Data structures in Information Retrieval

Max Gubin
mail@maxgubin.com

RuSSIR
Russian Summer School in Information Retrieval
2008

Information Retrieval History

4000 BC

1950

2000

Information Retrieval Tasks

Types of information:

- Text
- Sound
- Image

Mixes…

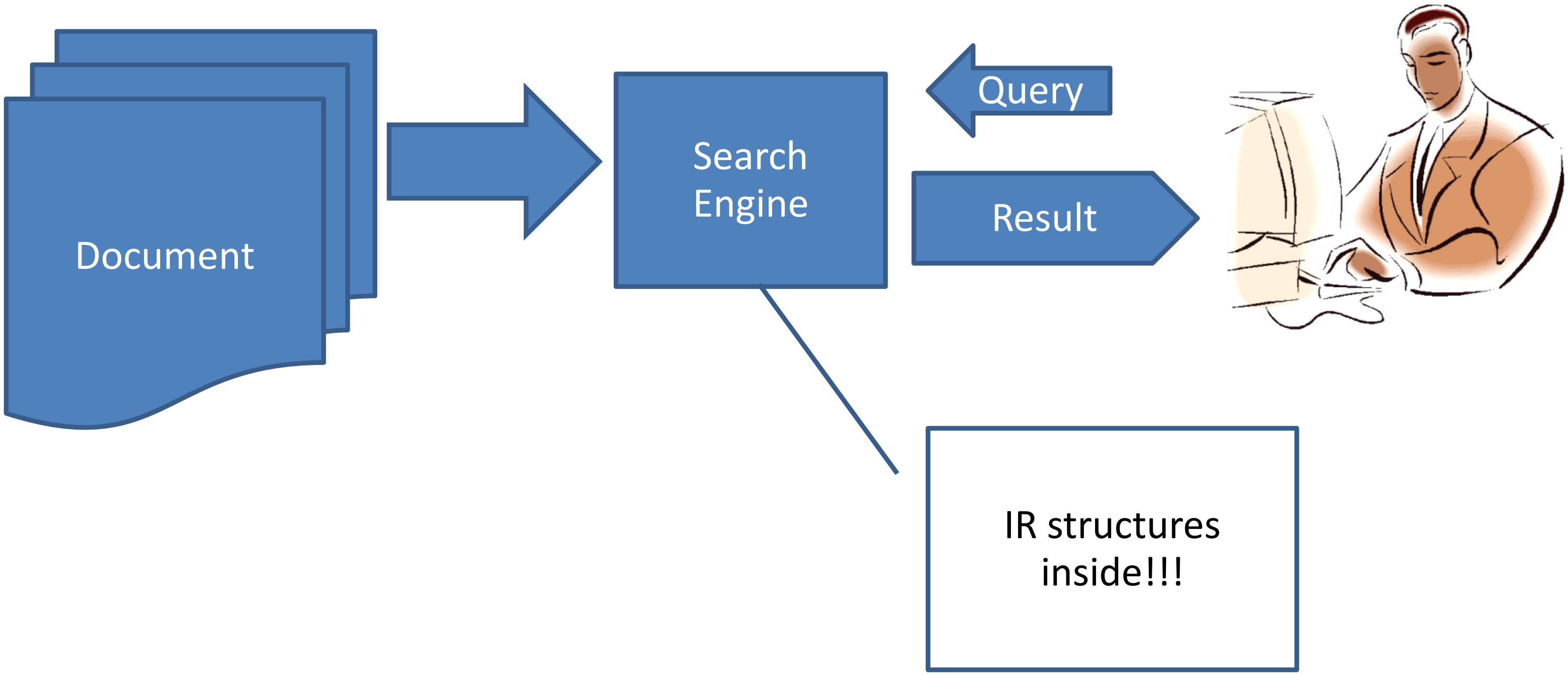
Types of tasks:

- Search
- Classification/clustering
- Extraction/Summarization

Mixes…

Toy project

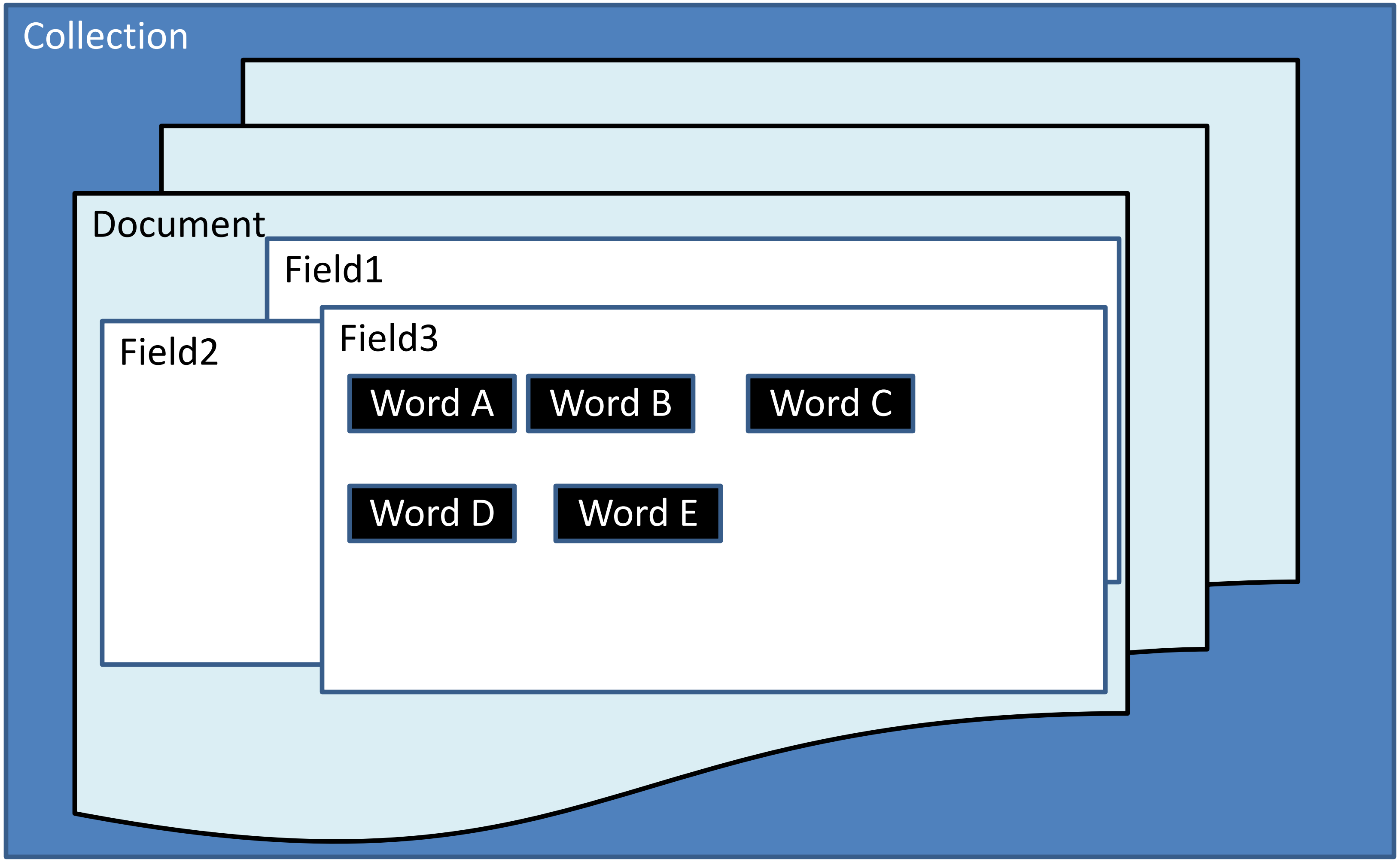
Let’s create a toy search engine:



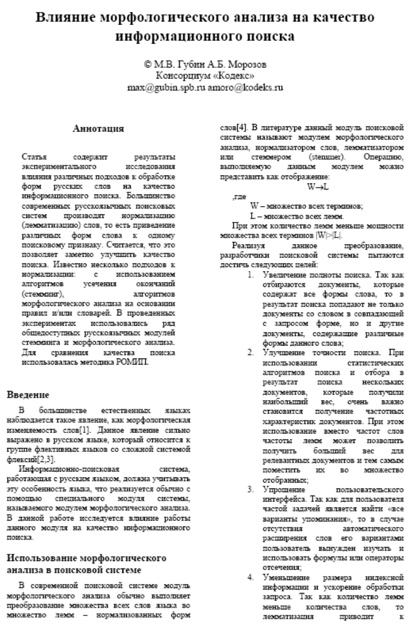
Course Outline

- Introduction (the problem definition)
- Basics (structures and environments)
- Building index
- Search!
- Other data: Language Models and Link Graphs

Hierarchy of data in text IR



Linearization (word extraction)



Влияние морфологического анализа на качество информационного поиска

© М.В. Губин А.Б. Морозов
Консорциум «Кодекс»
max@gubin.spb.ru amoro@kodeks.ru

Аннотация

Статья содержит результаты экспериментального исследования влияния различных подходов к обработке форм русских слов на качество информационного поиска. Большинство современных русскоязычных поисковых систем производят нормализацию (лемматизацию) слов, то есть приведение различных форм слова к одному поисковому признаку. Считается, что это позволяет заметно улучшить качество поиска. Известно несколько подходов к нормализации: с использованием алгоритмов усечения окончаний (стемминг), алгоритмов морфологического анализа на основании правил и/или словарей. В проведенных экспериментах использовались ряд общедоступных русскоязычных модулей стемминга и морфологического анализа. Для сравнения качества поиска использовалась методика РОМИП.

Введение

В большинстве естественных языках наблюдается такое явление, как морфологическая изменяемость слов[1]. Данное явление сильно выражено в русском языке, который относится к группе флективных языков со сложной системой флексий[2,3].

Информационно-поисковая система, работающая с русским языком, должна учитывать эту особенность языка, что реализуется обычно с помощью специального модуля системы, называемого модулем морфологического анализа. В данной работе исследуется влияние работы данного модуля на качество информационного поиска.

Использование морфологического анализа в поисковой системе

В современной поисковой системе модуль морфологического анализа обычно выполняет преобразование множества всех слов языка во множество лемм – нормализованных форм слов[4]. В литературе данный модуль поисковой системы называют модулем морфологического анализа, нормализатором слов, лемматизатором или стеммером (stemmer). Операцию, выполняемую данным модулем можно представить как отображение:

$$W \rightarrow L$$

,где

W – множество всех терминов;

L – множество всех лемм.

При этом количество лемм меньше мощности множества всех терминов |W|>|L|.

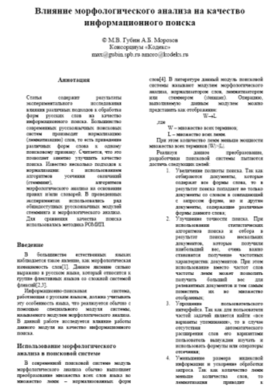
Реализуя данное преобразование, разработчики поисковой системы пытаются достичь следующих целей:

1. Увеличение полноты поиска. Так как отбираются документы, которые содержат все формы слова, то в результат поиска попадают не только документы со словом в совпадающей с запросом форме, но и другие документы, содержащие различные формы данного слова;
2. Улучшение точности поиска. При использовании статистических алгоритмов поиска и отбора в результат поиска нескольких документов, которые получили наибольший вес, очень важно становится получение частотных характеристик документов. При этом использование вместо частот слов частоты лемм может позволить получить больший вес для релевантных документов и тем самым поместить их во множество отобранных;
3. Упрощение пользовательского интерфейса. Так как для пользователя частой задачей является найти «все варианты упоминания», то в случае отсутствия автоматического расширения слов его вариантами пользователь вынужден изучать и использовать формулы или операторы отсечения;
4. Уменьшение размера индексной информации и ускорение обработки запроса. Так как количество лемм меньше количества слов, то лемматизация приводит к

(“To”, 1, Body, Document1)
(“BE”,2, Body,Document1)
(“or”,3,Body,Document1)
(“not”,4, Body,Document1)
(“to”, 5, Body, Document1)
(“be”,6, Body,Document1)

Document formats

- Presentation oriented (PDF, RTF)

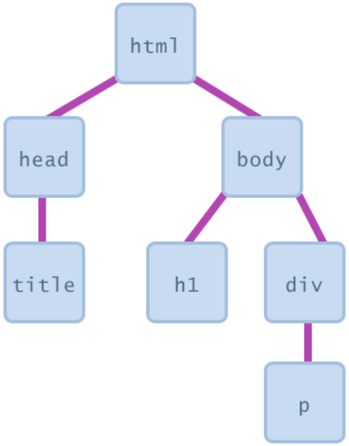


Влияние морфологического анализа на качество информационного поиска

© М.В. Губин А.Б. Морозов
Консорциум «Кодекс»

Аннотация

- Structure Oriented (SGML, HTML, XML)



Encodings

- **Present all letters** of the alphabet
- **Collation (case)** – can be complex in some languages: a A ä Ä**;**
  ئۈ ئۈ ئۆ ئۇ ئو ئـ ئـ ئھ ئھ ئن ئم ئم ئز ئر ئخ ئح ئحـ ئج ئجـ ئا ئا ئى ئى ئـ ئـ

Official standard Unicode

Latest version 5.10 about 100000 characters:

Character codes (codepoints 0 10FFFF)

Encoding rules (utf-8, utf-16, utf-32)

Algorithms

RuSSIR
Russian Summer School in Information Retrieval
2008

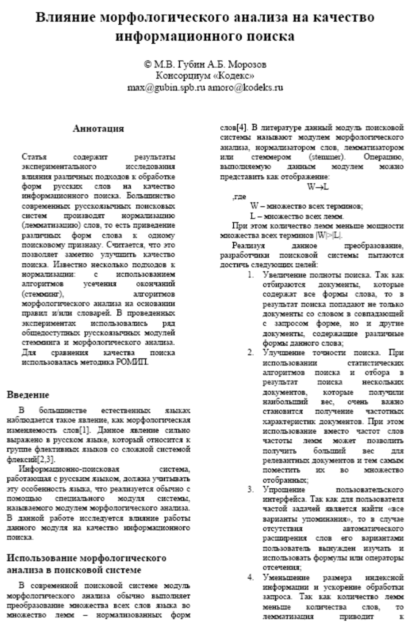
Words

- Morphology agglunative, multiroot,
- Abbreviations
- Spelling variants
- Stop-words

How to handle:

1. During document analysis
2. During search

Linearization (complex)



Влияние морфологического анализа на качество информационного поиска

© М.В. Губин А.Б. Морозов
Консорциум «Кодекс»
max@gubin.spb.ru amoro@kodeks.ru

Аннотация

Статья содержит результаты экспериментального исследования влияния различных подходов к обработке форм русских слов на качество информационного поиска. Большинство современных русскоязычных поисковых систем производят нормализацию (лемматизацию) слов, то есть приведение различных форм слова к одному поисковому признаку. Считается, что это позволяет заметно улучшить качество поиска. Известно несколько подходов к нормализации: с использованием алгоритмов усечения окончаний (стемминг), алгоритмов морфологического анализа на основании правил и/или словарей. В проведенных экспериментах использовались ряд общедоступных русскоязычных модулей стемминга и морфологического анализа. Для сравнения качества поиска использовалась методика РОМИП.

Введение

В большинстве естественных языках наблюдается такое явление, как морфологическая изменяемость слов[1]. Данное явление сильно выражено в русском языке, который относится к группе флективных языков со сложной системой флексий[2,3].

Информационно-поисковая система, работающая с русским языком, должна учитывать эту особенность языка, что реализуется обычно с помощью специального модуля системы, называемого модулем морфологического анализа. В данной работе исследуется влияние работы данного модуля на качество информационного поиска.

Использование морфологического анализа в поисковой системе

В современной поисковой системе модуль морфологического анализа обычно выполняет преобразование множества всех слов языка во множество лемм – нормализованных форм слов[4]. В литературе данный модуль поисковой системы называют модулем морфологического анализа, нормализатором слов, лемматизатором или стеммером (stemmer). Операцию, выполняемую данным модулем можно представить как отображение:

W→L

,где

W – множество всех терминов;

L – множество всех лемм.

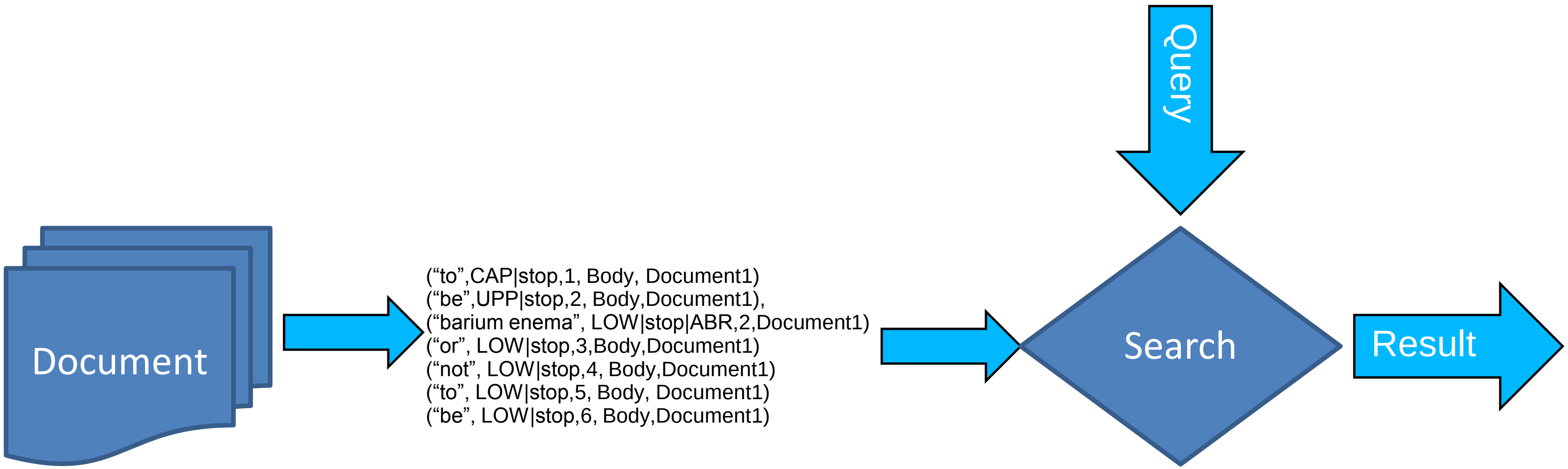
При этом количество лемм меньше мощности множества всех терминов |W|>|L|.

Реализуя данное преобразование, разработчики поисковой системы пытаются достичь следующих целей:

1. Увеличение полноты поиска. Так как отбираются документы, которые содержат все формы слова, то в результат поиска попадают не только документы со словом в совпадающей с запросом форме, но и другие документы, содержащие различные формы данного слова;
2. Улучшение точности поиска. При использовании статистических алгоритмов поиска и отбора в результат поиска нескольких документов, которые получили наибольший вес, очень важно становится получение частотных характеристик документов. При этом использование вместо частот слов частоты лемм может позволить получить больший вес для релевантных документов и тем самым поместить их во множество отобранных;
3. Упрощение пользовательского интерфейса. Так как для пользователя частой задачей является найти «все варианты упоминания», то в случае отсутствия автоматического расширения слов его вариантами пользователь вынужден изучать и использовать формулы или операторы отсечения;
4. Уменьшение размера индексной информации и ускорение обработки запроса. Так как количество лемм меньше количества слов, то лемматизация приводит к

(“to”,CAP|stop,1, Body, Document1)
(“be”,UPP|stop,2, Body,Document1),
(“barium enema”, LOW|stop|ABR,2,Document1)
(“or”, LOW|stop,3,Body,Document1)
(“not”, LOW|stop,4, Body,Document1)
(“to”, LOW|stop,5, Body, Document1)
(“be”, LOW|stop,6, Body,Document1)

Naïve Scan (grep approach)



- Have the whole context for analysis
- Match current hardware architecture
- Usually can be easily parallelized

Adding index

Two meanings of index:

- Taxonomy that accelerates human search
- **Special data structure** that accelerate data access

Using Standard Database

Dictionary

Word	ID
to	1
be	2
not	4
or	3

Doctable

Document	ID
Hamlet	1
Introduction to…	2
Dive into Python	3

Positions

WordID	DocID	Flags	Fields	Pos
1	1	CAP	BODY	1
2	1	CAP	BODY	2
3	1		BODY	3
4	1		BODY	4
1	1		BODY	5
2	1		BODY	6

```
SELECT DocTable.Document FROM Dictionary,Doctable,Positions
WHERE Dictionary.word=? AND Dictionary.ID=Positions.WordID AND
Doctable.ID=Positions.DocID
```

Bag of words

Dictionary

Word	ID
to	1
be	2
not	4
or	3

Positions

WordID	DocID	Flags	Fields	Count
1	1	CAP	BODY	2
2	1	CAP	BODY	2
3	1		BODY	1
4	1		BODY	1

Doctable

Document	ID
Hamlet	1
Introduction to…	2
Dive into Python	3

Problems with General Purpose Databases

1. Size
2. Speed build
3. Speed search

This is a tool for another task

Matrix representation

Simple example

1. Dad is reading a book
2. Mom is watching TV
3. Dad and Mom are at home

	1	2	3
a	1	0	0
and	0	0	1
are	0	0	1
at	0	0	1
book	1	0	0
Dad	1	0	1
Mom	0	1	1
is	1	1	0
reading	1	0	0
home	0	0	1
TV	0	1	0

Main IR structure

A sparse n-dimensional matrix in different presentations is
"**THE MAIN IR STRUCTURE**"

Search – inverted index

Language models – table of probabilities

Link analysis – Adjacency matrix

Sparseness of the matrix

Example:

N - 1 mln documents

Ds - 1000 words/document

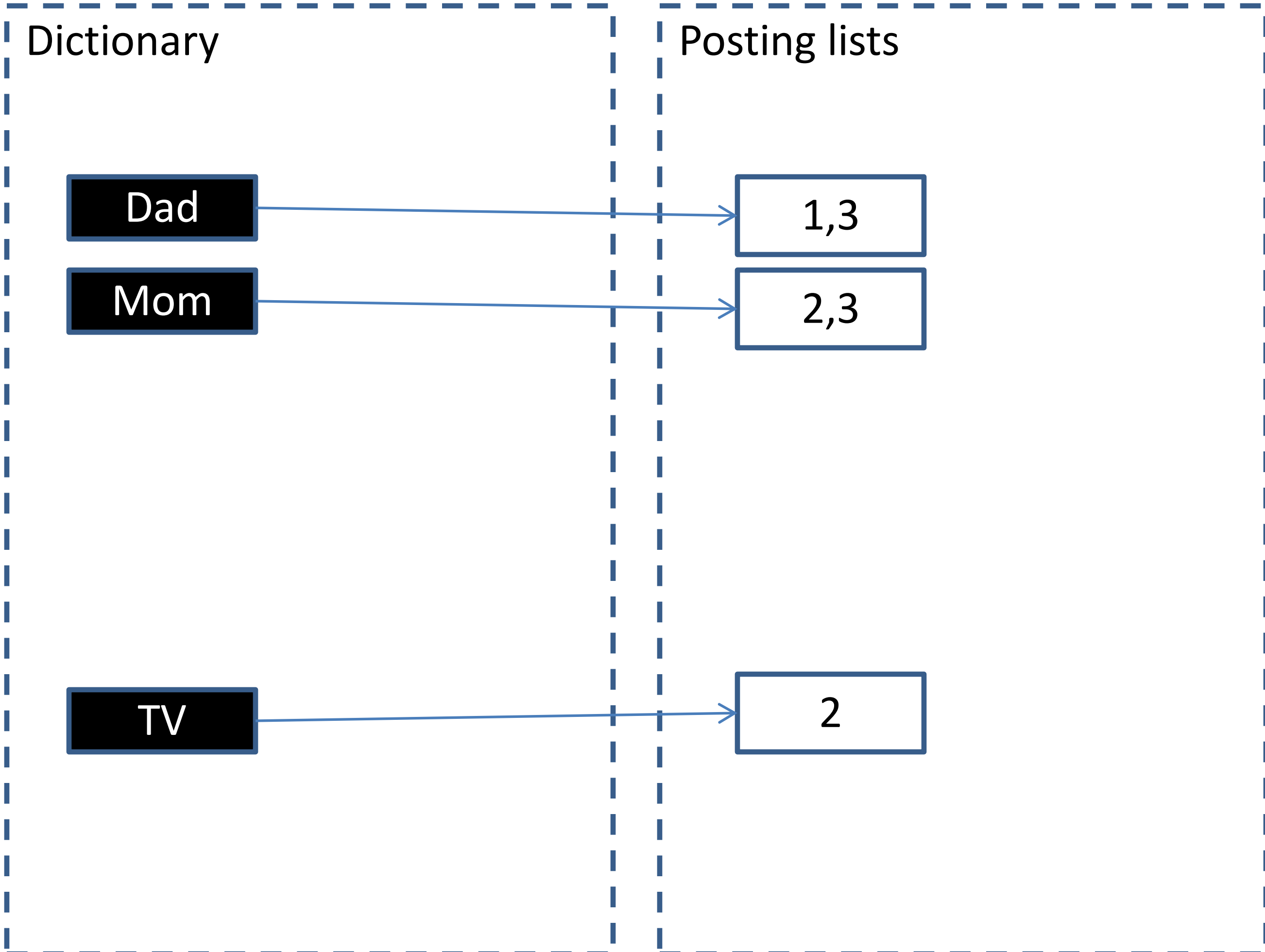
D – 500 000 words in dictionary

|Word/Document matrix| = D*N = 500 bln

Words in collection = 1 mln * 1000 = 1 bln

<u>Only 0.2% elements in the matrix are not 0</u>

Inverted file



Signature file

Signatures for words (function)

Dad → 00000001

Mom → 00001000

TV → 10000000

watching → 00001000

football → 00001000

Doc Signature = OR words

1	**00110001**
2	01011000
3	10001001

Signature file (Search)

Query = “Mom Dad”
q_s = 00001001

1	**00110001**
2	01011000
3	10001001

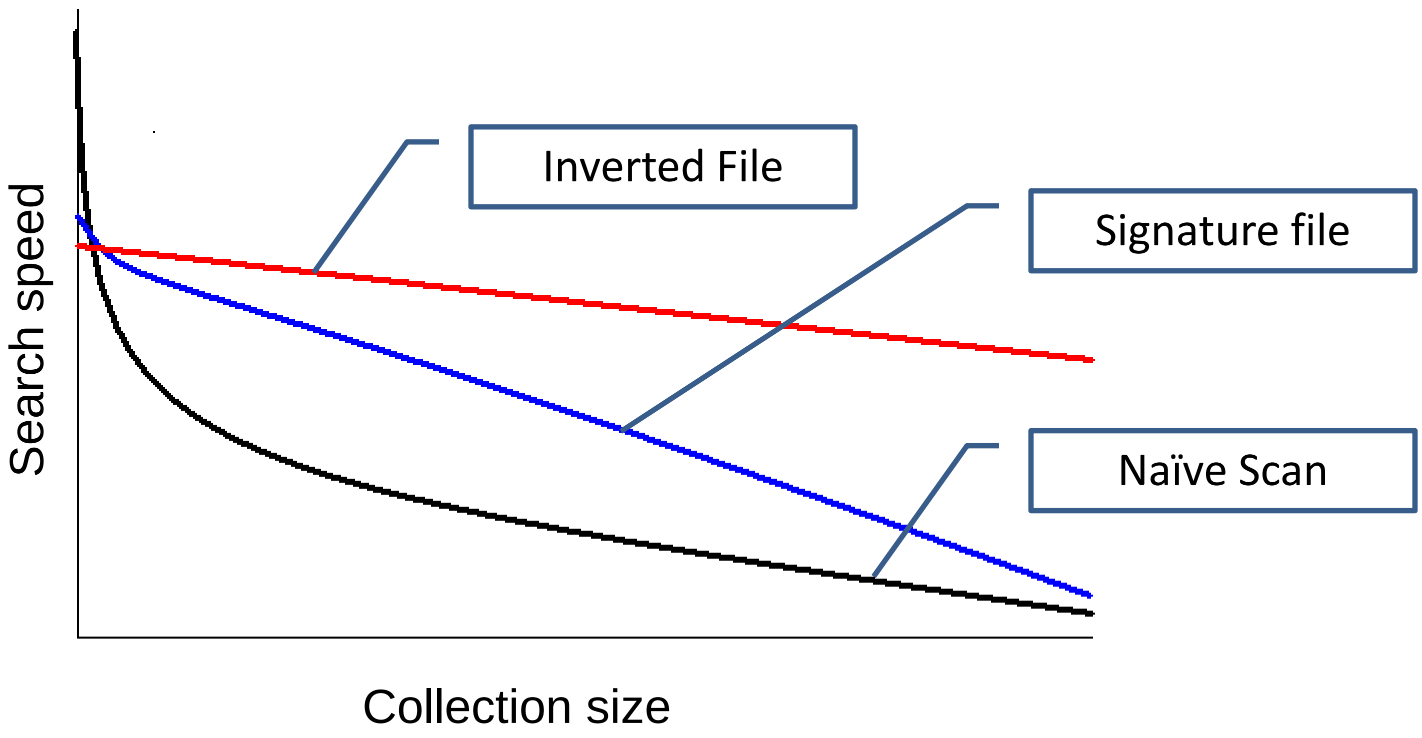
for doc **in** Document_Signatures:
    if doc.signature & q_s = q_s:
        ScanDocument(doc.id)

An old structure = hash + bloom filter + scan

IR Packages

- Lucene (http://lucene.apache.org/)
- Terrier (http://ir.dcs.gla.ac.uk/terrier/)
- Lemur & Indri (http://www.lemurproject.org/)
- Zettair (http://www.seg.rmit.edu.au/zettair/)
- Zebra (http://www.indexdata.dk/zebra/)

Search speed



RuSSIR
Russian Summer School
in Information Retrieval
2008

Speed (Size) depends on

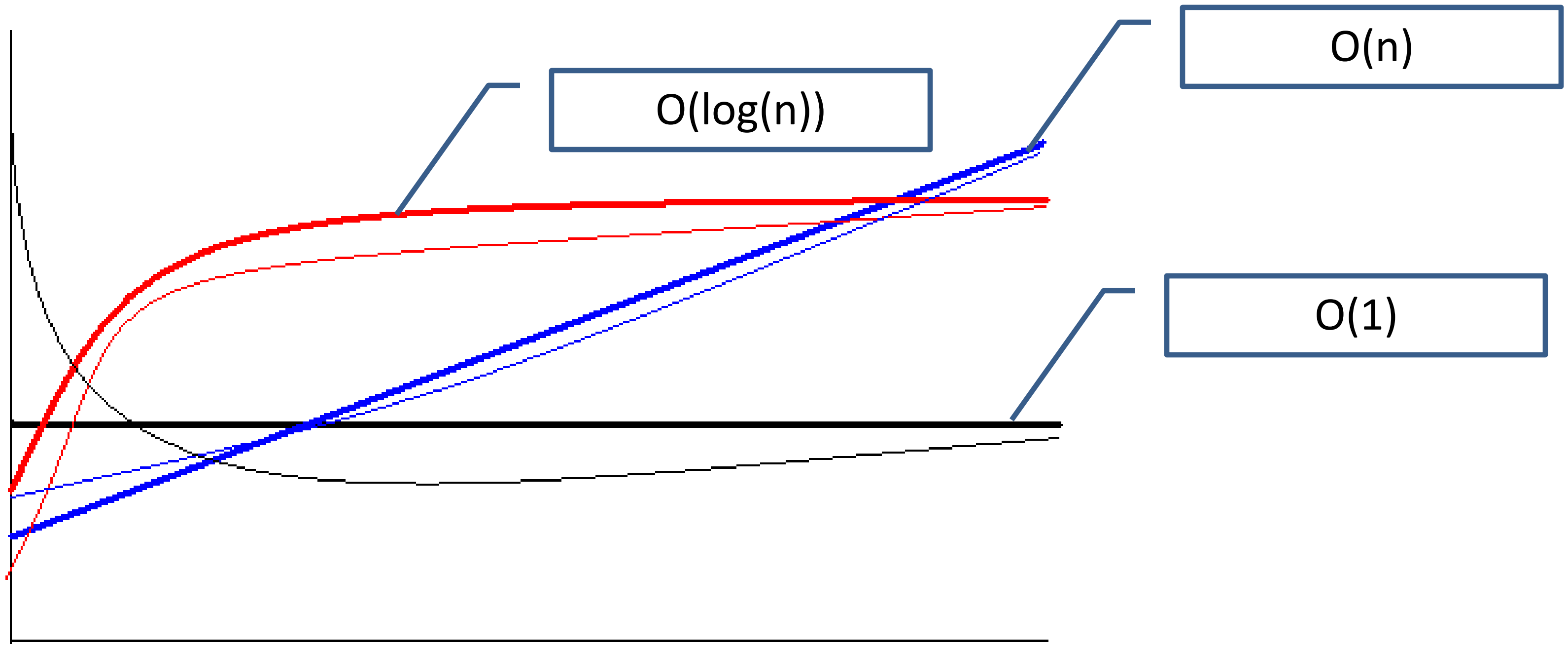
- Algorithm
- Size of data
- Hardware

Algorithm complexity

- Storage complexity (How much memory we need)
- Time complexity (How many operations we need)

O(f(n)) notation

$x(n)$ is $O(f(n))$ if $x(n) \leq C^* f(n)$, C – const $n \rightarrow \infty$



Structure characteristics

- **Theoretical:** Processing algorithm complexity

=

- **Practical:**
  - Memory access pattern
  - Parallelization

Summary

- IR is old ☺
- Main Structure is sparse matrix
- Index = Inverted file
- Speed & Size

Q&A