметрический): координаты для каждого значения поля**
 - **Иногда более сложные структуры, например для диапазонов**
- Запросы к полям обычно конъюнктивные
 - (документ должен быть за авторством `shakespeare`)

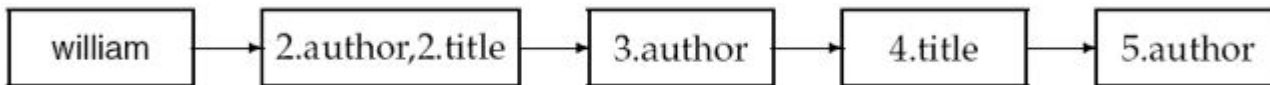


- Зона это часть документа, содержащая какой-то особенный текст, например,
 - Заголовок
 - Описание
 - Ссылки ...
- Для зон так же строится обратный индекс
- Теперь можно выполнять запросы вида «найди документы с merchant в заголовке и удовлетворяющие запросу gentle rain»

Индексы для зон



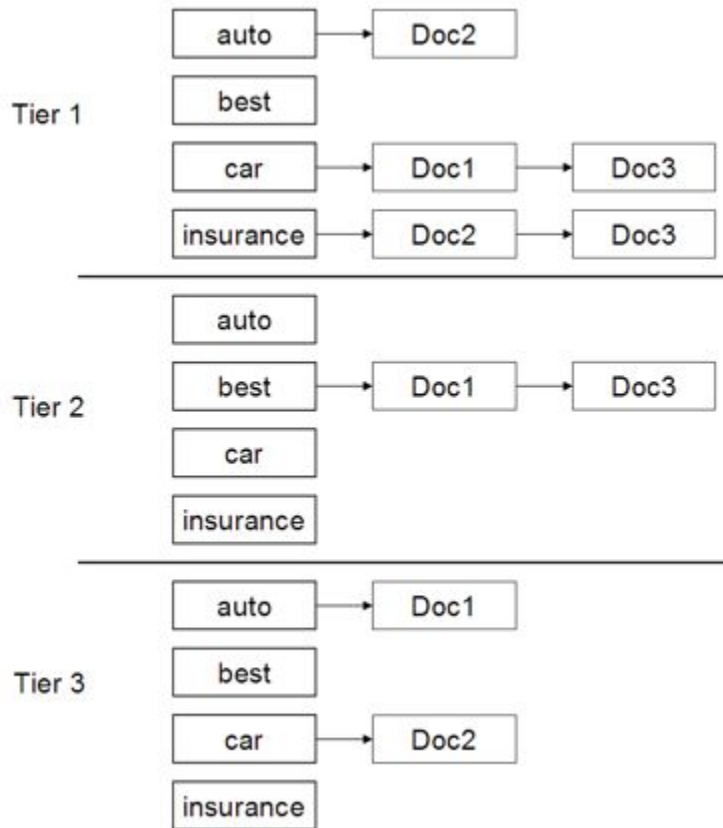
Можно хранить зоны в словаре или в координатах





- Делим координаты по их важности
 - Самые важные
 - ...
 - Менее важные
- Можем оценивать по $g(d)$
- Тем самым индекс поделён на эшелоны по убыванию важности
- Во время выполнения запроса берём первый эшелон, если в нём получилось найти K документов
 - Иначе переходим к следующим эшелонам

Пример эшелонирования





- Текстовые запросы: набор терминов, введённых в поисковую строку
- Пользователи предпочитают документы, где термины запроса находятся на небольшом расстоянии друг относительно друга
- Пусть w будет наименьшим окном в документе, содержащим все термины запроса. Например,
- Для запроса [strained mercy] такое окно в документе The quality of mercy is not strained равно 4 (в словах)
- Хотим включить это в итоговый вес – как?

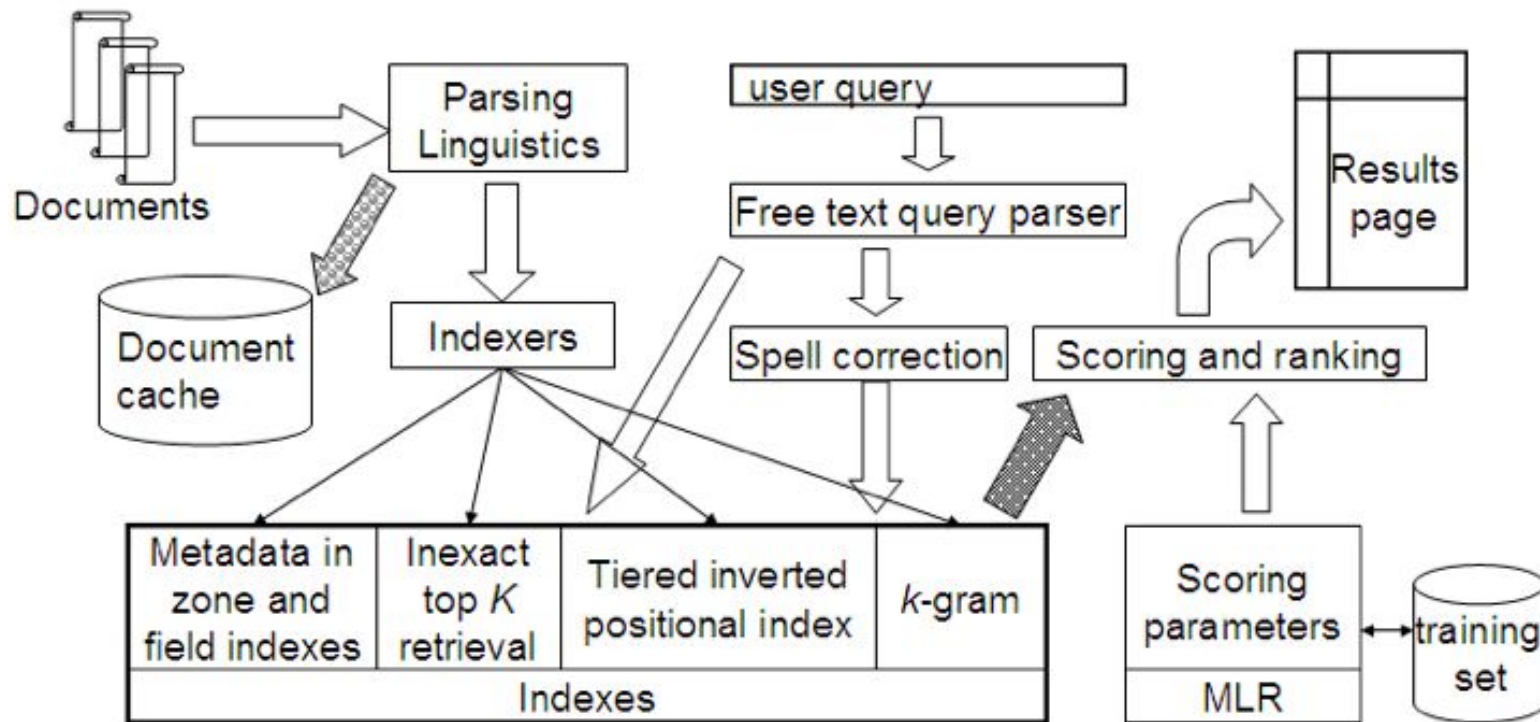


- Парсер запроса может подготовить по одному запросу несколько фактических запросов к индексу, например **rising interest rates**
 - Обработать как цитатный
 - Если нашли <K документов с цитатой **rising interest rates**, обработать как два цитатных запроса **rising interest** и **interest rates**
 - Если всё ещё <K документов, обработать как векторный запрос **rising interest rates**
 - Отсортировать документы по весу в векторном пространстве
- Такую последовательность операций может сгенерировать парсер запроса.



- Мы уже построили функции весов, объединяющие в себе косинусы, авторитетность, компактность и т.п.
- Как найти лучшую такую функцию?
- Иногда – вручную
- В последнее время – машинное обучение
 - Ещё поговорим об этом (а может и нет)

Соединяем всё вместе



Введение в информационный поиск
I Маннинг Кристофер Д., Шютце
Хайнрих

Рекомендуемая
литература

Для саморазвития (опционально)
Чтобы не набирать двумя
пальчиками



Спасибо за
внимание!

Антон Кухтичев



a.kukhtichev@mail.ru



[@toshunster](https://t.me/toshunster)