МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

Урок №1

Вводная лекция

на которой расскажут про курс, что такое система информационного поиска, булевый поиск.

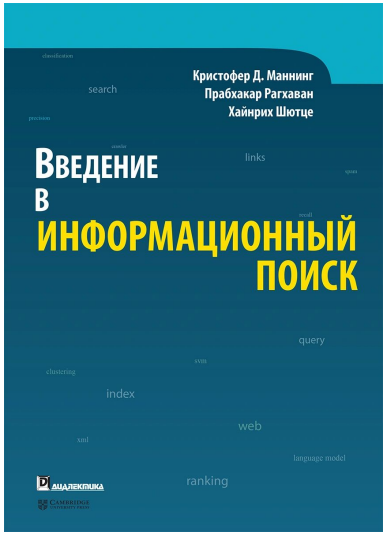
18 февраля 2021 года

Антон Кухтичев

О курсе

Основан на Introduction to Information Retrieval

http://nlp.stanford.edu/IR-book/ (доступен русский перевод)



Много слайдов тоже взяты оттуда, авторы:

Pandu Nayak, Prabhakar Raghavan

Hinrich Schutze, Christina Lioma

До 2021 года курс в МАИ читал Андрей Калинин

Основные тезисы

- Цель курса — разработать поисковую систему;
- Разработка — индивидуальная, без команд;
- ≤ 14 лекций;
- ≤ 14 лабораторных работы;
- 1 курсовая работа;
- 1 экзамен;

Ссылка на презентации/задания: https://github.com/toshunster/MAI-IR

Основные тезисы

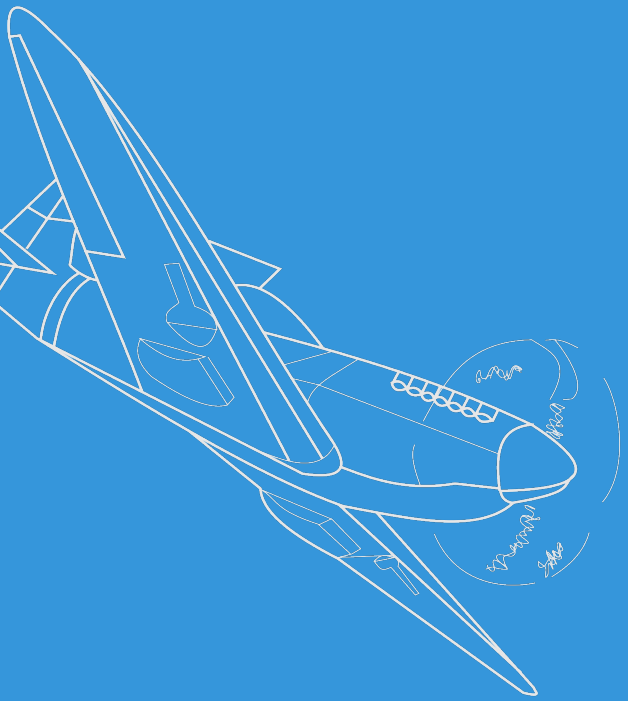
- В качестве языка программирования для всех основных компонент поисковой системы может быть выбран C или C++ без STL;
- Для обвязки, выкачки, может быть выбран любой интерпретируемый язык программирования (Python, Perl, Shell, ...) и дополнительные утилиты (curl, wget, ...)
- Оценка 3 ставится если задача выполнена для корпуса размером в 30-50 тысяч документов
- Оценка 5 ставится при количестве документов больше 1 миллиона (при условии, что они не помещаются все в оперативную память используемого компьютера)
- Кодировка файлов ввода-вывода должна быть единой для всех лабораторных работ, UTF-8.

О лекторе

- Кухтичев Антон Алексеевич,
- Окончил МАИ в 2012 году,
- Работаю в Mail.Ru Group с 2011 года;
- Преподаю:
  - в МАИ с 2013 года принимаю лабораторные работы по курсу «Дискретный анализ»,
  - в 2018-2020 гг. в МФТИ вёл лекции по курсу «Backend разработка» (от Mail.Ru Group),
  - в 2019-2020 гг. в МИФИ вёл лекции по курсу «Backend разработка» и «Углублённый Python» (от Mail.Ru Group),
  - с 2020 года в МГУ веду лекции по курсу «Углублённый C++» (от Mail.Ru Group) и «Углублённый Python»,
  - с 2020 года в МАИ веду лекции по курсу «Программная инженерия».

Содержание занятия

1. Информационный поиск
2. Булевский поиск
3. Домашнее задание;



Информационный поиск

Информационный поиск

Поиск информации (обычно содержащейся в документах) бесструктурной природы (обычно, текстовой), удовлетворяющей информационным нуждам пользователя в больших массивах данных (обычно в компьютерных хранилищах).

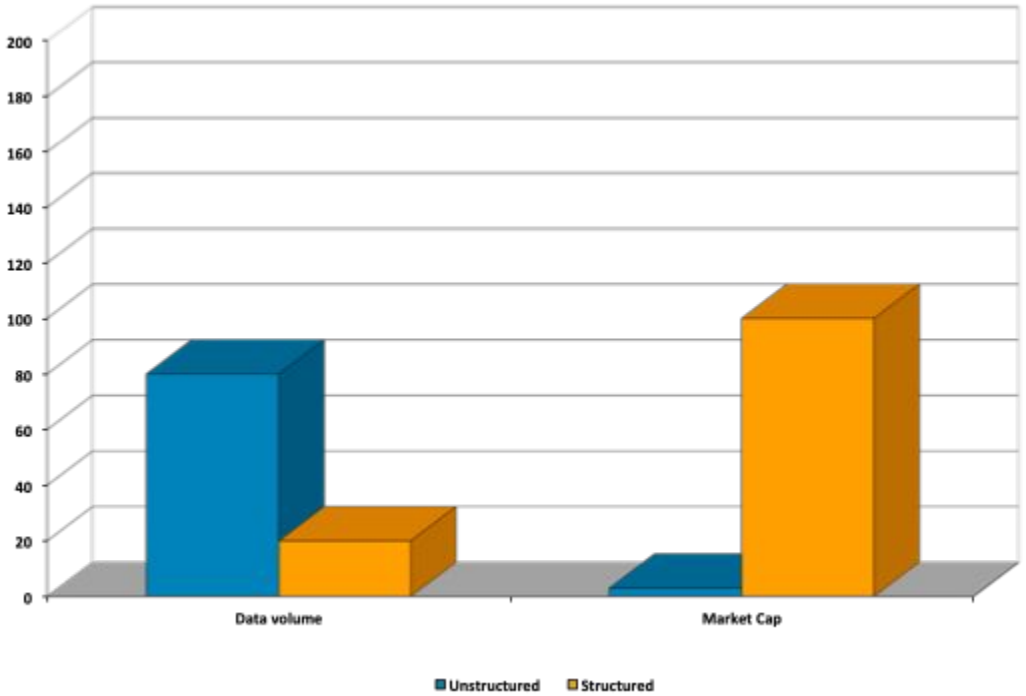
Примеры поисковых систем

- grep,
- find,
- запросы в багтрекере:
  - поиск по патентам
  - поиск по тендерам
- новости и блоги,
- поиск по микроблогам (например, twitter или instagram),
- поиск по изображениям, по видео, по музыке,
- веб-поиск.

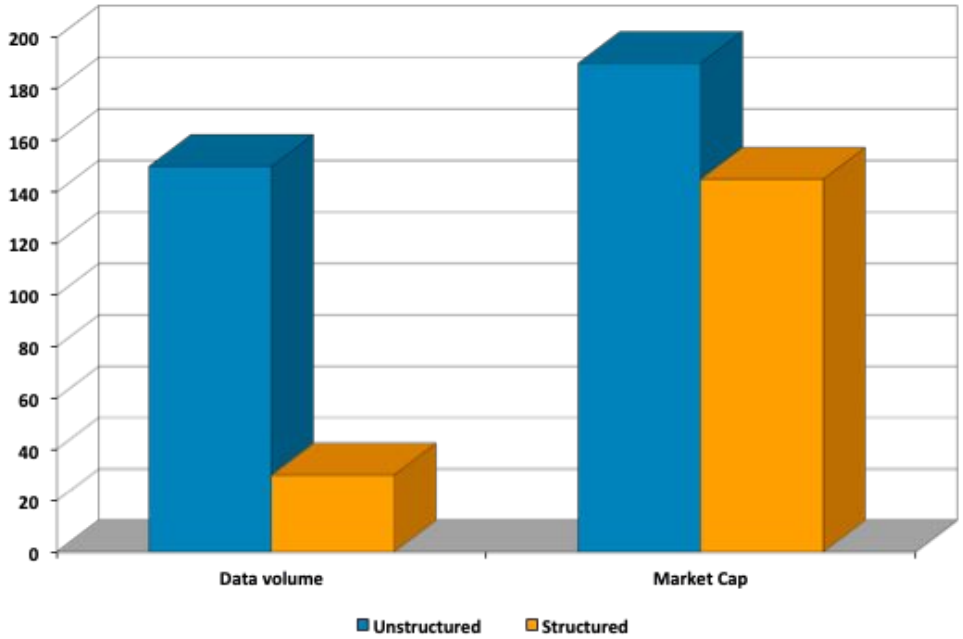
Связанные задачи

- Машинный перевод
- Извлечение мнений
- Распознавание речи
- Синтез речи
- Организация диалога с пользователем

Текстовый поиск против SQL, 1996



Текстовый поиск против SQL, 2006



Информационный поиск как...

- **Наука**
  - Математические модели
  - Статистика, теория вероятностей
  - Лингвистика
- **Практическая задача**
  - Сложность в объёме
  - Принцип KISS
    - Keep It Simple, Stupid
  - Много нюансов и неочевидных, но простых решений

Предположения

Корпус:

- фиксированный набор документов

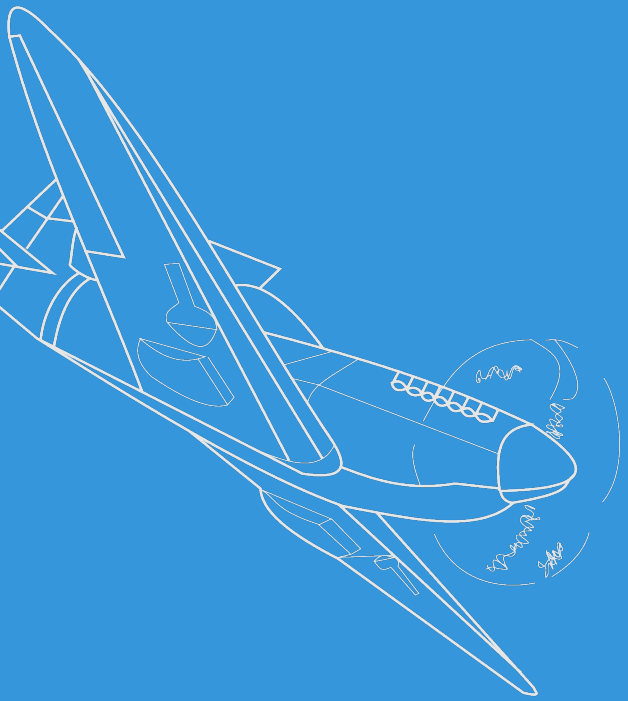
Цель:

- найти документы, релевантные информационным потребностям пользователя
- помогающие ему решить свою задачу.

Модель поиска

С какими сложностями сталкиваемся

- лингвистика: что написано в тексте?
- запросы: что хотел пользователь?
- статистика;
- машинное обучение;
- большие объёмы данных;
- большие нагрузки;
- пользовательский интерфейс;
- форматы данных.



Булевский поиск

Принципы булевского поиска

Запросы – булевские выражения, предикаты

- *Brutus AND Caesar AND NOT Calpurnia*

Возвращаем документы, удовлетворяющие предикату

Результат поиска

Antony and Cleopatra, Act III, Scene ii

Agrippa [Aside to DOMITIUS ENOBARBUS]: Why, Enobarbus,

When Antony found Julius *Caesar* dead,

He cried almost to roaring; and he wept

When at Philippi he found *Brutus* slain.

Hamlet, Act III, Scene ii

Lord Polonius: I did enact Julius *Caesar* I was killed i' the

Capitol; *Brutus* killed me.

Google, Яндекс, Поиск Mail.ru – булевские?

Запрос [w_1 w_2 ... w_n] интерпретируется как

w_1 *AND* w_2 *AND* ... *AND* w_n

Но можно получить документ без w_i:

- Ссылки;
- Вариант w_i (морфология, опечатка, синоним);
- Длинный запрос;
- Булев поиск вернёт мало документов.

Ранжированный поиск

Как реализовать булев поиск?

grep

- найти строчки с ***Brutus AND Caesar***;
- удалить ***Calpurnia***.

Почему плохо?

- медленно;
- построчно, нужно подокументно;
- сложно для NOT Calpurnia;
- плохо расширяется;
- не отменяет ранжирование.

Матрица «термин-документ»

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	1	1	0	0	0	1
Brutus	1	1	0	1	0	0
Caesar	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0

Brutus AND Caesar BUT NOT Calpurnia

В пьесе Julius Caesar встречается слово Calpurnia

Битовые операции

- Для каждого термина – вектор из 0 и 1
- Извлекаем столбцы для *Brutus*, *Caesar* и инвертируем столбец для *Calpurnia*
- 110100 *AND* 110111 *AND* 101111 = 100100.
- А какие есть проблемы с таким подходом к поиску?

Построить матрицу нельзя

- N=10^6 документов по 1000 токенов
- 6 байтов на токен (8-9 для русского языка), включая пробелы и пунктуацию
- Размер корпуса 6*109 = 6 ГБ
- M=500 000 разных терминов
- 500 000 * 10^6 = полтриллиона нулей и единиц
- Но единиц – не более миллиарда

Кстати, о количестве терминов

- Телефонные справочники
- md5-хеши файлов
- Udaff.com
- Сайт с диссертациями

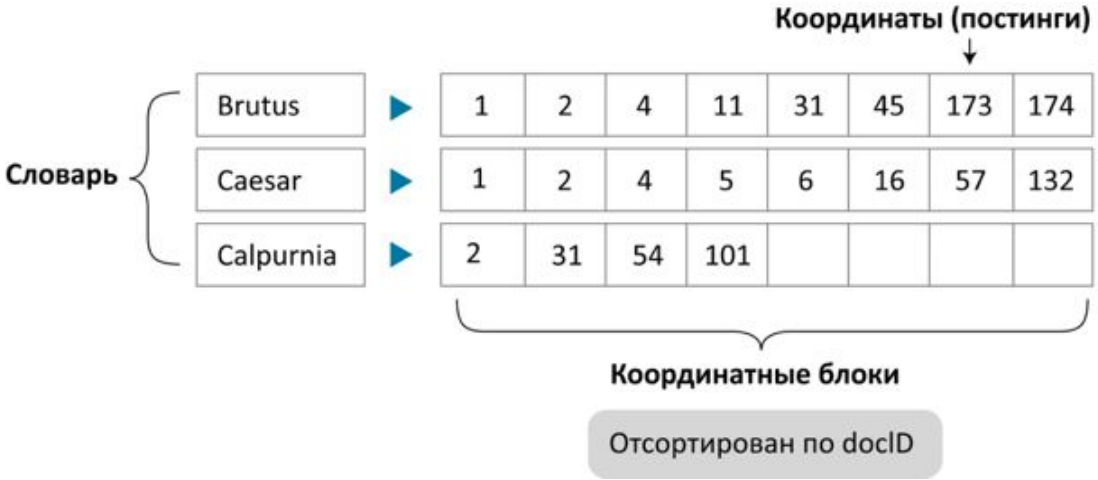
Обратный индекс

- Для каждого термина t будем хранить список документов, где он встречается
- Каждый документ представлен docID
- Как хранить? Как изменять?

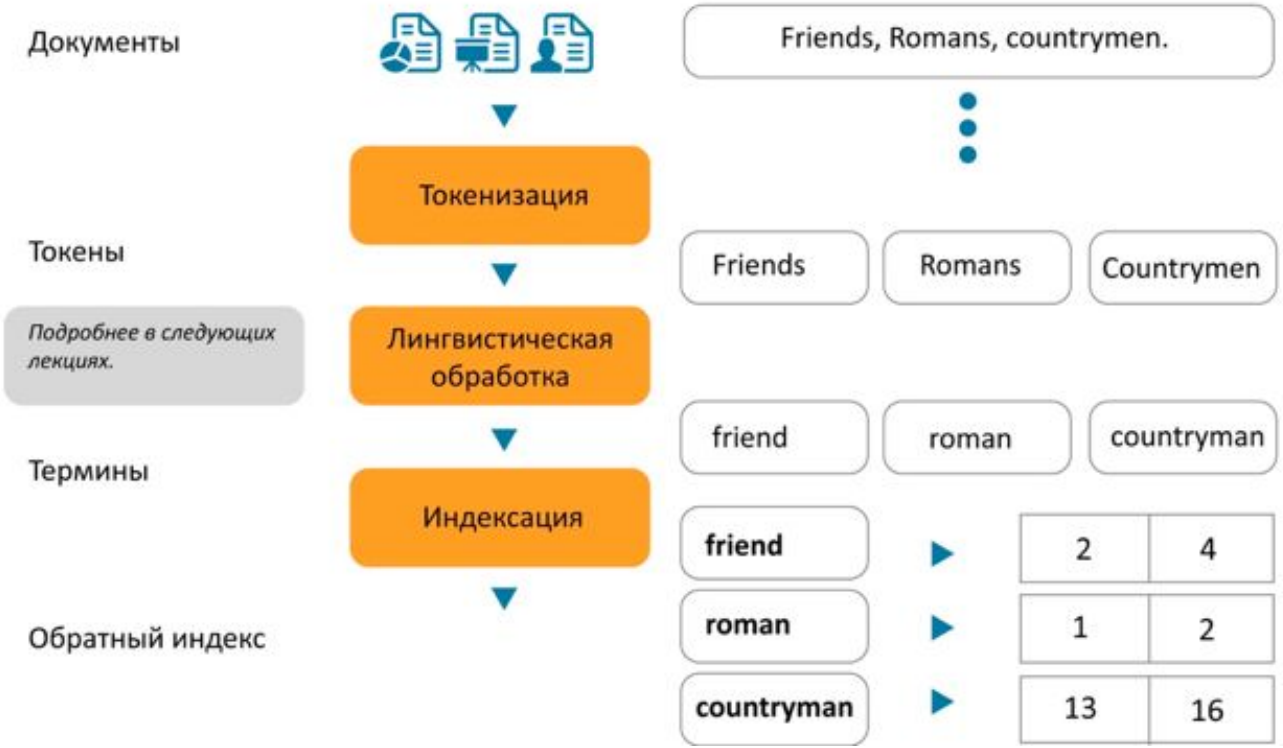
Brutus	▶	1	1	4	11	31	45	173	174
Caesar	▶	1	2	4	5	6	16	57	132
Calpurnia	▶	31	54	101					

Словарь и координатные блоки

- Массивы переменной длины
- Списки (или списки массивов)
- Всегда компромисс



Создание обратного индекса



Разбиение на токены

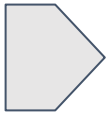
Последовательность пар (термин, DocID)

Doc 1

I did enact Julius Caesar I was killed i' the Capitol; Brutus killed me.

Doc 2

So let it be with Caesar. The noble Brutus hath told you Caesar was ambitious



Term	docID
I	1
did	1
enact	1
julius	1
caesar	1
I	1
was	1
killed	1
i'	1
the	1
capitol	1
brutus	1
killed	1
me	1
so	2
let	2
it	2
be	2
with	2
caesar	2
the	2
noble	2
brutus	2
hath	2
told	2
you	2
caesar	2
was	2
ambitious	2

Сортировка

Сортируем по терминам

Потом по docID

Основа индескации!

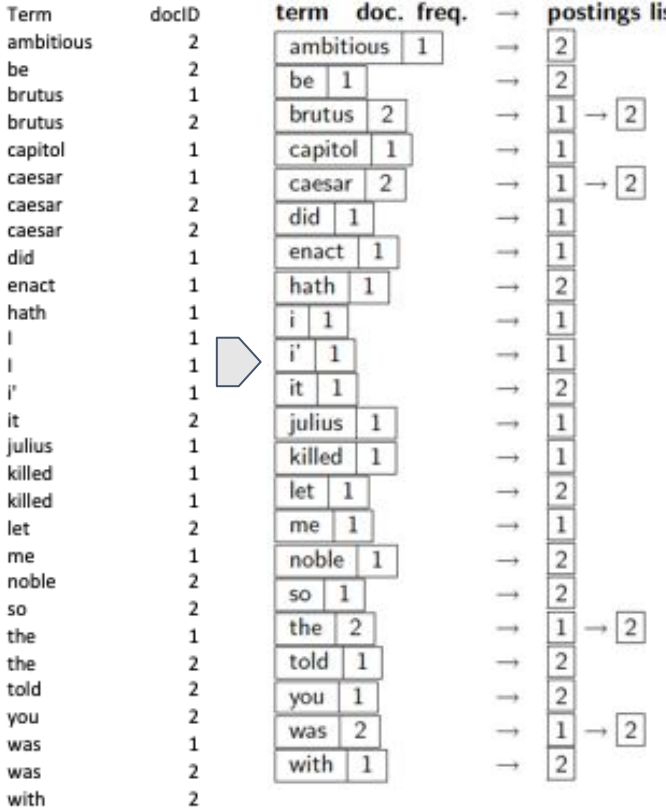
Term	docID
I	1
did	1
enact	1
julius	1
caesar	1
I	1
was	1
killed	1
i'	1
the	1
capitol	1
brutus	1
killed	1
me	1
so	2
let	2
it	2
be	2
with	2
caesar	2
the	2
noble	2
brutus	2
hath	2
told	2
you	2
caesar	2
was	2
ambitious	2

Term	docID
ambitious	2
be	2
brutus	1
brutus	2
capitol	1
caesar	1
caesar	2
caesar	2
did	1
enact	1
hath	1
I	1
I	1
i'	1
it	2
julius	1
killed	1
killed	1
let	2
me	1
noble	2
so	2
the	1
the	2
told	2
you	2
was	1
was	2
with	2

Словари и координатные блоки

- Вхождения термина в один документ сливаются
- Делим на словарь и координатные блоки
- Запоминаем частотную информацию

Про необходимость частот позже



Затраты на хранение

Термины
и счётчики

Списки docID

Указатели

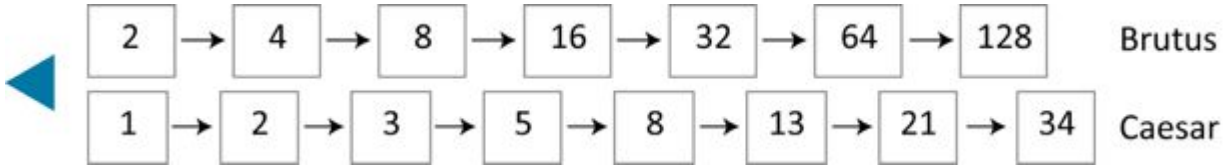
Позже:

- Эффективная индексация
- Сколько нужно места

Выполнение запросов: AND

Запрос: *Brutus AND Caesar*

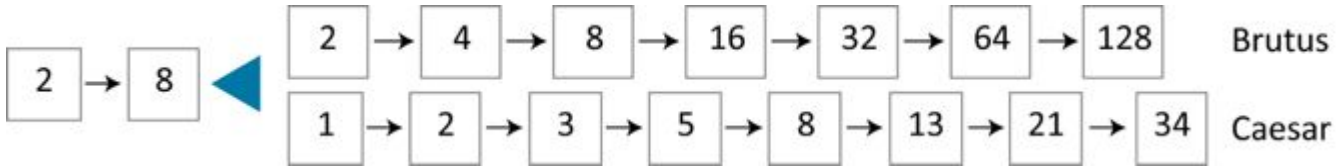
- Найдём ***Brutus*** в словаре;
  - Получим координатные блоки.
- Найдём ***Caesar*** в словаре;
  - Получим координатные блоки.
- Пересечём координатные блоки:



Пересечение

Будем идти по спискам параллельно друг другу

- Если размеры списков m and n, потребуется O(m+n) действий
- **Важно**: сортировка по docID.



Пересечение координатных блоков

```
INTERSECT(p1, p2)
 1  answer ← ⟨ ⟩
 2  while p1 ≠ NIL and p2 ≠ NIL
 3  do if docID(p1) = docID(p2)
 4        then ADD(answer, docID(p1))
 5             p1 ← next(p1)
 6             p2 ← next(p2)
 7        else if docID(p1) < docID(p2)
 8                then p1 ← next(p1)
 9                else p2 ← next(p2)
10  return answer
```

Точное совпадение

Булев поиск прост для реализации и понимания пользователем

- Документ – множество слов
- Условие либо выполняется, либо нет

Основная модель в течение 3 десятилетий

Множество поисковых систем до сих пор булевские

Ещё запросы – как их выполнять?

- *Brutus AND NOT Caesar*
- *Brutus OR NOT Caesar*
- *(Brutus OR Caesar) AND*

  NOT (Antony OR Cleopatra)

- Будут ли эти запросы выполняться линейно?
  - Вообще – линейно относительно чего?
- Можно представить в виде дерева!

Оптимизация выполнения запросов

- Запрос из N операторов AND
- Brutus AND Calpurnia AND Caesar
- В каком порядке выполнять запрос?

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

Пример оптимизации запроса

Выполняем в порядке возрастания частот:

- От самой маленькой к большим
- Запрос выполняется как ***(Calpurnia AND Brutus) AND Caesar.***

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

Более общая оптимизация

Запрос: (madding OR crowd) AND (ignoble OR strife)

- Получаем частоты для всех терминов
- Оцениваем «частоты» каждого OR суммой терминов
- Обрабатываем в порядке возрастания частот.

Домашнее задание №1

1. Необходимо подготовить корпус документов, который будет использован при выполнении остальных лабораторных работ;
   a. Ознакомиться с ним, изучить его характеристики. Из чего состоит текст? Есть ли дополнительная мета-информация? Если разметка текста, какая она?
   b. Разбить на документы.
   c. Выделить текст.
2. Подготовить отчёт

Рекомендуемая литература

[Введение в информационный поиск | Маннинг Кристофер Д., Шютце Хайнрих]

Для саморазвития (опционально)
[Чтобы не набирать двумя пальчиками]

Спасибо за внимание!

Антон Кухтичев

a.kukhtichev@mail.ru

@toshunster