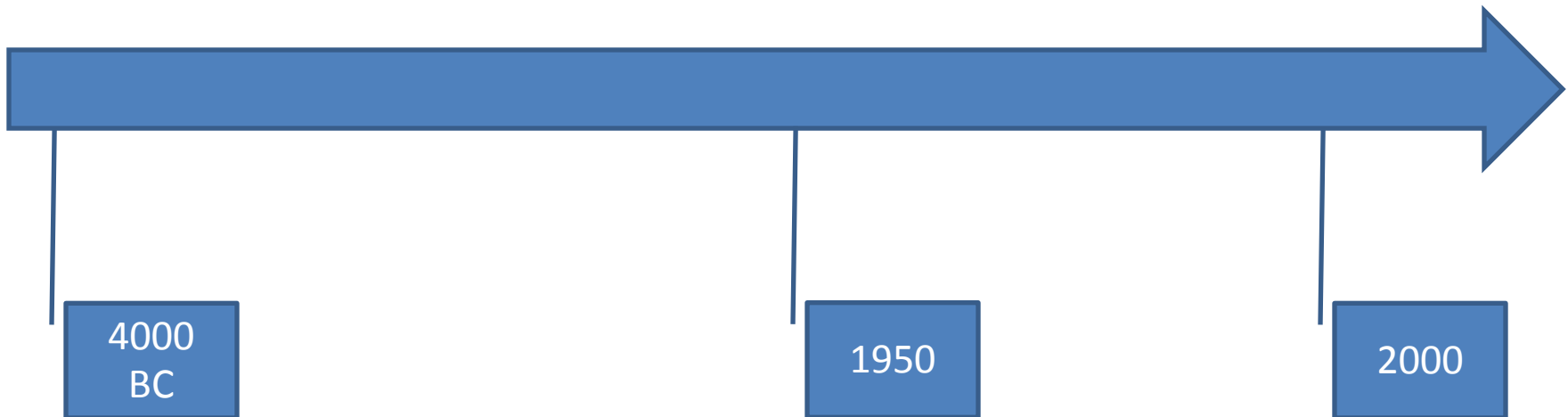
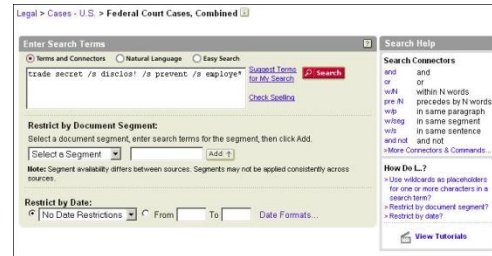


Data structures in Information Retrieval

Max Gubin

mail@maxgubin.com

Information Retrieval History



Information Retrieval Tasks

Types of information:

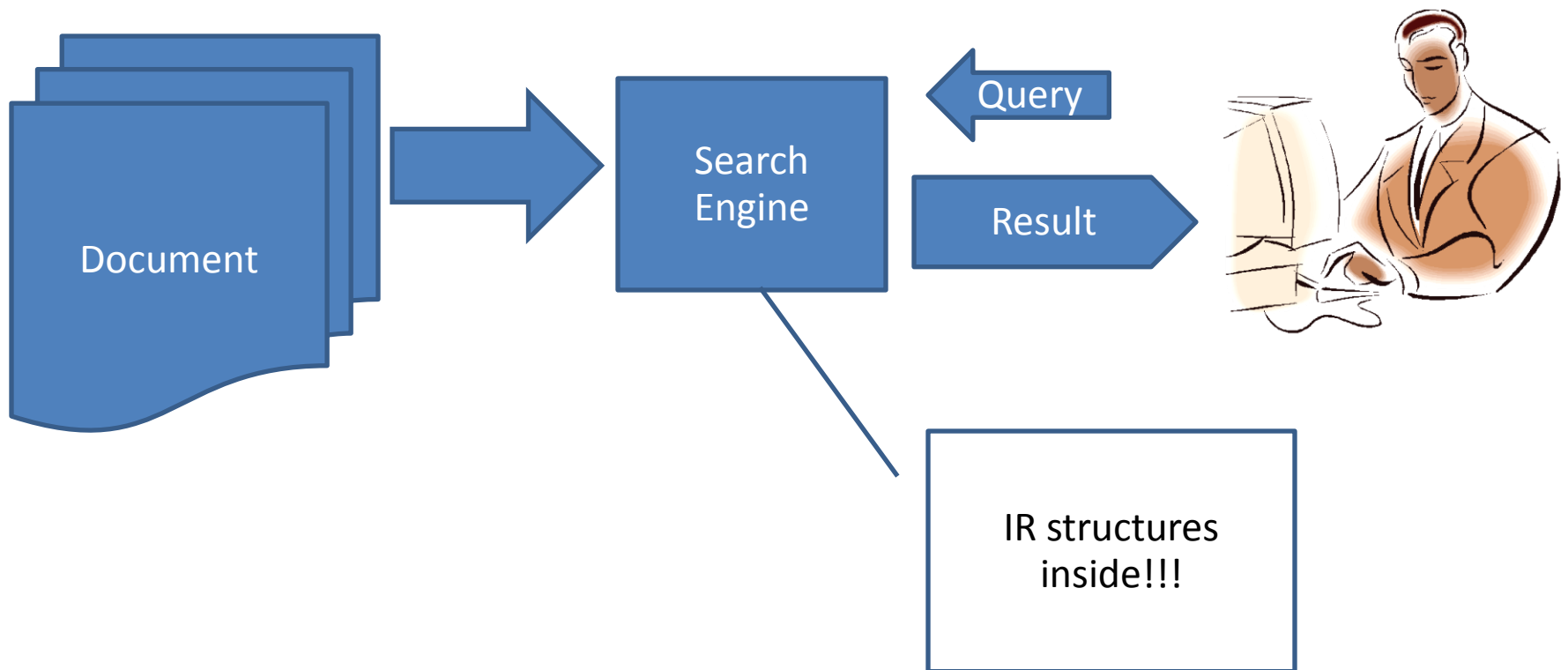
- Text
 - Sound
 - Image
- } Mixes...

Types of tasks:

- Search
 - Classification/clustering
 - Extraction/Summarization
- } Mixes...

Toy project

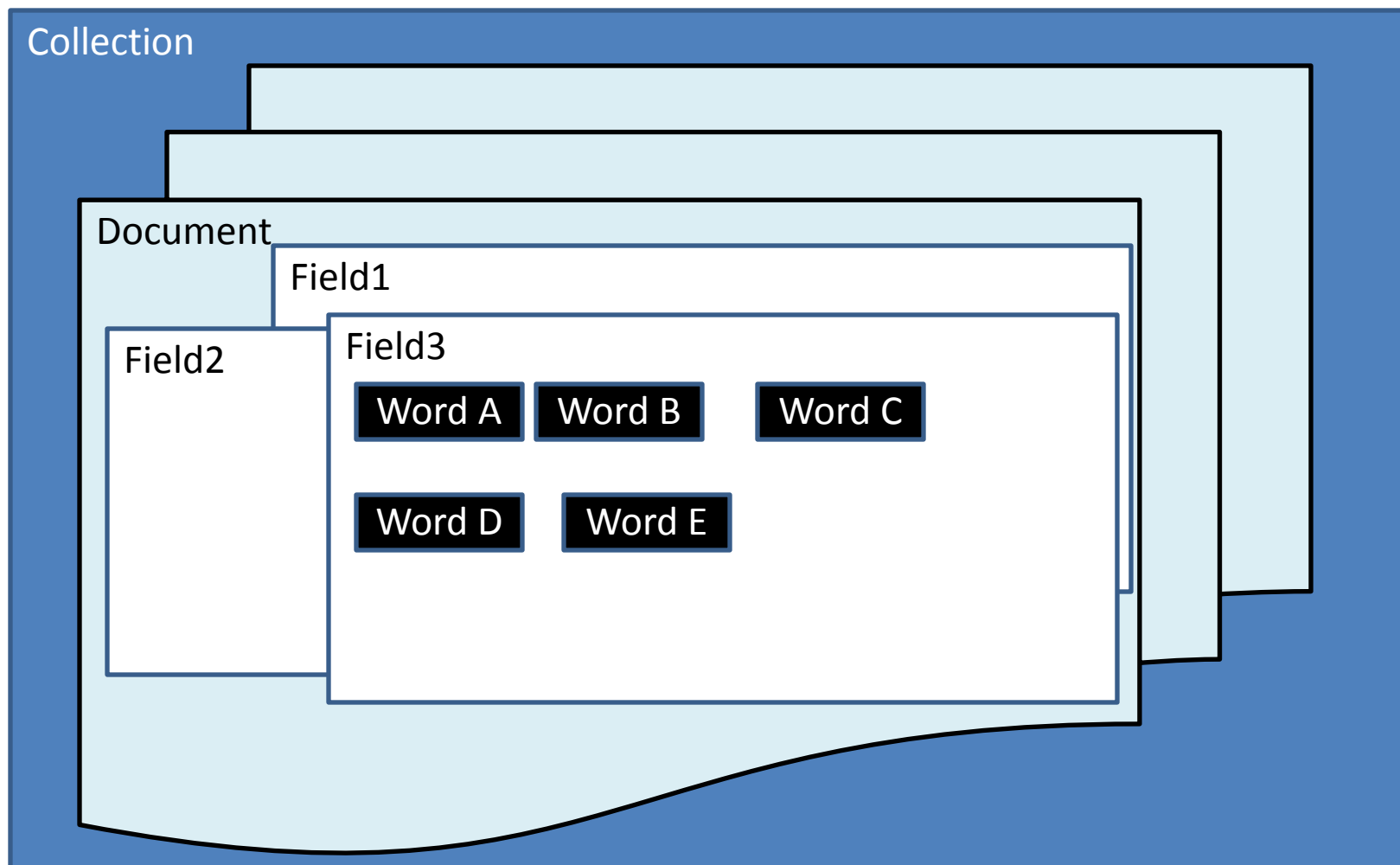
Let's create a toy search engine:



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
Коскоратура «Кодекс»
max@rubin.spb.ru amoro@kodeks.ru

Аннотация

Статья содержит результаты экспериментального исследования влияния различных подходов к обработке форм русского слова на качество информационного поиска. Большинство современных русскоязычных поисковых систем используют нормализованно (линеаризовано) слова, то есть приравнивая различные формы слова к одному поисковому признаку. Считается, что это позволяет заметно улучшить качество поиска. Вместо искомого поиска и нормализации с использованием алгоритмов учета синтаксиса (грамматика), морфологического анализа на основании правил и/или словарей. В проведенных экспериментах использовались ряд общедоступных русскоязычных модулей стеммы и морфологического анализа. Для сравнения качества поиска использовалась метрика F0.91/F1.

Введение

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

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 слов языка во множество форм – нормализованных форм

слов[4]. В литературе данный модуль поисковой системы называют модулем морфологического анализа, нормализатором слов, линеаризатором или стеммером (stemmer). Однако, выполняемому данным модулем, можно представить как отображение $W \rightarrow W'$, где

W – множество всех терминов;

W' – множество всех лемм;

При этом множество лемм имеет мощность множества всех терминов $|W'| \leq |W|$.

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

1. Улучшение полноты поиска. Так как отбираются документы, которые содержат все формы слова, то в результате поиска попадают не только документы со словами в совпадающей с запросом форме, но и другие документы, содержащие различные формы данного слова.
2. Улучшение точности поиска. При использовании синтаксиса алгоритмов поиска и отбора в результате поиска исключаются документы, которые получили избыточный вес, если слово становится ведущим частотным характеристикам 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. Улучшение пользовательского интерфейса. Так как для пользователя частот лемм является найти «все варианты употребления», то в случае отсутствия автоматического расширения слов его приходится пользоваться словарем изучать и использовать формулы или операторы отеченки.
4. Уменьшение размера индексной информации и ускорение обработки запроса. Так как количество лемм меньше количества слов, то индексация происходит в



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

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
Косцюк-Кодеса
max@rubin.spb.ru amoro@kodeks.ru

Аннотация

Статья содержит результаты экспериментального исследования влияния различных подходов к обработке форм русского слова на качество информационного поиска. Большинство современных русскоязычных поисковых систем используют нормализованно (линеаризовано) слова, то есть приравнивая различные формы слова к одному поисковому признаку. Считается, что это позволяет заметно улучшить качество поиска. Известно несколько подходов к нормализации с использованием алгоритмов членения слов (грамматический морфологический анализ на основании правил или словарей). В проведенных экспериментах использовались ряд общедоступных русскоязычных модулей членения и морфологического анализа. Для сравнения качества поиска использовались метрика F0.5@10.

Введение

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

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 слов языка во множество имен – нормализованных форм

слов[4]. В литературе данный модуль поисковой системы называют модулем морфологического анализа, нормализатором слов, линеаризатором или стеммером (stemmer). Обычно, выполняемому данным модулем, можно представить как отображение $W \rightarrow W'$, где

W – множество всех терминов;

W' – множество всех имен.

При этом множество имен меньше множества исходных терминов $|W'| < |W|$.

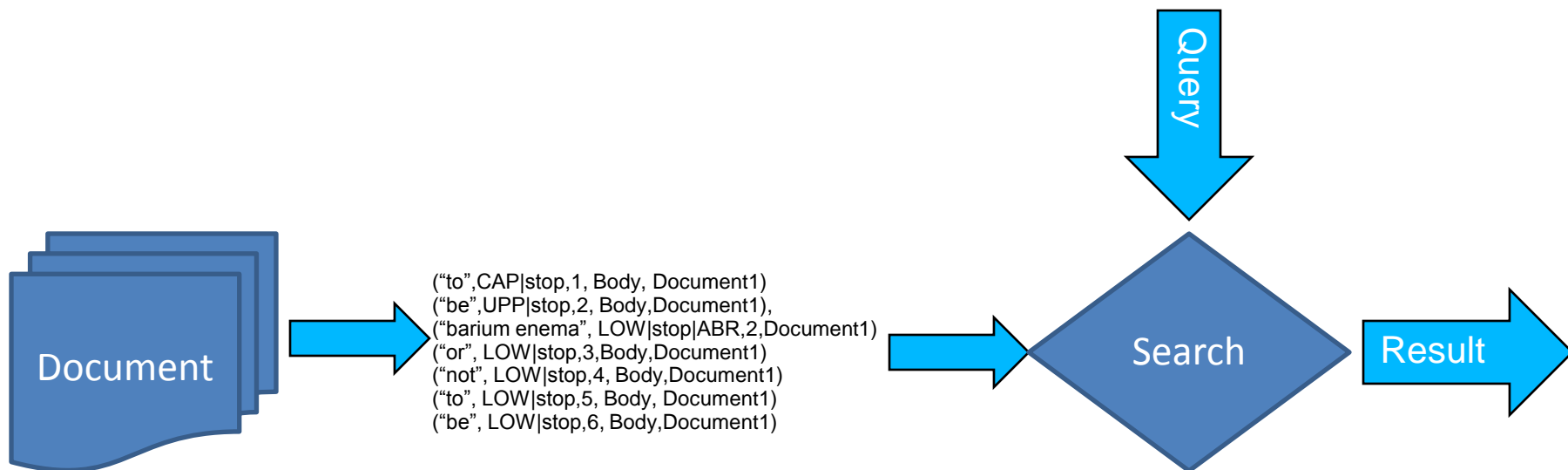
Реализуя данное преобразование, разработчики поисковой системы пытаются достичь следующих целей:

1. Улучшение полноты поиска. Так как отбираются документы, которые содержат все формы слова, то в результате поиска попадают не только документы со словами в совпадающей с запросом форме, но и другие документы, содержащие различные формы данного слова.
2. Улучшение точности поиска. При использовании синтаксически-алгоритмов поиска и сбора в результате поиска несколько документов, которые получили наибольший вес, очень часто становится лучшим частотным характеристикам документов. При этом использование вместо частот слов частоты имен может позволить получить больший вес для релевантных документов и тем самым поместить их во множество отображенных.
3. Улучшение пользовательского интерфейса. Так как для пользователя частотой заданной является найти «все варианты употребления», то в случае отсутствия автоматического расширения слова его приходится пользователю вручную изучать и использовать формулы или операторы отеченки.
4. Уменьшение размера индексной информации и ускорение обработки запроса. Так как количество имен меньше количества слов, то линеаризация 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

Doctable

Document	ID
Hamlet	1
Introduction to...	2
Dive into Python	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

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

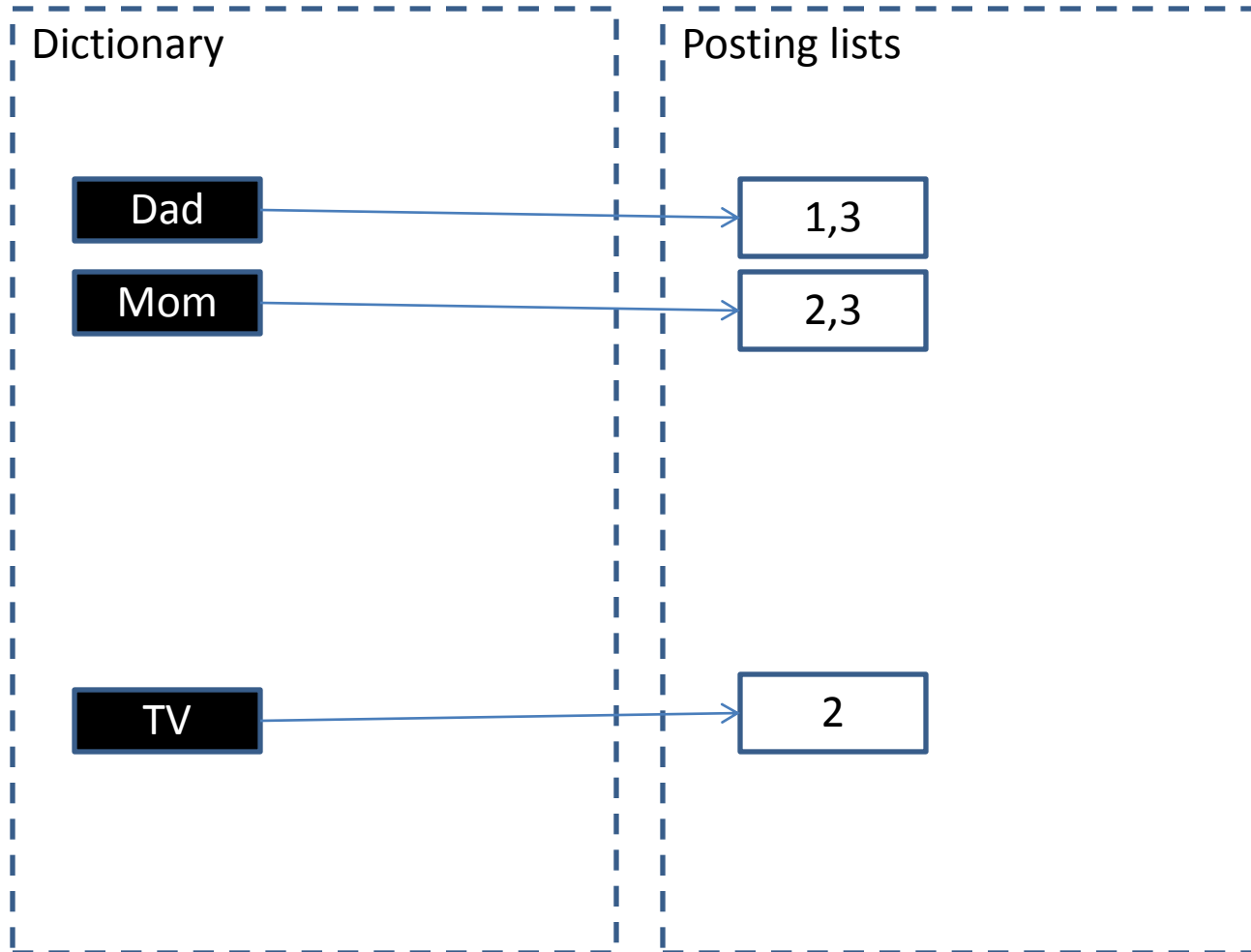
D – 500 000 words in dictionary

|Word/Document matrix| = $D * N = 500 \text{ bln}$

Words in collection = $1 \text{ mln} * 1000 = 1 \text{ bln}$

Only 0.2% elements in the matrix are not 0

Inverted file



Signature file

Signatures for words
(function)

Dad → 00000001

Mom → 00001000

TV → 10000000

watching → 00001000

football → 00001000

Doc Signature = OR words

1	0011000 <u>1</u>
2	0101 <u>1</u> 000
3	1000 <u>1</u> 00 <u>1</u>

Signature file (Search)

Query = "Mom Dad"
q_s = 00001001

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

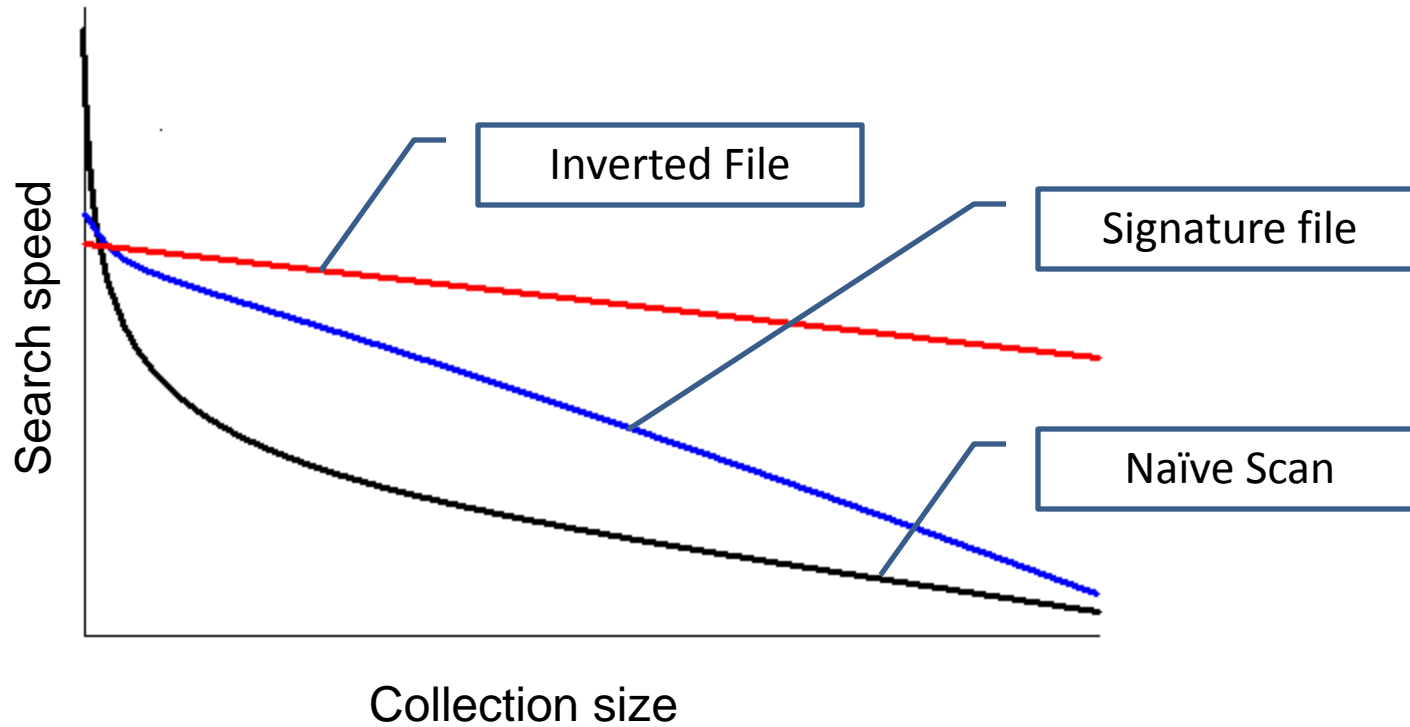
1	0011000 <u>1</u>
2	0101 <u>1</u> 000
3	1000 <u>1</u> 00 <u>1</u>

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



Speed (Size) depends on

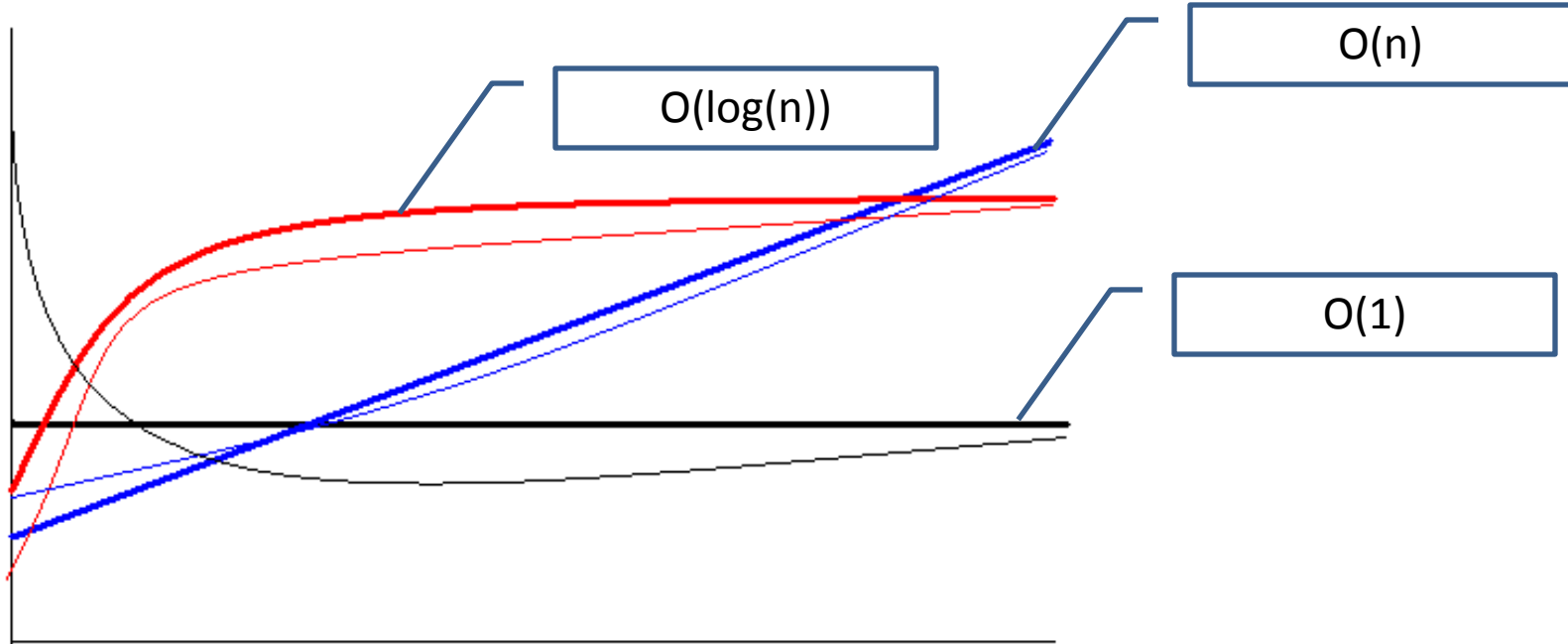
- Algorithm
- Size of data
- Hardware

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 $x(n) \leq C * f(n)$, $C - \text{const } n \rightarrow \infty$



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