Словосочетания, веса в реальной поисковой системе

Введение в информационный поиск, глава 7.

Обработка словосочетаний

- До сих пор:
  - Только пословные индексы
  - И векторные модели

Фразовые запросы

- Предположим мы хотим найти ответ на запрос «Stanford University» (в кавычках)
  - Фразовые запросы
  - Много запросов пользователей внутри себя содержат фразовые запросы
  - Не достаточно хранить только матрицу (индекс): терм-документ
- Проблема
  - Тогда предложение “I went to university of Stanford” не подходит

Подходы к обработке фраз: биграммный индекс

- Нарезаем тексты на пары слов, строим на парах индекс: *I went, went to, to university, university of, of stanford*.
- Длинные запросы также нарезаем на пары слов: stanford university palo alto=>
- stanford university AND university palo AND palo alto
- !Но могу встретиться документы, которые содержат пары слов, но не содержат фразу
- Проблемы:
  - сверхбольшой индекс
  - проблемы со случайным вхождением слов внутрь биграммы
  - не является стандартным решением

Решение 2. Позиционный индекс

- Храним позиции терма в документе
- <be: 993427;
  - 1 док.: 7, 18, 33, 72, 86, 231…
  - 2 док.: 3, 149…
  - 4 док.: 17, 191, 291, 430, 434
  - 5 док.: 363, 367

- В каком документе может содержаться фраза «to be or not to be»?
- Для сопоставления используется «merging алгоритм», на уровне документа

Обработка фразового запроса

- Извлекаем инвертированный индекс для слов *to, be, or, not*
- ***to****:*
- *2*:1,17,74,222,551; ***4*:8,16,190,429,433**; *7*:13,23,191; **…**
- ***be****:*
- *1*:17,19; ***4*:17,191,291,430,434**; *5*:14,19,101; ...

Обработка запросов с операторами близости

- *Москва /3 университет*
- Только позиционный индекс может использоваться для такой обработки, биграммный не может

- Позиционный индекс сейчас стандартно используется для обработки запросов на близость и фразовых запросов
- Возможно комбинирование биграммных и позиционных индексов

Алгоритм для пересечения близости по словам для двух индексов p1 и p2

```
POSITIONALINTERSECT(p1, p2, k)
 1  answer ← ⟨ ⟩
 2  while p1 ≠ NIL and p2 ≠ NIL
 3  do if docID(p1) = docID(p2)
 4        then l ← ⟨ ⟩
 5             pp1 ← positions(p1)
 6             pp2 ← positions(p2)
 7             while pp1 ≠ NIL
 8             do while pp2 ≠ NIL
 9                do if |pos(pp1) − pos(pp2)| ≤ k
10                      then ADD(l, pos(pp2))
11                      else if pos(pp2) > pos(pp1)
12                              then break
13                   pp2 ← next(pp2)
14                while l ≠ ⟨ ⟩ and |l[0] − pos(pp1)| > k
15                do DELETE(l[0])
16                for each ps ∈ l
17                do ADD(answer, ⟨docID(p1), pos(pp1), ps⟩)
18                pp1 ← next(pp1)
19             p1 ← next(p1)
20             p2 ← next(p2)
21        else if docID(p1) < docID(p2)
22                then p1 ← next(p1)
23                else p2 ← next(p2)
24  return answer
```

Близость слов запроса

- Запросы на естественном языке: набор терм, набиваемых в поисковую строку
- Пользователь предпочитает, чтобы термины запроса встречались недалеко друг от друга
- Пусть *w* – минимальное окно, содержащее все слова запроса, например,
- Для запроса *увольнение директора* минимальное окно в документе *Директора ждало неожиданное увольнение* – 4 слова
- Как учесть в скоринговой функции?

Разборщики запросов (колдунщик)

- Текстовый запрос пользователя может исполниться посредством нескольких запросов к системе, например, запрос: *повышение оплаты труда*
  - Исполняем запрос как фразовый запрос
  - Если <K документов содержат фразу повышение оплаты труда, то исполняются два фразовых запроса *повышение оплаты* и *оплата труда*
  - Если все еще меньше K документов, то запрос на векторное пространство *повышение оплаты труда*
  - Эти операции исполняет разборщик запросов
  - Скоринговая функция должна учитывать разные факторы, включая минимальное окно, в котором содержится запрос

Выводы: основные подходы к обработке фраз

- Фразовый индекс
  - «Запросы в кавычках»
- Позиционный индекс
  - «Запросы в кавычках»
  - Запрос с ограничением близости (proximity)
  - Учет в векторной модели окна расположения

Первые подходы к комбинированию признаков на семинаре РОМИП

Mail.ru на РОМИП-2005

- Комплексная функция релевантности
  - Частота термов
  - Совместная встречаемость термов
  - Зоны документа
  - Неполное вхождение термов запроса в документ
  - Особая обработка длинных документов

Mail.ru на РОМИП-2005

Функция релевантности

$$W = k_f W_f + k_p W_p + k_{ps} W_{ps}$$

W_f Частотность термов запроса по TF*IDF

W_p Встречаемость пар соседних слов запроса

W_{ps} Вес наилучшего пассажа в документе

Пассажи

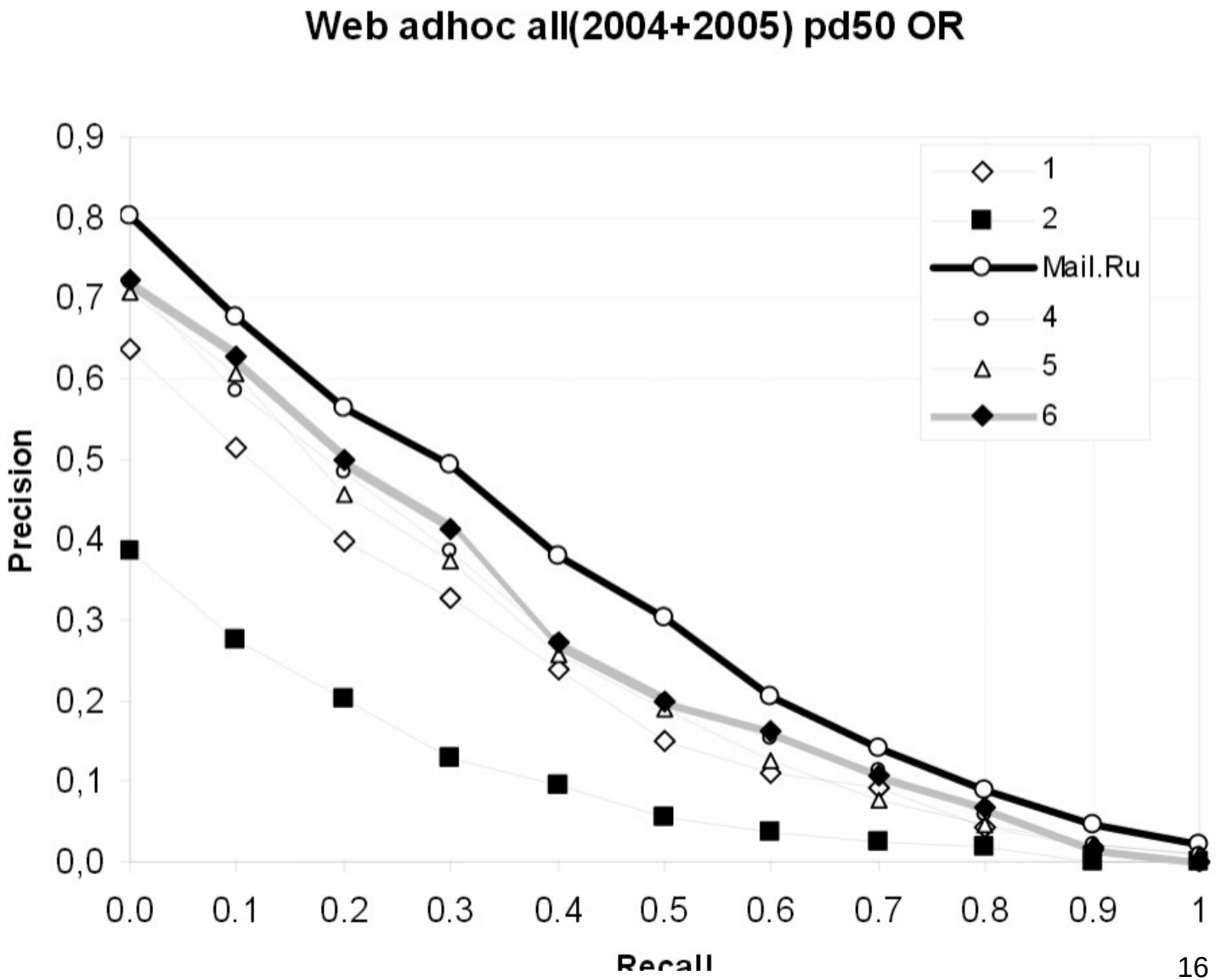
реформа законодательства России

“Введение в политику права” (“Киевские Университетские Известия” за 1896 — 97 гг.). предисловие к ст. “Предстоящая **реформа** акц. **законодательства** в **России**”, (в “Русском экономическом обозрении”, 1896, май — декабрь).

«Картофельные бунты» государственных крестьян в **России**. Отмена в **России** Литовского статута (действовал с 1588). Распространение на западные губернии общероссийского **законодательства**.

Основные характеристики пассажей:

- полнота, длина, порядок слов, зона документа, близость к началу.


Web adhoc all(2004+2005) pd50 OR
Precision
0,9
0,8
0,7
0,6
0,5
0,4
0,3
0,2
0,1
0,0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Recall
1
2
Mail.Ru
4
5
6

РОМИП-2010. Алгоритм А. Сафронова

Простая ранжирующая формула

$$Score(q,d) = \sum_i k_i * F_i(q,d)$$

Я

Факторы

- BM25 для всего документа
- BM25 заголовка
- BM25 начальной части документа
- Вес самой длинной непрерывной цепочки
- Кучность
- YMW

Yandex minimum window

$$YMW(d,q) = \frac{\log(\alpha)}{\log(mw(d,n) - |n| + \alpha)} * \frac{S(n)}{S(q) + \beta * (S(q) - S(n))}$$

$$n = d \cap q$$

$$S(x) = \sum_{t \in x} idf(t)$$

где

d – документ;

q – запрос;

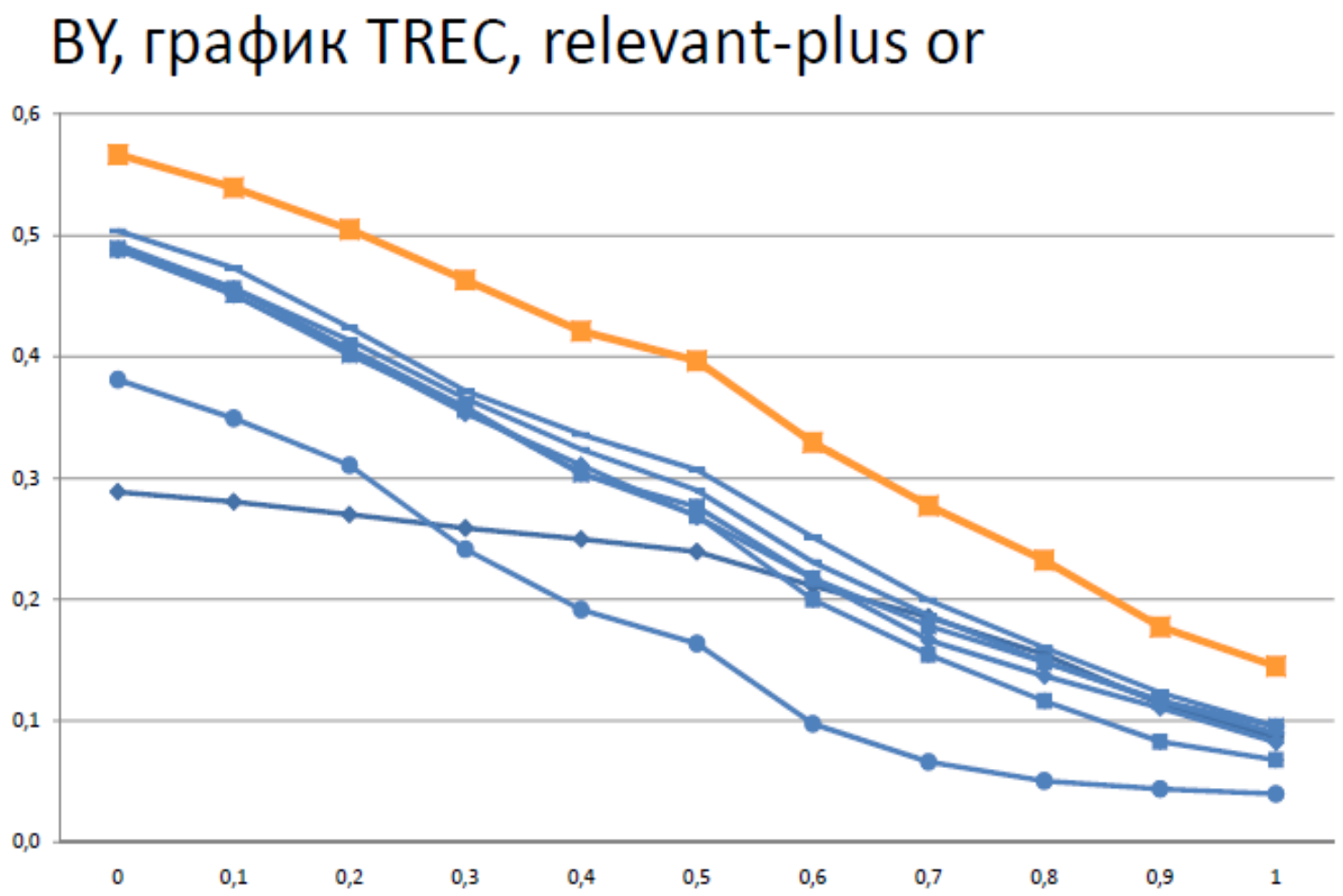
α – константа;

n – множество слов запроса q, встречающихся в документе d;

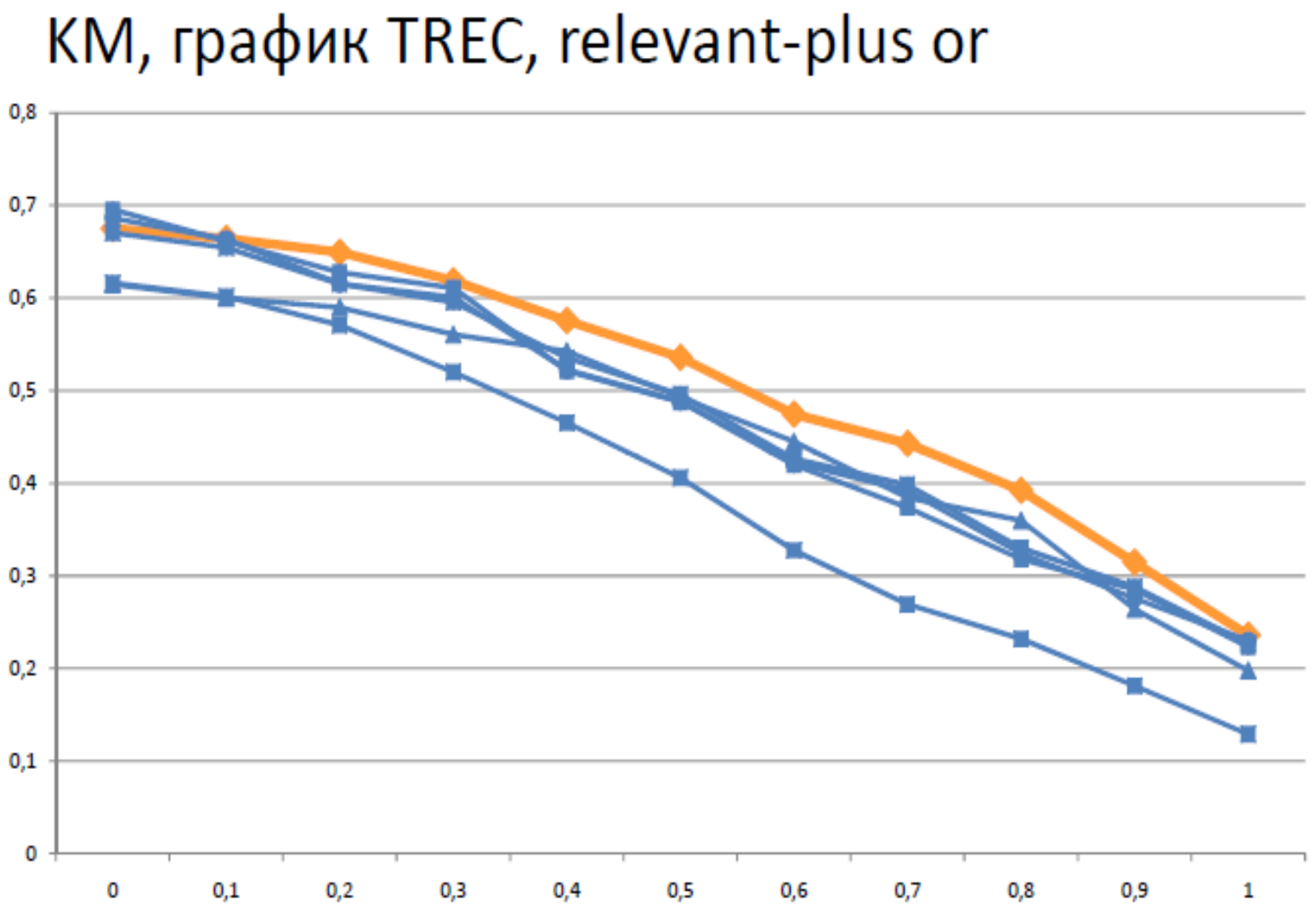
$|n|$ – количество слов запроса q, встречающихся в документе d;

$mw(d,n)$ – размер минимального «куска» текста, в котором встречаются все слова из n;

Результаты: By.Web (РОМИП-2010)



Результаты KM.ru (РОМИП-2010)



Проблема эффективности поиска

- Нужно найти K документов, наиболее похожих на запрос
  - посчитать максимальные косинусы векторов между запросом и документами
- Узкое место: вычисление косинусов
- Считать приблизительно?
- Проблемы
  - Документ, не относящийся к лучшим K документам, может «пробраться» в выдачу

Косинусная мера сама по себе приближение

- Пользователь имеет некоторую задачу и формулирует запрос
- Косинус сравнивает запрос и документ – это приближение потребности пользователя
- Допустимо: неточное вычисление K наиболее похожих документов

Неточная векторная модель: общий подход

- Найти A кандидатов $K < |A| << N$
  - A не обязательно содержит все K лучших документов, но содержит много документов из K
- Выдать K лучших документов из A
- Методы
  - Рассмотреть только слов с высокими idf
    - Ср. Алгоритм Merge в первых лекциях
  - Рассмотреть только документы с большим количеством терминов запроса

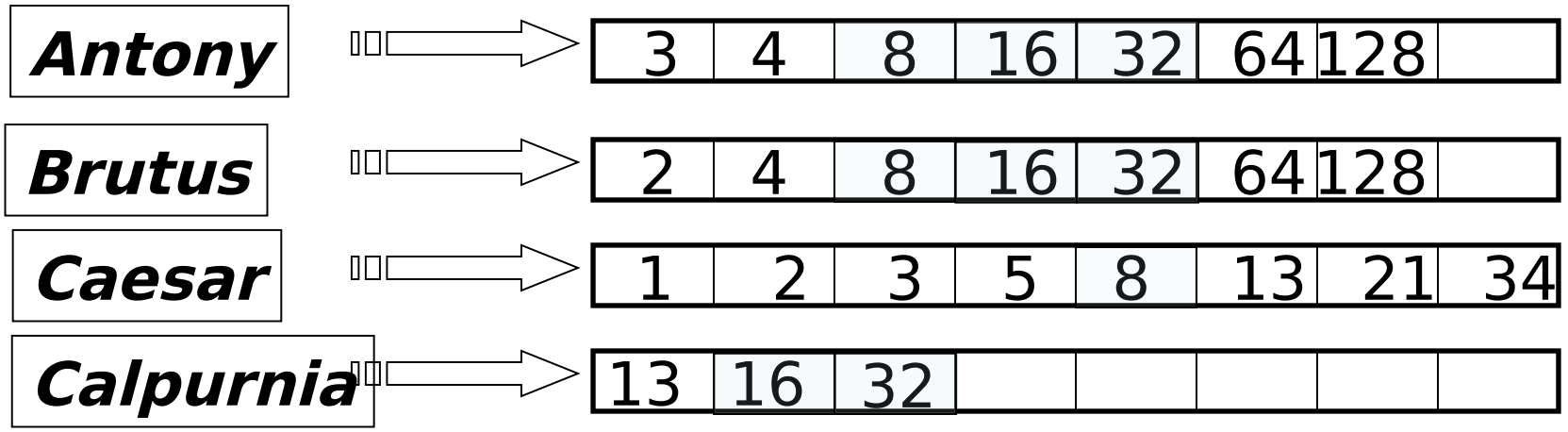
Термы с высоким idf

- Запрос: *Над пропастью во ржи*
- Рассматривать только веса: *пропасть* и *рожь*
- Интуиция : над и во мало вносят в вес документа и не сильно изменят ранжирование
- Преимущества
  - Частотные слова содержатся в большом количестве документов
  - Резкое сокращение просмотра документов

Документы, содержащие много термов запроса

- Любой документ, содержащий хотя бы один терм запроса – кандидат в выдачу
- Для многословных запросов – считать только документы, в которых содержится несколько слова
  - Например, 3 из 4
  - Т.н. мягкая конъюнкция – можно часто встретить в интернет поиске

3 из 4 терминов запроса



Веса считаются только для 8, 16 и 32.

Champion lists

- Заранее для каждого терма t вычисляются r документов с максимальным весом t
  - Топ документов для t
  - r – выбирается во время индексирования
  - r не обязательно то же самое для всех термов
- Во время запроса, считаются только веса документов, входящих хотя бы в один топ-лист одного их термов запроса
  - Выбирается K лучших документов (наиболее похожих на запрос) из документов топ-листов

Учет качества документа

- Авторитетность – качество документа, не зависимое от запроса
- Примеры факторов:
  - Страницы Википедии
  - Статьи из заданного списка СМИ
  - Научные статьи с большим количеством цитирования
  - Сайты с большим количеством входящих ссылок

Моделирование авторитетности

- Присвоим каждому документу не зависимый от запроса вес качества [0,1]
  - Обозначим *g(d)*
  - Например, это количество цитат, нормализованное в шкале [0,1]
- Тогда можно считать net-score:
  - *net-score=α*g(d)+ (1- α)* cosine (q, d)*
- Теперь выбираем лучшие K документов по *net-score*

Лучшие K документов по net-score: идеи

- 1) Упорядочим все документы, соответствующие слову по мере снижения g(d)
  - Топовые документы в первых позициях
- 2) Храним список лучших документов для слова, по net-score
- Ищем лучшие K-результатов только из этого списка

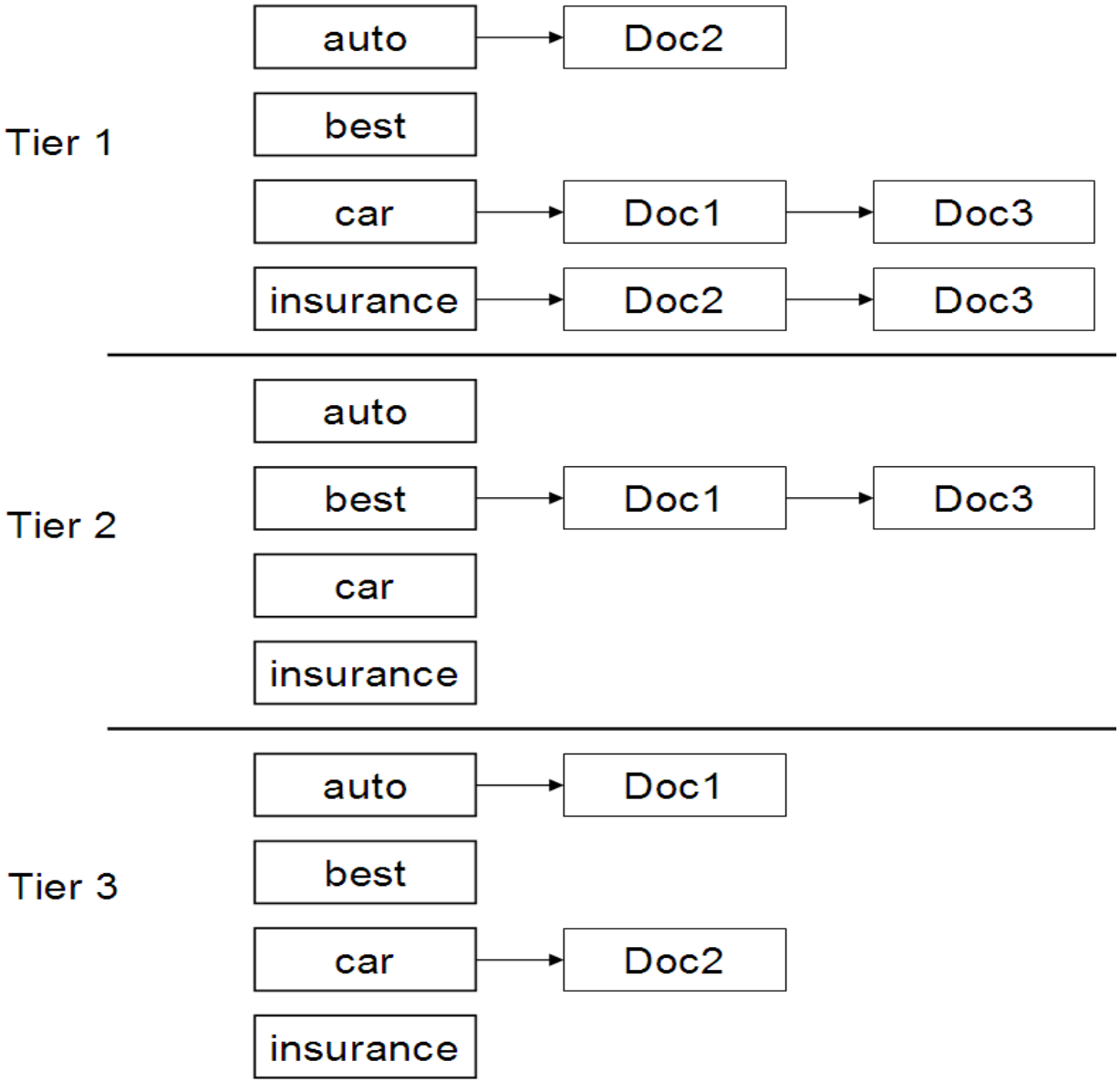
Лучшие K документов по net-score: идеи

- 3) Для каждого терма храним два списка документов high и low
- При проходе по документам проходим сначала по high спискам
  - Если получили K документов, то ОК
  - Иначе начинаем обрабатывать документы из low списков
- При проходе по документам проходим сначала по high спискам
  - Получаются как бы списки разных уровней

Многоуровневые списки

- Разбиение индексов, соответствующих документам по уровням
- Инвертированный список разбивается на уровни снижающейся важности
- При поиске используются индексы большей важности

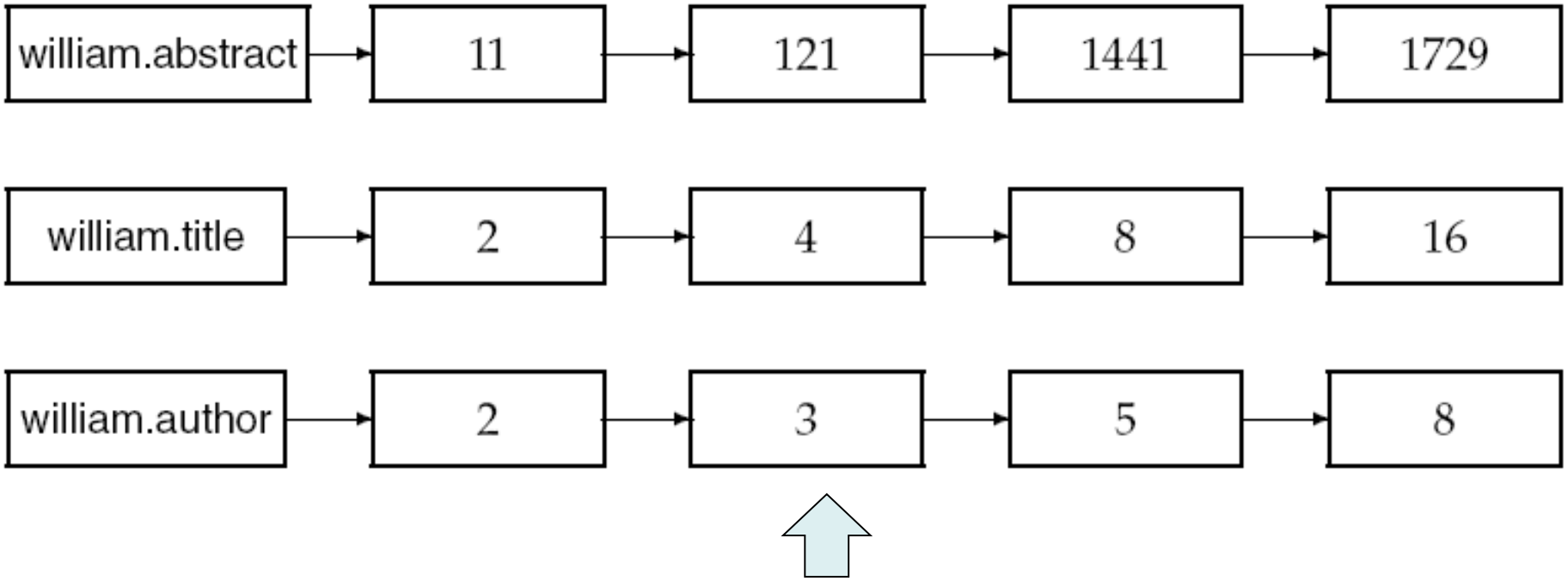
Пример многоуровневого списка



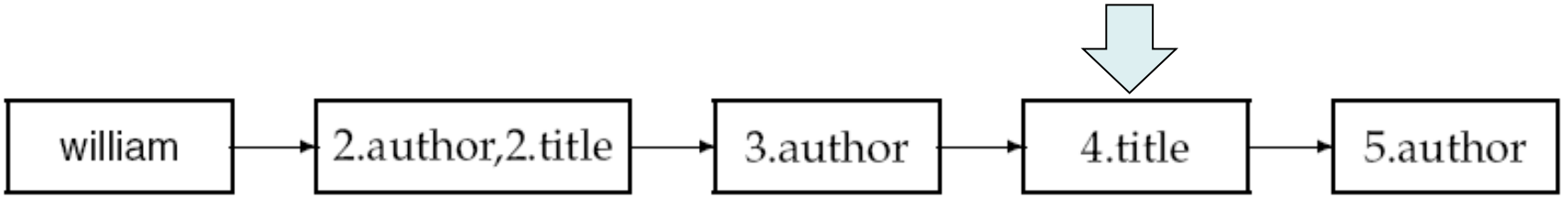
Зоны

- До сих пор: документ – последовательность термов
- Фактически документ имеет много частей, имеющих свой смысл
  - Автор
  - Заголовок
  - Дата публикации
  - Язык
  - Формат
  - и др. метаданные
- Отдельные индексы по зонам метаданных – параметрический индекс

Пример индексов для зон



Учтем зоны в словаре и полях индекса



Заключение

- Количество признаков нарастает
  - Разные модели учета весов слов
  - Различные подходы к расчету весов совместного расположения слов
  - Авторитетность (pagerank и hits)
  - Зоны документа, начало-конец документа
  - Качество сайтов: дубликаты и спам и др.

И это все имеет влияние на релевантность выдаваемых документов

- Нужны комбинированные модели
- Использование машинного обучение - задача learning to rank

Learning to Rank Framework