Introduction to Information Retrieval

http://informationretrieval.org

IIR 2: The term vocabulary and postings lists

Hinrich Schütze

Center for Information and Language Processing, University of Munich

2014-04-09

Overview

Outline

1. Recap
2. Documents
3. Terms
   - General + Non-English
   - English
4. Skip pointers
5. Phrase queries

Inverted index

For each term t, we store a list of all documents that contain t.

dictionary		postings								
BRUTUS	$\longrightarrow$	1	2	4	11	31	45	173	174	
CAESAR	$\longrightarrow$	1	2	4	5	6	16	57	132	...
CALPURNIA	$\longrightarrow$	2	31	54	101					
⋮										

Intersecting two postings lists

$$\textsc{Brutus} \longrightarrow \boxed{1} \to \boxed{2} \to \boxed{4} \to \boxed{11} \to \boxed{31} \to \boxed{45} \to \boxed{173} \to \boxed{174}$$

$$\textsc{Calpurnia} \longrightarrow \boxed{2} \to \boxed{31} \to \boxed{54} \to \boxed{101}$$

Intersection $\Longrightarrow$

Intersecting two postings lists

BRUTUS $\longrightarrow$ $\boxed{1} \rightarrow \boxed{2} \rightarrow \boxed{4} \rightarrow \boxed{11} \rightarrow \boxed{31} \rightarrow \boxed{45} \rightarrow \boxed{173} \rightarrow \boxed{174}$

CALPURNIA $\longrightarrow$ $\boxed{2} \rightarrow \boxed{31} \rightarrow \boxed{54} \rightarrow \boxed{101}$

Intersection $\Longrightarrow$

Intersecting two postings lists

$$
\begin{array}{lll}
\textsc{Brutus} & \longrightarrow & \boxed{1}\rightarrow\boxed{2}\rightarrow\boxed{4}\rightarrow\boxed{11}\rightarrow\boxed{31}\rightarrow\boxed{45}\rightarrow\boxed{173}\rightarrow\boxed{174} \\
\textsc{Calpurnia} & \longrightarrow & \boxed{2}\rightarrow\boxed{31}\rightarrow\boxed{54}\rightarrow\boxed{101} \\
\text{Intersection} & \Longrightarrow &
\end{array}
$$

Intersecting two postings lists

BRUTUS $\longrightarrow$ $\boxed{1} \rightarrow \boxed{2} \rightarrow \boxed{4} \rightarrow \boxed{11} \rightarrow \boxed{31} \rightarrow \boxed{45} \rightarrow \boxed{173} \rightarrow \boxed{174}$

CALPURNIA $\longrightarrow$ $\boxed{2} \rightarrow \boxed{31} \rightarrow \boxed{54} \rightarrow \boxed{101}$

Intersection $\Longrightarrow$ $\boxed{2}$

Intersecting two postings lists

BRUTUS $\longrightarrow$ $\boxed{1} \rightarrow \boxed{2} \rightarrow \boxed{4} \rightarrow \boxed{11} \rightarrow \boxed{31} \rightarrow \boxed{45} \rightarrow \boxed{173} \rightarrow \boxed{174}$

CALPURNIA $\longrightarrow$ $\boxed{2} \rightarrow \boxed{31} \rightarrow \boxed{54} \rightarrow \boxed{101}$

Intersection $\Longrightarrow$ $\boxed{2}$

Intersecting two postings lists

BRUTUS $\longrightarrow$ $\boxed{1} \rightarrow \boxed{2} \rightarrow \boxed{4} \rightarrow \boxed{11} \rightarrow \boxed{31} \rightarrow \boxed{45} \rightarrow \boxed{173} \rightarrow \boxed{174}$

CALPURNIA $\longrightarrow$ $\boxed{2} \rightarrow \boxed{31} \rightarrow \boxed{54} \rightarrow \boxed{101}$

Intersection $\Longrightarrow$ $\boxed{2}$

Intersecting two postings lists

BRUTUS $\longrightarrow$ $\boxed{1} \rightarrow \boxed{2} \rightarrow \boxed{4} \rightarrow \boxed{11} \rightarrow \boxed{31} \rightarrow \boxed{45} \rightarrow \boxed{173} \rightarrow \boxed{174}$

CALPURNIA $\longrightarrow$ $\boxed{2} \rightarrow \boxed{31} \rightarrow \boxed{54} \rightarrow \boxed{101}$

Intersection $\Longrightarrow$ $\boxed{2}$

Intersecting two postings lists

BRUTUS $\longrightarrow$ $\boxed{1} \rightarrow \boxed{2} \rightarrow \boxed{4} \rightarrow \boxed{11} \rightarrow \boxed{31} \rightarrow \boxed{45} \rightarrow \boxed{173} \rightarrow \boxed{174}$

CALPURNIA $\longrightarrow$ $\boxed{2} \rightarrow \boxed{31} \rightarrow \boxed{54} \rightarrow \boxed{101}$

Intersection $\Longrightarrow$ $\boxed{2} \rightarrow \boxed{31}$

Intersecting two postings lists

BRUTUS	$\longrightarrow$	$\boxed{1} \rightarrow \boxed{2} \rightarrow \boxed{4} \rightarrow \boxed{11} \rightarrow \boxed{31} \rightarrow \boxed{45} \rightarrow \boxed{173} \rightarrow \boxed{174}$
CALPURNIA	$\longrightarrow$	$\boxed{2} \rightarrow \boxed{31} \rightarrow \boxed{54} \rightarrow \boxed{101}$
Intersection	$\Longrightarrow$	$\boxed{2} \rightarrow \boxed{31}$

Intersecting two postings lists

$$\textsc{Brutus} \longrightarrow \boxed{1} \to \boxed{2} \to \boxed{4} \to \boxed{11} \to \boxed{31} \to \boxed{45} \to \boxed{173} \to \boxed{174}$$

$$\textsc{Calpurnia} \longrightarrow \boxed{2} \to \boxed{31} \to \boxed{54} \to \boxed{101}$$

$$\text{Intersection} \Longrightarrow \boxed{2} \to \boxed{31}$$

Intersecting two postings lists

$$\textsc{Brutus} \longrightarrow \boxed{1} \rightarrow \boxed{2} \rightarrow \boxed{4} \rightarrow \boxed{11} \rightarrow \boxed{31} \rightarrow \boxed{45} \rightarrow \boxed{173} \rightarrow \boxed{174}$$

$$\textsc{Calpurnia} \longrightarrow \boxed{2} \rightarrow \boxed{31} \rightarrow \boxed{54} \rightarrow \boxed{101}$$

$$\text{Intersection} \Longrightarrow \boxed{2} \rightarrow \boxed{31}$$

Constructing the inverted index: Sort postings

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

$\Longrightarrow$

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

Westlaw: Example queries

Information need: Information on the legal theories involved in preventing the disclosure of trade secrets by employees formerly employed by a competing company

Query: "trade secret" /s disclos! /s prevent /s employe!

Information need: Requirements for disabled people to be able to access a workplace

Query: disab! /p access! /s work-site work-place (employment /3 place)

Information need: Cases about a host's responsibility for drunk guests

Query: host! /p (responsib! liab!) /p (intoxicat! drunk!) /p guest

Does Google use the Boolean model?

- On Google, the default interpretation of a query [w_1 w_2 $\ldots w_n$] is w_1 AND w_2 AND $\ldots$ AND w_n
- Cases where you get hits that do not contain one of the w_i:
  - anchor text
  - page contains variant of w_i (morphology, spelling correction, synonym)
  - long queries (n large)
  - boolean expression generates very few hits
- Simple Boolean vs. Ranking of result set
  - Simple Boolean retrieval returns matching documents in no particular order.
  - Google (and most well designed Boolean engines) rank the result set – they rank good hits (according to some estimator of relevance) higher than bad hits.

Take-away

Take-away

- Understanding of the basic unit of classical information retrieval systems: words and documents: What is a document, what is a term?

Take-away

- Understanding of the basic unit of classical information retrieval systems: words and documents: What is a document, what is a term?
- Tokenization: how to get from raw text to words (or tokens)

Take-away

- Understanding of the basic unit of classical information retrieval systems: words and documents: What is a document, what is a term?
- Tokenization: how to get from raw text to words (or tokens)
- More complex indexes: skip pointers and phrases

Outline

Documents

- Last lecture: Simple Boolean retrieval system

Documents

- Last lecture: Simple Boolean retrieval system
- Our assumptions were:

Documents

- Last lecture: Simple Boolean retrieval system
- Our assumptions were:
  - We know what a document is.

Documents

- Last lecture: Simple Boolean retrieval system
- Our assumptions were:
  - We know what a document is.
  - We can "machine-read" each document.

Documents

- Last lecture: Simple Boolean retrieval system
- Our assumptions were:
  - We know what a document is.
  - We can "machine-read" each document.
- This can be complex in reality.

Parsing a document

- We need to deal with format and language of each document.

Parsing a document

- We need to deal with format and language of each document.
- What format is it in? pdf, word, excel, html etc.

Parsing a document

- We need to deal with format and language of each document.
- What format is it in? pdf, word, excel, html etc.
- What language is it in?

Parsing a document

- We need to deal with format and language of each document.
- What format is it in? pdf, word, excel, html etc.
- What language is it in?
- What character set is in use?

Parsing a document

- We need to deal with format and language of each document.
- What format is it in? pdf, word, excel, html etc.
- What language is it in?
- What character set is in use?
- Each of these is a classification problem, which we will study later in this course (IIR 13).

Parsing a document

- We need to deal with format and language of each document.
- What format is it in? pdf, word, excel, html etc.
- What language is it in?
- What character set is in use?
- Each of these is a classification problem, which we will study later in this course (IIR 13).
- Alternative: use heuristics

Format/Language: Complications

- A single index usually contains terms of several languages.

Format/Language: Complications

- A single index usually contains terms of several languages.
- Sometimes a document or its components contain multiple languages/formats.

Format/Language: Complications

- A single index usually contains terms of several languages.
- Sometimes a document or its components contain multiple languages/formats.
  - French email with Spanish pdf attachment

Format/Language: Complications

- A single index usually contains terms of several languages.
- Sometimes a document or its components contain multiple languages/formats.
  - French email with Spanish pdf attachment
- What is the document unit for indexing?

Format/Language: Complications

- A single index usually contains terms of several languages.
- Sometimes a document or its components contain multiple languages/formats.
  - French email with Spanish pdf attachment
- What is the document unit for indexing?
- A file?

Format/Language: Complications

- A single index usually contains terms of several languages.
- Sometimes a document or its components contain multiple languages/formats.
  - French email with Spanish pdf attachment
- What is the document unit for indexing?
- A file?
- An email?

Format/Language: Complications

- A single index usually contains terms of several languages.
- Sometimes a document or its components contain multiple languages/formats.
  - French email with Spanish pdf attachment
- What is the document unit for indexing?
- A file?
- An email?
- An email with 5 attachments?

Format/Language: Complications

- A single index usually contains terms of several languages.
- Sometimes a document or its components contain multiple languages/formats.
  - French email with Spanish pdf attachment
- What is the document unit for indexing?
- A file?
- An email?
- An email with 5 attachments?
- A group of files (ppt or latex in HTML)?

Format/Language: Complications

- A single index usually contains terms of several languages.
- Sometimes a document or its components contain multiple languages/formats.
  - French email with Spanish pdf attachment
- What is the document unit for indexing?
- A file?
- An email?
- An email with 5 attachments?
- A group of files (ppt or latex in HTML)?
- Upshot: Answering the question "what is a document?" is not trivial and requires some design decisions.

Format/Language: Complications

- A single index usually contains terms of several languages.
- Sometimes a document or its components contain multiple languages/formats.
  - French email with Spanish pdf attachment
- What is the document unit for indexing?
- A file?
- An email?
- An email with 5 attachments?
- A group of files (ppt or latex in HTML)?
- Upshot: Answering the question "what is a document?" is not trivial and requires some design decisions.
- Also: XML

Outline

Outline

Definitions

- Word – A delimited string of characters as it appears in the text.

Definitions

- Word – A delimited string of characters as it appears in the text.
- Term – A "normalized" word (case, morphology, spelling etc); an equivalence class of words.

Definitions

- Word – A delimited string of characters as it appears in the text.
- Term – A “normalized” word (case, morphology, spelling etc); an equivalence class of words.
- Token – An instance of a word or term occurring in a document.

Definitions

- Word – A delimited string of characters as it appears in the text.
- Term – A "normalized" word (case, morphology, spelling etc); an equivalence class of words.
- Token – An instance of a word or term occurring in a document.
- Type – The same as a term in most cases: an equivalence class of tokens.

Normalization

- Need to "normalize" words in indexed text as well as query terms into the same form.

Normalization

- Need to “normalize” words in indexed text as well as query terms into the same form.
- Example: We want to match *U.S.A.* and *USA*

Normalization

- Need to “normalize” words in indexed text as well as query terms into the same form.
- Example: We want to match *U.S.A.* and *USA*
- We most commonly implicitly define equivalence classes of terms.

Normalization

- Need to "normalize" words in indexed text as well as query terms into the same form.
- Example: We want to match *U.S.A.* and *USA*
- We most commonly implicitly define equivalence classes of terms.
- Alternatively: do asymmetric expansion

Normalization

- Need to “normalize” words in indexed text as well as query terms into the same form.
- Example: We want to match *U.S.A.* and *USA*
- We most commonly implicitly define equivalence classes of terms.
- Alternatively: do asymmetric expansion
  - window $\rightarrow$ window, windows

Normalization

- Need to "normalize" words in indexed text as well as query terms into the same form.
- Example: We want to match *U.S.A.* and *USA*
- We most commonly implicitly define equivalence classes of terms.
- Alternatively: do asymmetric expansion
  - window $\rightarrow$ window, windows
  - windows $\rightarrow$ Windows, windows

Normalization

- Need to "normalize" words in indexed text as well as query terms into the same form.
- Example: We want to match *U.S.A.* and *USA*
- We most commonly implicitly define equivalence classes of terms.
- Alternatively: do asymmetric expansion
  - window $\rightarrow$ window, windows
  - windows $\rightarrow$ Windows, windows
  - Windows (no expansion)

Normalization

- Need to "normalize" words in indexed text as well as query terms into the same form.
- Example: We want to match *U.S.A.* and *USA*
- We most commonly implicitly define equivalence classes of terms.
- Alternatively: do asymmetric expansion
  - window $\rightarrow$ window, windows
  - windows $\rightarrow$ Windows, windows
  - Windows (no expansion)
- More powerful, but less efficient

Normalization

- Need to "normalize" words in indexed text as well as query terms into the same form.
- Example: We want to match *U.S.A.* and *USA*
- We most commonly implicitly define equivalence classes of terms.
- Alternatively: do asymmetric expansion
  - window → window, windows
  - windows → Windows, windows
  - Windows (no expansion)
- More powerful, but less efficient
- Why don't you want to put *window*, *Window*, *windows*, and *Windows* in the same equivalence class?

Normalization: Other languages

- Normalization and language detection interact.

Normalization: Other languages

- Normalization and language detection interact.
- *PETER WILL NICHT MIT.* $\rightarrow$ MIT $=$ mit

Normalization: Other languages

- Normalization and language detection interact.
- *PETER WILL NICHT MIT.* $\rightarrow$ MIT $=$ mit
- *He got his PhD from MIT.* $\rightarrow$ MIT $\neq$ mit

Tokenization: Recall construction of inverted index

- Input:

  Friends, Romans, countrymen. | So let it be with Caesar | . . .

Tokenization: Recall construction of inverted index

- Input:

  | Friends, Romans, countrymen. | | So let it be with Caesar | ...

- Output:

  | friend | | roman | | countryman | | so | ...

Tokenization: Recall construction of inverted index

- Input:

  | Friends, Romans, countrymen. | | So let it be with Caesar | | . . .

- Output:

  | friend | | roman | | countryman | | so | | . . .

- Each token is a candidate for a postings entry.

Tokenization: Recall construction of inverted index

- Input:

Friends, Romans, countrymen.	So let it be with Caesar	. . .

- Output:

friend	roman	countryman	so	. . .

- Each token is a candidate for a postings entry.
- What are valid tokens to emit?

Exercises

In June, the dog likes to chase the cat in the barn. – How many word tokens? How many word types?

Why tokenization is difficult – even in English. Tokenize: *Mr. O'Neill thinks that the boys' stories about Chile's capital aren't amusing.*

Tokenization problems: One word or two? (or several)

- Hewlett-Packard

Tokenization problems: One word or two? (or several)

- Hewlett-Packard
- State-of-the-art

Tokenization problems: One word or two? (or several)

- Hewlett-Packard
- State-of-the-art
- co-education

Tokenization problems: One word or two? (or several)

- Hewlett-Packard
- State-of-the-art
- co-education
- the hold-him-back-and-drag-him-away maneuver

Tokenization problems: One word or two? (or several)

- Hewlett-Packard
- State-of-the-art
- co-education
- the hold-him-back-and-drag-him-away maneuver
- data base

Tokenization problems: One word or two? (or several)

- Hewlett-Packard
- State-of-the-art
- co-education
- the hold-him-back-and-drag-him-away maneuver
- data base
- San Francisco

Tokenization problems: One word or two? (or several)

- Hewlett-Packard
- State-of-the-art
- co-education
- the hold-him-back-and-drag-him-away maneuver
- data base
- San Francisco
- Los Angeles-based company

Tokenization problems: One word or two? (or several)

- Hewlett-Packard
- State-of-the-art
- co-education
- the hold-him-back-and-drag-him-away maneuver
- data base
- San Francisco
- Los Angeles-based company
- cheap San Francisco-Los Angeles fares

Tokenization problems: One word or two? (or several)

- Hewlett-Packard
- State-of-the-art
- co-education
- the hold-him-back-and-drag-him-away maneuver
- data base
- San Francisco
- Los Angeles-based company
- cheap San Francisco-Los Angeles fares
- York University vs. New York University

Numbers

- 3/20/91

Numbers

- 3/20/91
- 20/3/91

Numbers

- 3/20/91
- 20/3/91
- Mar 20, 1991

Numbers

- 3/20/91
- 20/3/91
- Mar 20, 1991
- B-52

Numbers

- 3/20/91
- 20/3/91
- Mar 20, 1991
- B-52
- 100.2.86.144

Numbers

- 3/20/91
- 20/3/91
- Mar 20, 1991
- B-52
- 100.2.86.144
- (800) 234-2333

Numbers

- 3/20/91
- 20/3/91
- Mar 20, 1991
- B-52
- 100.2.86.144
- (800) 234-2333
- 800.234.2333

Numbers

- 3/20/91
- 20/3/91
- Mar 20, 1991
- B-52
- 100.2.86.144
- (800) 234-2333
- 800.234.2333
- Older IR systems may not index numbers . . .

Numbers

- 3/20/91
- 20/3/91
- Mar 20, 1991
- B-52
- 100.2.86.144
- (800) 234-2333
- 800.234.2333
- Older IR systems may not index numbers . . .
- . . . but generally it's a useful feature.

Numbers

- 3/20/91
- 20/3/91
- Mar 20, 1991
- B-52
- 100.2.86.144
- (800) 234-2333
- 800.234.2333
- Older IR systems may not index numbers . . .
- . . . but generally it's a useful feature.
- Google example

Chinese: No whitespace

莎拉波娃**现**在居住在美国**东**南部的佛**罗**里**达**。今年４月９日，莎拉波娃在美国第一大城市**纽约**度过了１８**岁**生日。生日派**对**上，莎拉波娃露出了甜美的微笑。

Ambiguous segmentation in Chinese

和尚

The two characters can be treated as one word meaning 'monk' or as a sequence of two words meaning 'and' and 'still'.

Other cases of "no whitespace"

- Compounds in Dutch, German, Swedish

Other cases of “no whitespace”

- Compounds in Dutch, German, Swedish
- Computerlinguistik $\rightarrow$ Computer $+$ Linguistik

Other cases of “no whitespace”

- Compounds in Dutch, German, Swedish
- Computerlinguistik $\rightarrow$ Computer $+$ Linguistik
- Lebensversicherungsgesellschaftsangestellter

Other cases of "no whitespace"

- Compounds in Dutch, German, Swedish
- Computerlinguistik $\rightarrow$ Computer + Linguistik
- Lebensversicherungsgesellschaftsangestellter
- $\rightarrow$ leben + versicherung + gesellschaft + angestellter

Other cases of "no whitespace"

- Compounds in Dutch, German, Swedish
- Computerlinguistik $\rightarrow$ Computer + Linguistik
- Lebensversicherungsgesellschaftsangestellter
- $\rightarrow$ leben + versicherung + gesellschaft + angestellter
- Inuit: tusaatsiarunnanngittualuujunga (I can't hear very well.)

Other cases of “no whitespace”

- Compounds in Dutch, German, Swedish
- Computerlinguistik $\rightarrow$ Computer $+$ Linguistik
- Lebensversicherungsgesellschaftsangestellter
- $\rightarrow$ leben $+$ versicherung $+$ gesellschaft $+$ angestellter
- Inuit: tusaatsiarunnanngittualuujunga (I can’t hear very well.)
- Many other languages with segmentation difficulties: Finnish, Urdu, ...

Japanese

ノーベル平和賞を受賞したワンガリ・マータイさんが名誉会長を務めるＭＯＴＴＡＩＮＡＩキャンペーンの一環として、毎日新聞社とマガジンハウスは「私 の、もったいない」を募集します。皆様が日ごろ「もったいない」と感じて実践していることや、それにまつわるエピソードを８００字以内の文章にまとめ、簡 単な写真、イラスト、図などを添えて１０月２０日までにお送りください。大賞受賞者には、５０万円相当の旅行券とエコ製品２点の副賞が贈られます。

Japanese

ノーベル平和賞を受賞したワンガリ・マータイさんが名誉会長を務めるＭＯＴＴＡＩＮＡＩキャンペーンの一環として、毎日新聞社とマガジンハウスは「私 の、もったいない」を募集します。皆様が日ごろ「もったいない」と感じて実践していることや、それにまつわるエピソードを８００字以内の文章にまとめ、簡 単な写真、イラスト、図などを添えて１０月２０日までにお送りください。大賞受賞者には、５０万円相当の旅行券とエコ製品２点の副賞が贈られます。

4 different "alphabets": Chinese characters, hiragana syllabary for inflectional endings and function words, katakana syllabary for transcription of foreign words and other uses, and latin. No spaces (as in Chinese).

Japanese

ノーベル平和賞を受賞したワンガリ・マータイさんが名誉会長を務めるＭＯＴＴＡＩＮＡＩキャンペーンの一環として、毎日新聞社とマガジンハウスは「私 の、もったいない」を募集します。皆様が日ごろ「もったいない」と感じて実践していることや、それにまつわるエピソードを８００字以内の文章にまとめ、簡 単な写真、イラスト、図などを添えて１０月２０日までにお送りください。大賞受賞者には、５０万円相当の旅行券とエコ製品２点の副賞が贈られます。

4 different "alphabets": Chinese characters, hiragana syllabary for inflectional endings and function words, katakana syllabary for transcription of foreign words and other uses, and latin. No spaces (as in Chinese).

End user can express query entirely in hiragana!

Arabic script

ك ِ ت ا ب ٌ ⇐ كِتَابٌ

un b ā t i k

/kitābun/ *‘a book’*

Arabic script: Bidirectionality

استقلت الجزائر في سنة 1962 بعد 132 عاما من الاحتلال الفرنسي.

← → ← → ← START

'Algeria achieved its independence in 1962 after 132 years of French occupation.'

Arabic script: Bidirectionality

استقلت الجزائر في سنة 1962 بعد 132 عاما من الاحتلال الفرنسي.

$\leftarrow \rightarrow \quad \leftarrow \rightarrow \qquad\qquad \leftarrow$ START

'Algeria achieved its independence in 1962 after 132 years of French occupation.'

Bidirectionality is not a problem if text is coded in Unicode.

Accents and diacritics

- Accents: résumé vs. resume (simple omission of accent)

Accents and diacritics

- Accents: résumé vs. resume (simple omission of accent)
- Umlauts: Universität vs. Universitaet (substitution with special letter sequence "ae")

Accents and diacritics

- Accents: résumé vs. resume (simple omission of accent)
- Umlauts: Universität vs. Universitaet (substitution with special letter sequence "ae")
- Most important criterion: How are users likely to write their queries for these words?

Accents and diacritics

- Accents: résumé vs. resume (simple omission of accent)
- Umlauts: Universität vs. Universitaet (substitution with special letter sequence "ae")
- Most important criterion: How are users likely to write their queries for these words?
- Even in languages that standardly have accents, users often do not type them. (Polish?)

Outline

Case folding

- Reduce all letters to lower case

Case folding

- Reduce all letters to lower case
- Even though case can be semantically meaningful

Case folding

- Reduce all letters to lower case
- Even though case can be semantically meaningful
    - capitalized words in mid-sentence

Case folding

- Reduce all letters to lower case
- Even though case can be semantically meaningful
  - capitalized words in mid-sentence
  - MIT vs. mit

Case folding

- Reduce all letters to lower case
- Even though case can be semantically meaningful
  - capitalized words in mid-sentence
  - MIT vs. mit
  - Fed vs. fed

Case folding

- Reduce all letters to lower case
- Even though case can be semantically meaningful
  - capitalized words in mid-sentence
  - MIT vs. mit
  - Fed vs. fed
  - ...

Case folding

- Reduce all letters to lower case
- Even though case can be semantically meaningful
  - capitalized words in mid-sentence
  - MIT vs. mit
  - Fed vs. fed
  - ...
- It's often best to lowercase everything since users will use lowercase regardless of correct capitalization.

Stop words

- stop words = extremely common words which would appear to be of little value in helping select documents matching a user need

Stop words

- stop words = extremely common words which would appear to be of little value in helping select documents matching a user need
- Examples: *a, an, and, are, as, at, be, by, for, from, has, he, in, is, it, its, of, on, that, the, to, was, were, will, with*

Stop words

- stop words = extremely common words which would appear to be of little value in helping select documents matching a user need
- Examples: *a, an, and, are, as, at, be, by, for, from, has, he, in, is, it, its, of, on, that, the, to, was, were, will, with*
- Stop word elimination used to be standard in older IR systems.

Stop words

- stop words = extremely common words which would appear to be of little value in helping select documents matching a user need
- Examples: *a, an, and, are, as, at, be, by, for, from, has, he, in, is, it, its, of, on, that, the, to, was, were, will, with*
- Stop word elimination used to be standard in older IR systems.
- But you need stop words for phrase queries, e.g. "King of Denmark"

Stop words

- stop words = extremely common words which would appear to be of little value in helping select documents matching a user need
- Examples: *a, an, and, are, as, at, be, by, for, from, has, he, in, is, it, its, of, on, that, the, to, was, were, will, with*
- Stop word elimination used to be standard in older IR systems.
- But you need stop words for phrase queries, e.g. "King of Denmark"
- Most web search engines index stop words.

More equivalence classing

- Soundex: IIR 3 (phonetic equivalence, Muller = Mueller)

More equivalence classing

- Soundex: IIR 3 (phonetic equivalence, Muller = Mueller)
- Thesauri: IIR 9 (semantic equivalence, car = automobile)

Lemmatization

- Reduce inflectional/variant forms to base form

Lemmatization

- Reduce inflectional/variant forms to base form
- Example: *am*, *are*, *is* $\rightarrow$ *be*

Lemmatization

- Reduce inflectional/variant forms to base form
- Example: *am, are, is* $\rightarrow$ *be*
- Example: *car, cars, car's, cars'* $\rightarrow$ *car*

Lemmatization

- Reduce inflectional/variant forms to base form
- Example: *am, are, is* $\rightarrow$ *be*
- Example: *car, cars, car's, cars'* $\rightarrow$ *car*
- Example: *the boy's cars are different colors* $\rightarrow$ *the boy car be different color*

Lemmatization

- Reduce inflectional/variant forms to base form
- Example: *am, are, is* $\rightarrow$ *be*
- Example: *car, cars, car's, cars'* $\rightarrow$ *car*
- Example: *the boy's cars are different colors* $\rightarrow$ *the boy car be different color*
- Lemmatization implies doing "proper" reduction to dictionary headword form (the lemma).

Lemmatization

- Reduce inflectional/variant forms to base form
- Example: *am, are, is* $\rightarrow$ *be*
- Example: *car, cars, car's, cars'* $\rightarrow$ *car*
- Example: *the boy's cars are different colors* $\rightarrow$ *the boy car be different color*
- Lemmatization implies doing "proper" reduction to dictionary headword form (the lemma).
- Inflectional morphology (*cutting* $\rightarrow$ *cut*) vs. derivational morphology (*destruction* $\rightarrow$ *destroy*)

Stemming

- Definition of stemming: Crude heuristic process that chops off the ends of words in the hope of achieving what "principled" lemmatization attempts to do with a lot of linguistic knowledge.

Stemming

- Definition of stemming: Crude heuristic process that chops off the ends of words in the hope of achieving what "principled" lemmatization attempts to do with a lot of linguistic knowledge.
- Language dependent

Stemming

- Definition of stemming: Crude heuristic process that chops off the ends of words in the hope of achieving what "principled" lemmatization attempts to do with a lot of linguistic knowledge.
- Language dependent
- Often inflectional and derivational

Stemming

- Definition of stemming: Crude heuristic process that chops off the ends of words in the hope of achieving what “principled” lemmatization attempts to do with a lot of linguistic knowledge.
- Language dependent
- Often inflectional and derivational
- Example for derivational: *automate, automatic, automation* all reduce to *automat*

Porter algorithm

- Most common algorithm for stemming English

Porter algorithm

- Most common algorithm for stemming English
- Results suggest that it is at least as good as other stemming options

Porter algorithm

- Most common algorithm for stemming English
- Results suggest that it is at least as good as other stemming options
- Conventions + 5 phases of reductions

Porter algorithm

- Most common algorithm for stemming English
- Results suggest that it is at least as good as other stemming options
- Conventions + 5 phases of reductions
- Phases are applied sequentially

Porter algorithm

- Most common algorithm for stemming English
- Results suggest that it is at least as good as other stemming options
- Conventions + 5 phases of reductions
- Phases are applied sequentially
- Each phase consists of a set of commands.

Porter algorithm

- Most common algorithm for stemming English
- Results suggest that it is at least as good as other stemming options
- Conventions + 5 phases of reductions
- Phases are applied sequentially
- Each phase consists of a set of commands.
  - Sample command: Delete final *ement* if what remains is longer than 1 character

Porter algorithm

- Most common algorithm for stemming English
- Results suggest that it is at least as good as other stemming options
- Conventions + 5 phases of reductions
- Phases are applied sequentially
- Each phase consists of a set of commands.
    - Sample command: Delete final *ement* if what remains is longer than 1 character
    - replacement → replac

Porter algorithm

- Most common algorithm for stemming English
- Results suggest that it is at least as good as other stemming options
- Conventions + 5 phases of reductions
- Phases are applied sequentially
- Each phase consists of a set of commands.
  - Sample command: Delete final *ement* if what remains is longer than 1 character
  - replacement → replac
  - cement → cement

Porter algorithm

- Most common algorithm for stemming English
- Results suggest that it is at least as good as other stemming options
- Conventions + 5 phases of reductions
- Phases are applied sequentially
- Each phase consists of a set of commands.
  - Sample command: Delete final *ement* if what remains is longer than 1 character
  - replacement → replac
  - cement → cement
- Sample convention: Of the rules in a compound command, select the one that applies to the longest suffix.

Porter stemmer: A few rules

Rule			**Example**		
SSES	$\rightarrow$	SS	caresses	$\rightarrow$	caress
IES	$\rightarrow$	I	ponies	$\rightarrow$	poni
SS	$\rightarrow$	SS	caress	$\rightarrow$	caress
S	$\rightarrow$		cats	$\rightarrow$	cat

Three stemmers: A comparison

Sample text: Such an analysis can reveal features that are not easily visible from the variations in the individual genes and can lead to a picture of expression that is more biologically transparent and accessible to interpretation

Porter stemmer: such an analysi can reveal featur that ar not easili visibl from the variat in the individu gene and can lead to a pictur of express that is more biolog transpar and access to interpret

Lovins stemmer: such an analys can reve featur that ar not eas vis from th vari in th individu gen and can lead to a pictur of expres that is mor biolog transpar and acces to interpres

Paice stemmer: such an analys can rev feat that are not easy vis from the vary in the individ gen and can lead to a pict of express that is mor biolog transp and access to interpret

Does stemming improve effectiveness?

- In general, stemming increases effectiveness for some queries, and decreases effectiveness for others.

Does stemming improve effectiveness?

- In general, stemming increases effectiveness for some queries, and decreases effectiveness for others.
- Queries where stemming is likely to help: [tartan sweaters], [sightseeing tour san francisco]

Does stemming improve effectiveness?

- In general, stemming increases effectiveness for some queries, and decreases effectiveness for others.
- Queries where stemming is likely to help: [tartan sweaters], [sightseeing tour san francisco]
- (equivalence classes: {sweater,sweaters}, {tour,tours})

Does stemming improve effectiveness?

- In general, stemming increases effectiveness for some queries, and decreases effectiveness for others.
- Queries where stemming is likely to help: [tartan sweaters], [sightseeing tour san francisco]
- (equivalence classes: {sweater,sweaters}, {tour,tours})
- Porter Stemmer equivalence class *oper* contains all of *operate operating operates operation operative operatives operational*.

Does stemming improve effectiveness?

- In general, stemming increases effectiveness for some queries, and decreases effectiveness for others.
- Queries where stemming is likely to help: [tartan sweaters], [sightseeing tour san francisco]
- (equivalence classes: {sweater,sweaters}, {tour,tours})
- Porter Stemmer equivalence class *oper* contains all of *operate operating operates operation operative operatives operational*.
- Queries where stemming hurts: [operational AND research], [operating AND system], [operative AND dentistry]

Exercise: What does Google do?

- Stop words
- Normalization
- Tokenization
- Lowercasing
- Stemming
- Non-latin alphabets
- Umlauts
- Compounds
- Numbers

Outline

Recall basic intersection algorithm

BRUTUS $\longrightarrow$ $\boxed{1} \rightarrow \boxed{2} \rightarrow \boxed{4} \rightarrow \boxed{11} \rightarrow \boxed{31} \rightarrow \boxed{45} \rightarrow \boxed{173} \rightarrow \boxed{174}$

CALPURNIA $\longrightarrow$ $\boxed{2} \rightarrow \boxed{31} \rightarrow \boxed{54} \rightarrow \boxed{101}$

Intersection $\Longrightarrow$

Recall basic intersection algorithm

Brutus $\longrightarrow$ $\boxed{1} \rightarrow \boxed{2} \rightarrow \boxed{4} \rightarrow \boxed{11} \rightarrow \boxed{31} \rightarrow \boxed{45} \rightarrow \boxed{173} \rightarrow \boxed{174}$

Calpurnia $\longrightarrow$ $\boxed{2} \rightarrow \boxed{31} \rightarrow \boxed{54} \rightarrow \boxed{101}$

Intersection $\Longrightarrow$

Recall basic intersection algorithm

BRUTUS	$\longrightarrow$	$\boxed{1} \rightarrow \boxed{2} \rightarrow \boxed{4} \rightarrow \boxed{11} \rightarrow \boxed{31} \rightarrow \boxed{45} \rightarrow \boxed{173} \rightarrow \boxed{174}$
CALPURNIA	$\longrightarrow$	$\boxed{2} \rightarrow \boxed{31} \rightarrow \boxed{54} \rightarrow \boxed{101}$
Intersection	$\Longrightarrow$	

Recall basic intersection algorithm

BRUTUS ⟶ $\boxed{1} \rightarrow \boxed{2} \rightarrow \boxed{4} \rightarrow \boxed{11} \rightarrow \boxed{31} \rightarrow \boxed{45} \rightarrow \boxed{173} \rightarrow \boxed{174}$

CALPURNIA ⟶ $\boxed{2} \rightarrow \boxed{31} \rightarrow \boxed{54} \rightarrow \boxed{101}$

Intersection ⟹ $\boxed{2}$

Recall basic intersection algorithm

BRUTUS	$\longrightarrow$	$\boxed{1} \to \boxed{2} \to \boxed{4} \to \boxed{11} \to \boxed{31} \to \boxed{45} \to \boxed{173} \to \boxed{174}$
CALPURNIA	$\longrightarrow$	$\boxed{2} \to \boxed{31} \to \boxed{54} \to \boxed{101}$
Intersection	$\Longrightarrow$	$\boxed{2}$

Recall basic intersection algorithm

BRUTUS $\longrightarrow$ $\boxed{1} \to \boxed{2} \to \boxed{4} \to \boxed{11} \to \boxed{31} \to \boxed{45} \to \boxed{173} \to \boxed{174}$

CALPURNIA $\longrightarrow$ $\boxed{2} \to \boxed{31} \to \boxed{54} \to \boxed{101}$

Intersection $\Longrightarrow$ $\boxed{2}$

Recall basic intersection algorithm

BRUTUS $\longrightarrow$ $\boxed{1} \to \boxed{2} \to \boxed{4} \to \boxed{11} \to \boxed{31} \to \boxed{45} \to \boxed{173} \to \boxed{174}$

CALPURNIA $\longrightarrow$ $\boxed{2} \to \boxed{31} \to \boxed{54} \to \boxed{101}$

Intersection $\Longrightarrow$ $\boxed{2}$

Recall basic intersection algorithm

BRUTUS $\longrightarrow$ $\boxed{1} \rightarrow \boxed{2} \rightarrow \boxed{4} \rightarrow \boxed{11} \rightarrow \boxed{31} \rightarrow \boxed{45} \rightarrow \boxed{173} \rightarrow \boxed{174}$

CALPURNIA $\longrightarrow$ $\boxed{2} \rightarrow \boxed{31} \rightarrow \boxed{54} \rightarrow \boxed{101}$

Intersection $\Longrightarrow$ $\boxed{2} \rightarrow \boxed{31}$

Recall basic intersection algorithm

BRUTUS	$\longrightarrow$	1 → 2 → 4 → 11 → 31 → 45 → 173 → 174
CALPURNIA	$\longrightarrow$	2 → 31 → 54 → 101
Intersection	$\Longrightarrow$	2 → 31

Recall basic intersection algorithm

BRUTUS $\longrightarrow$ $\boxed{1} \rightarrow \boxed{2} \rightarrow \boxed{4} \rightarrow \boxed{11} \rightarrow \boxed{31} \rightarrow \boxed{45} \rightarrow \boxed{173} \rightarrow \boxed{174}$

CALPURNIA $\longrightarrow$ $\boxed{2} \rightarrow \boxed{31} \rightarrow \boxed{54} \rightarrow \boxed{101}$

Intersection $\Longrightarrow$ $\boxed{2} \rightarrow \boxed{31}$

Recall basic intersection algorithm

BRUTUS	$\longrightarrow$	$\boxed{1} \rightarrow \boxed{2} \rightarrow \boxed{4} \rightarrow \boxed{11} \rightarrow \boxed{31} \rightarrow \boxed{45} \rightarrow \boxed{173} \rightarrow \boxed{174}$
CALPURNIA	$\longrightarrow$	$\boxed{2} \rightarrow \boxed{31} \rightarrow \boxed{54} \rightarrow \boxed{101}$
Intersection	$\Longrightarrow$	$\boxed{2} \rightarrow \boxed{31}$

Recall basic intersection algorithm

$\textsc{Brutus} \longrightarrow \boxed{1} \to \boxed{2} \to \boxed{4} \to \boxed{11} \to \boxed{31} \to \boxed{45} \to \boxed{173} \to \boxed{174}$

$\textsc{Calpurnia} \longrightarrow \boxed{2} \to \boxed{31} \to \boxed{54} \to \boxed{101}$

Intersection $\Longrightarrow \boxed{2} \to \boxed{31}$

Recall basic intersection algorithm

BRUTUS	$\longrightarrow$	$\boxed{1} \to \boxed{2} \to \boxed{4} \to \boxed{11} \to \boxed{31} \to \boxed{45} \to \boxed{173} \to \boxed{174}$
CALPURNIA	$\longrightarrow$	$\boxed{2} \to \boxed{31} \to \boxed{54} \to \boxed{101}$
Intersection	$\Longrightarrow$	$\boxed{2} \to \boxed{31}$

- Linear in the length of the postings lists.

Recall basic intersection algorithm

$$\textsc{Brutus} \longrightarrow \boxed{1} \to \boxed{2} \to \boxed{4} \to \boxed{11} \to \boxed{31} \to \boxed{45} \to \boxed{173} \to \boxed{174}$$

$$\textsc{Calpurnia} \longrightarrow \boxed{2} \to \boxed{31} \to \boxed{54} \to \boxed{101}$$

$$\text{Intersection} \Longrightarrow \boxed{2} \to \boxed{31}$$

- Linear in the length of the postings lists.
- Can we do better?

Skip pointers

- Skip pointers allow us to skip postings that will not figure in the search results.

Skip pointers

- Skip pointers allow us to skip postings that will not figure in the search results.
- This makes intersecting postings lists more efficient.

Skip pointers

- Skip pointers allow us to skip postings that will not figure in the search results.
- This makes intersecting postings lists more efficient.
- Some postings lists contain several million entries – so efficiency can be an issue even if basic intersection is linear.

Skip pointers

- Skip pointers allow us to skip postings that will not figure in the search results.
- This makes intersecting postings lists more efficient.
- Some postings lists contain several million entries – so efficiency can be an issue even if basic intersection is linear.
- Where do we put skip pointers?

Skip pointers

- Skip pointers allow us to skip postings that will not figure in the search results.
- This makes intersecting postings lists more efficient.
- Some postings lists contain several million entries – so efficiency can be an issue even if basic intersection is linear.
- Where do we put skip pointers?
- How do we make sure insection results are correct?

Basic idea

34 128

BRUTUS 2 4 8 34 35 64 128

8 31

CAESAR 1 2 3 5 8 17 21 31 75 81 84 89 92

Basic idea

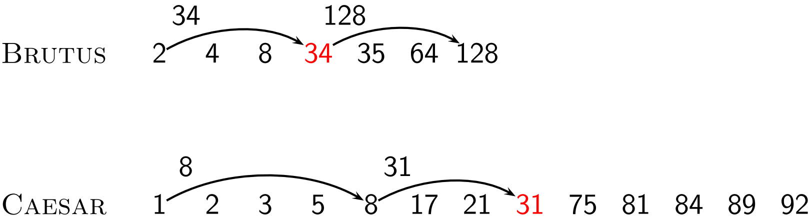
34 128

BRUTUS 2 4 8 34 35 64 128

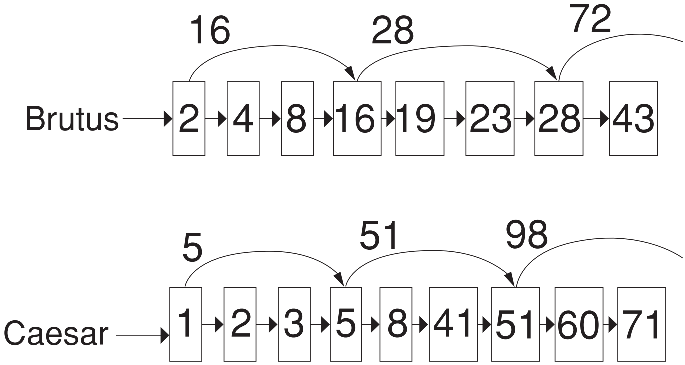
8 31

CAESAR 1 2 3 5 8 17 21 31 75 81 84 89 92

Basic idea


34
128
Brutus 2 4 8 34 35 64 128
8
31
Caesar 1 2 3 5 8 17 21 31 75 81 84 89 92

Skip lists: Larger example



Intersecting with skip pointers

```
IntersectWithSkips(p1, p2)
answer ← ⟨ ⟩
while p1 ≠ NIL and p2 ≠ NIL
do if docID(p1) = docID(p2)
      then Add(answer, docID(p1))
           p1 ← next(p1)
           p2 ← next(p2)
      else if docID(p1) < docID(p2)
              then if hasSkip(p1) and (docID(skip(p1)) ≤ docID(p2))
                      then while hasSkip(p1) and (docID(skip(p1)) ≤ docID(p2))
                              do p1 ← skip(p1)
                      else p1 ← next(p1)
              else if hasSkip(p2) and (docID(skip(p2)) ≤ docID(p1))
                      then while hasSkip(p2) and (docID(skip(p2)) ≤ docID(p1))
                              do p2 ← skip(p2)
                      else p2 ← next(p2)
return answer
```

Where do we place skips?

- Tradeoff: number of items skipped vs. frequency skip can be taken

Where do we place skips?

- Tradeoff: number of items skipped vs. frequency skip can be taken
- More skips: Each skip pointer skips only a few items, but we can frequently use it.

Where do we place skips?

- Tradeoff: number of items skipped vs. frequency skip can be taken
- More skips: Each skip pointer skips only a few items, but we can frequently use it.
- Fewer skips: Each skip pointer skips many items, but we can not use it very often.

Where do we place skips? (cont)

- Simple heuristic: for postings list of length P, use $\sqrt{P}$ evenly-spaced skip pointers.

Where do we place skips? (cont)

- Simple heuristic: for postings list of length P, use $\sqrt{P}$ evenly-spaced skip pointers.
- This ignores the distribution of query terms.

Where do we place skips? (cont)

- Simple heuristic: for postings list of length P, use $\sqrt{P}$ evenly-spaced skip pointers.
- This ignores the distribution of query terms.
- Easy if the index is static; harder in a dynamic environment because of updates.

Where do we place skips? (cont)

- Simple heuristic: for postings list of length P, use $\sqrt{P}$ evenly-spaced skip pointers.
- This ignores the distribution of query terms.
- Easy if the index is static; harder in a dynamic environment because of updates.
- How much do skip pointers help?

Where do we place skips? (cont)

- Simple heuristic: for postings list of length P, use $\sqrt{P}$ evenly-spaced skip pointers.
- This ignores the distribution of query terms.
- Easy if the index is static; harder in a dynamic environment because of updates.
- How much do skip pointers help?
- They used to help a lot.

Where do we place skips? (cont)

- Simple heuristic: for postings list of length P, use $\sqrt{P}$ evenly-spaced skip pointers.
- This ignores the distribution of query terms.
- Easy if the index is static; harder in a dynamic environment because of updates.
- How much do skip pointers help?
- They used to help a lot.
- With today's fast CPUs, they don't help that much anymore.

Outline

Phrase queries

- We want to answer a query such as [stanford university] – as a phrase.

Phrase queries

- We want to answer a query such as [stanford university] – as a phrase.
- Thus *The inventor Stanford Ovshinsky never went to university* should not be a match.

Phrase queries

- We want to answer a query such as [stanford university] – as a phrase.
- Thus *The inventor Stanford Ovshinsky never went to university* should not be a match.
- The concept of phrase query has proven easily understood by users.

Phrase queries

- We want to answer a query such as [stanford university] – as a phrase.
- Thus *The inventor Stanford Ovshinsky never went to university* should not be a match.
- The concept of phrase query has proven easily understood by users.
- Significant part of web queries are phrase queries (explicitly entered or interpreted as such)

Phrase queries

- We want to answer a query such as [stanford university] – as a phrase.
- Thus *The inventor Stanford Ovshinsky never went to university* should not be a match.
- The concept of phrase query has proven easily understood by users.
- Significant part of web queries are phrase queries (explicitly entered or interpreted as such)
- Consequence for inverted index: it no longer suffices to store docIDs in postings lists.

Phrase queries

- We want to answer a query such as [stanford university] – as a phrase.
- Thus *The inventor Stanford Ovshinsky never went to university* should not be a match.
- The concept of phrase query has proven easily understood by users.
- Significant part of web queries are phrase queries (explicitly entered or interpreted as such)
- Consequence for inverted index: it no longer suffices to store docIDs in postings lists.
- Two ways of extending the inverted index:

Phrase queries

- We want to answer a query such as [stanford university] – as a phrase.
- Thus *The inventor Stanford Ovshinsky never went to university* should not be a match.
- The concept of phrase query has proven easily understood by users.
- Significant part of web queries are phrase queries (explicitly entered or interpreted as such)
- Consequence for inverted index: it no longer suffices to store docIDs in postings lists.
- Two ways of extending the inverted index:
  - biword index

Phrase queries

- We want to answer a query such as [stanford university] – as a phrase.
- Thus *The inventor Stanford Ovshinsky never went to university* should not be a match.
- The concept of phrase query has proven easily understood by users.
- Significant part of web queries are phrase queries (explicitly entered or interpreted as such)
- Consequence for inverted index: it no longer suffices to store docIDs in postings lists.
- Two ways of extending the inverted index:
  - biword index
  - positional index

Biword indexes

- Index every consecutive pair of terms in the text as a phrase.

Biword indexes

- Index every consecutive pair of terms in the text as a phrase.
- For example, *Friends, Romans, Countrymen* would generate two biwords: *"friends romans"* and *"romans countrymen"*

Biword indexes

- Index every consecutive pair of terms in the text as a phrase.
- For example, *Friends, Romans, Countrymen* would generate two biwords: *"friends romans"* and *"romans countrymen"*
- Each of these biwords is now a vocabulary term.

Biword indexes

- Index every consecutive pair of terms in the text as a phrase.
- For example, *Friends, Romans, Countrymen* would generate two biwords: *"friends romans"* and *"romans countrymen"*
- Each of these biwords is now a vocabulary term.
- Two-word phrases can now easily be answered.

Longer phrase queries

- A long phrase like *"stanford university palo alto"* can be represented as the Boolean query "STANFORD UNIVERSITY" AND "UNIVERSITY PALO" AND "PALO ALTO"

Longer phrase queries

- A long phrase like *"stanford university palo alto"* can be represented as the Boolean query "STANFORD UNIVERSITY" AND "UNIVERSITY PALO" AND "PALO ALTO"
- We need to do post-filtering of hits to identify subset that actually contains the 4-word phrase.

Issues with biword indexes

- Why are biword indexes rarely used?

Issues with biword indexes

- Why are biword indexes rarely used?
- False positives, as noted above

Issues with biword indexes

- Why are biword indexes rarely used?
- False positives, as noted above
- Index blowup due to very large term vocabulary

Positional indexes

Positional indexes

- Positional indexes are a more efficient alternative to biword indexes.

Positional indexes

- Positional indexes are a more efficient alternative to biword indexes.
- Postings lists in a nonpositional index: each posting is just a docID

Positional indexes

- Positional indexes are a more efficient alternative to biword indexes.
- Postings lists in a nonpositional index: each posting is just a docID
- Postings lists in a positional index: each posting is a docID and a list of positions

Positional indexes: Example

Positional indexes: Example

Query: *"to_1 be_2 or_3 not_4 to_5 be_6"*

Positional indexes: Example

Query: "to_1 be_2 or_3 not_4 to_5 be_6"

TO, 993427:

⟨ 1: ⟨7, 18, 33, 72, 86, 231⟩;
2: ⟨1, 17, 74, 222, 255⟩;
4: ⟨8, 16, 190, 429, 433⟩;
5: ⟨363, 367⟩;
7: ⟨13, 23, 191⟩; . . . ⟩

BE, 178239:

⟨ 1: ⟨17, 25⟩;
4: ⟨17, 191, 291, 430, 434⟩;
5: ⟨14, 19, 101⟩; . . . ⟩

Positional indexes: Example

Query: "to_1 be_2 or_3 not_4 to_5 be_6"

TO, 993427:

⟨ 1: ⟨7, 18, 33, 72, 86, 231⟩;
2: ⟨1, 17, 74, 222, 255⟩;
4: ⟨8, 16, 190, 429, 433⟩;
5: ⟨363, 367⟩;
7: ⟨13, 23, 191⟩; . . . ⟩

BE, 178239:

⟨ 1: ⟨17, 25⟩;
4: ⟨17, 191, 291, 430, 434⟩;
5: ⟨14, 19, 101⟩; . . . ⟩

Positional indexes: Example

Query: “to_1 be_2 or_3 not_4 to_5 be_6”

TO, 993427:

⟨ 1: ⟨7, 18, 33, 72, 86, 231⟩;
2: ⟨1, 17, 74, 222, 255⟩;
4: ⟨8, 16, 190, 429, 433⟩;
5: ⟨363, 367⟩;
7: ⟨13, 23, 191⟩; . . . ⟩

BE, 178239:

⟨ 1: ⟨17, 25⟩;
4: ⟨17, 191, 291, 430, 434⟩;
5: ⟨14, 19, 101⟩; . . . ⟩

Positional indexes: Example

Query: "to_1 be_2 or_3 not_4 to_5 be_6"

TO, 993427:

$\langle$ 1: $\langle$7, 18, 33, 72, 86, 231$\rangle$;
2: $\langle$1, 17, 74, 222, 255$\rangle$;
4: $\langle$8, 16, 190, 429, 433$\rangle$;
5: $\langle$363, 367$\rangle$;
7: $\langle$13, 23, 191$\rangle$; . . . $\rangle$

BE, 178239:

$\langle$ 1: $\langle$17, 25$\rangle$;
4: $\langle$17, 191, 291, 430, 434$\rangle$;
5: $\langle$14, 19, 101$\rangle$; . . . $\rangle$

Positional indexes: Example

Query: “to_1 be_2 or_3 not_4 to_5 be_6”

TO, 993427:
$\langle$ 1: $\langle$7, 18, 33, 72, 86, 231$\rangle$;
2: $\langle$1, 17, 74, 222, 255$\rangle$;
4: $\langle$8, 16, 190, 429, 433$\rangle$;
5: $\langle$363, 367$\rangle$;
7: $\langle$13, 23, 191$\rangle$; . . . $\rangle$

BE, 178239:
$\langle$ 1: $\langle$17, 25$\rangle$;
4: $\langle$17, 191, 291, 430, 434$\rangle$;
5: $\langle$14, 19, 101$\rangle$; . . . $\rangle$

Positional indexes: Example

Query: "to_1 be_2 or_3 not_4 to_5 be_6"

TO, 993427:

⟨ 1: ⟨7, 18, 33, 72, 86, 231⟩;
2: ⟨1, 17, 74, 222, 255⟩;
4: ⟨8, 16, 190, 429, 433⟩;
5: ⟨363, 367⟩;
7: ⟨13, 23, 191⟩; . . . ⟩

BE, 178239:

⟨ 1: ⟨17, 25⟩;
4: ⟨17, 191, 291, 430, 434⟩;
5: ⟨14, 19, 101⟩; . . . ⟩

Positional indexes: Example

Query: *"to_1 be_2 or_3 not_4 to_5 be_6"*

TO, 993427:

$\langle$ 1: $\langle$7, 18, 33, 72, 86, 231$\rangle$;
2: $\langle$1, 17, 74, 222, 255$\rangle$;
4: $\langle$8, 16, 190, 429, 433$\rangle$;
5: $\langle$363, 367$\rangle$;
7: $\langle$13, 23, 191$\rangle$; . . . $\rangle$

BE, 178239:

$\langle$ 1: $\langle$17, 25$\rangle$;
4: $\langle$17, 191, 291, 430, 434$\rangle$;
5: $\langle$14, 19, 101$\rangle$; . . . $\rangle$

Positional indexes: Example

Query: "to_1 be_2 or_3 not_4 to_5 be_6"

TO, 993427:

⟨ 1: ⟨7, 18, 33, 72, 86, 231⟩;
2: ⟨1, 17, 74, 222, 255⟩;
4: ⟨8, 16, 190, 429, 433⟩;
5: ⟨363, 367⟩;
7: ⟨13, 23, 191⟩; . . . ⟩

BE, 178239:

⟨ 1: ⟨17, 25⟩;
4: ⟨17, 191, 291, 430, 434⟩;
5: ⟨14, 19, 101⟩; . . . ⟩

Positional indexes: Example

Query: *"to_1 be_2 or_3 not_4 to_5 be_6"*

TO, 993427:

⟨ 1: ⟨7, 18, 33, 72, 86, 231⟩;
  2: ⟨1, 17, 74, 222, 255⟩;
  4: ⟨8, 16, 190, 429, 433⟩;
  5: ⟨363, 367⟩;
  7: ⟨13, 23, 191⟩; . . . ⟩

BE, 178239:

⟨ 1: ⟨17, 25⟩;
  4: ⟨17, 191, 291, 430, 434⟩;
  5: ⟨14, 19, 101⟩; . . . ⟩

Positional indexes: Example

Query: “to_1 be_2 or_3 not_4 to_5 be_6”

TO, 993427:

⟨ 1: ⟨7, 18, 33, 72, 86, 231⟩;
2: ⟨1, 17, 74, 222, 255⟩;
4: ⟨8, 16, 190, 429, 433⟩;
5: ⟨363, 367⟩;
7: ⟨13, 23, 191⟩; . . . ⟩

BE, 178239:

⟨ 1: ⟨17, 25⟩;
4: ⟨17, 191, 291, 430, 434⟩;
5: ⟨14, 19, 101⟩; . . . ⟩

Positional indexes: Example

Query: "to_1 be_2 or_3 not_4 to_5 be_6"

TO, 993427:
⟨ 1: ⟨7, 18, 33, 72, 86, 231⟩;
2: ⟨1, 17, 74, 222, 255⟩;
4: ⟨8, 16, 190, 429, 433⟩;
5: ⟨363, 367⟩;
7: ⟨13, 23, 191⟩; . . . ⟩

BE, 178239:
⟨ 1: ⟨17, 25⟩;
4: ⟨17, 191, 291, 430, 434⟩;
5: ⟨14, 19, 101⟩; . . . ⟩

Positional indexes: Example

Query: "to_1 be_2 or_3 not_4 to_5 be_6"

TO, 993427:

⟨ 1: ⟨7, 18, 33, 72, 86, 231⟩;
2: ⟨1, 17, 74, 222, 255⟩;
4: ⟨8, 16, 190, 429, 433⟩;
5: ⟨363, 367⟩;
7: ⟨13, 23, 191⟩; . . . ⟩

BE, 178239:

⟨ 1: ⟨17, 25⟩;
4: ⟨17, 191, 291, 430, 434⟩;
5: ⟨14, 19, 101⟩; . . . ⟩

Positional indexes: Example

Query: “to_1 be_2 or_3 not_4 to_5 be_6”

TO, 993427:

⟨ 1: ⟨7, 18, 33, 72, 86, 231⟩;
2: ⟨1, 17, 74, 222, 255⟩;
4: ⟨8, 16, 190, 429, 433⟩;
5: ⟨363, 367⟩;
7: ⟨13, 23, 191⟩; . . . ⟩

BE, 178239:

⟨ 1: ⟨17, 25⟩;
4: ⟨17, 191, 291, 430, 434⟩;
5: ⟨14, 19, 101⟩; . . . ⟩

Positional indexes: Example

Query: "to_1 be_2 or_3 not_4 to_5 be_6"

TO, 993427:

⟨ 1: ⟨7, 18, 33, 72, 86, 231⟩;
2: ⟨1, 17, 74, 222, 255⟩;
4: ⟨8, 16, 190, 429, 433⟩;
5: ⟨363, 367⟩;
7: ⟨13, 23, 191⟩; . . . ⟩

BE, 178239:

⟨ 1: ⟨17, 25⟩;
4: ⟨17, 191, 291, 430, 434⟩;
5: ⟨14, 19, 101⟩; . . . ⟩

Positional indexes: Example

Query: "to_1 be_2 or_3 not_4 to_5 be_6"

TO, 993427:
⟨ 1: ⟨7, 18, 33, 72, 86, 231⟩;
2: ⟨1, 17, 74, 222, 255⟩;
4: ⟨8, 16, 190, 429, 433⟩;
5: ⟨363, 367⟩;
7: ⟨13, 23, 191⟩; . . . ⟩

BE, 178239:
⟨ 1: ⟨17, 25⟩;
4: ⟨17, 191, 291, 430, 434⟩;
5: ⟨14, 19, 101⟩; . . . ⟩

Positional indexes: Example

Query: *"to_1 be_2 or_3 not_4 to_5 be_6"*

TO, 993427:

$\langle$ 1: $\langle$7, 18, 33, 72, 86, 231$\rangle$;
2: $\langle$1, 17, 74, 222, 255$\rangle$;
4: $\langle$8, 16, 190, 429, 433$\rangle$;
5: $\langle$363, 367$\rangle$;
7: $\langle$13, 23, 191$\rangle$; . . . $\rangle$

BE, 178239:

$\langle$ 1: $\langle$17, 25$\rangle$;
4: $\langle$17, 191, 291, 430, 434$\rangle$;
5: $\langle$14, 19, 101$\rangle$; . . . $\rangle$

Positional indexes: Example

Query: "to_1 be_2 or_3 not_4 to_5 be_6"

TO, 993427:

$\langle$ 1: $\langle$7, 18, 33, 72, 86, 231$\rangle$;
2: $\langle$1, 17, 74, 222, 255$\rangle$;
4: $\langle$8, 16, 190, 429, 433$\rangle$;
5: $\langle$363, 367$\rangle$;
7: $\langle$13, 23, 191$\rangle$; . . . $\rangle$

BE, 178239:

$\langle$ 1: $\langle$17, 25$\rangle$;
4: $\langle$17, 191, 291, 430, 434$\rangle$;
5: $\langle$14, 19, 101$\rangle$; . . . $\rangle$

Document 4 is a match!

Proximity search

- We just saw how to use a positional index for phrase searches.

Proximity search

- We just saw how to use a positional index for phrase searches.
- We can also use it for proximity search.

Proximity search

- We just saw how to use a positional index for phrase searches.
- We can also use it for proximity search.
- For example: employment /4 place

Proximity search

- We just saw how to use a positional index for phrase searches.
- We can also use it for proximity search.
- For example: employment /4 place
- Find all documents that contain EMPLOYMENT and PLACE within 4 words of each other.

Proximity search

- We just saw how to use a positional index for phrase searches.
- We can also use it for proximity search.
- For example: employment /4 place
- Find all documents that contain EMPLOYMENT and PLACE within 4 words of each other.
- *Employment agencies that place healthcare workers are seeing growth* is a hit.

Proximity search

- We just saw how to use a positional index for phrase searches.
- We can also use it for proximity search.
- For example: employment /4 place
- Find all documents that contain EMPLOYMENT and PLACE within 4 words of each other.
- *Employment agencies that place healthcare workers are seeing growth* is a hit.
- *Employment agencies that have learned to adapt now place healthcare workers* is not a hit.

Proximity search

- Use the positional index

Proximity search

- Use the positional index
- Simplest algorithm: look at cross-product of positions of (i) EMPLOYMENT in document and (ii) PLACE in document

Proximity search

- Use the positional index
- Simplest algorithm: look at cross-product of positions of (i) EMPLOYMENT in document and (ii) PLACE in document
- Very inefficient for frequent words, especially stop words

Proximity search

- Use the positional index
- Simplest algorithm: look at cross-product of positions of (i) EMPLOYMENT in document and (ii) PLACE in document
- Very inefficient for frequent words, especially stop words
- Note that we want to return the actual matching positions, not just a list of documents.

Proximity search

- Use the positional index
- Simplest algorithm: look at cross-product of positions of (i) EMPLOYMENT in document and (ii) PLACE in document
- Very inefficient for frequent words, especially stop words
- Note that we want to return the actual matching positions, not just a list of documents.
- This is important for dynamic summaries etc.

"Proximity" intersection

```
PositionalIntersect(p1, p2, k)
answer ← ⟨ ⟩
while p1 ≠ NIL and p2 ≠ NIL
do if docID(p1) = docID(p2)
      then l ← ⟨ ⟩
           pp1 ← positions(p1)
           pp2 ← positions(p2)
           while pp1 ≠ NIL
           do while pp2 ≠ NIL
              do if |pos(pp1) − pos(pp2)| ≤ k
                    then Add(l, pos(pp2))
                    else if pos(pp2) > pos(pp1)
                            then break
                 pp2 ← next(pp2)
              while l ≠ ⟨ ⟩ and |l[0] − pos(pp1)| > k
              do Delete(l[0])
              for each ps ∈ l
              do Add(answer, ⟨docID(p1), pos(pp1), ps⟩)
              pp1 ← next(pp1)
           p1 ← next(p1)
           p2 ← next(p2)
      else if docID(p1) < docID(p2)
              then p1 ← next(p1)
              else p2 ← next(p2)
return answer
```

Combination scheme

- Biword indexes and positional indexes can be profitably combined.

Combination scheme

- Biword indexes and positional indexes can be profitably combined.
- Many biwords are extremely frequent: Michael Jackson, Britney Spears etc

Combination scheme

- Biword indexes and positional indexes can be profitably combined.
- Many biwords are extremely frequent: Michael Jackson, Britney Spears etc
- For these biwords, increased speed compared to positional postings intersection is substantial.

Combination scheme

- Biword indexes and positional indexes can be profitably combined.
- Many biwords are extremely frequent: Michael Jackson, Britney Spears etc
- For these biwords, increased speed compared to positional postings intersection is substantial.
- Combination scheme: Include frequent biwords as vocabulary terms in the index. Do all other phrases by positional intersection.

Combination scheme

- Biword indexes and positional indexes can be profitably combined.
- Many biwords are extremely frequent: Michael Jackson, Britney Spears etc
- For these biwords, increased speed compared to positional postings intersection is substantial.
- Combination scheme: Include frequent biwords as vocabulary terms in the index. Do all other phrases by positional intersection.
- Williams et al. (2004) evaluate a more sophisticated mixed indexing scheme. Faster than a positional index, at a cost of 26% more space for index.

"Positional" queries on Google

- For web search engines, positional queries are much more expensive than regular Boolean queries.

“Positional” queries on Google

- For web search engines, positional queries are much more expensive than regular Boolean queries.
- Let’s look at the example of phrase queries.

"Positional" queries on Google

- For web search engines, positional queries are much more expensive than regular Boolean queries.
- Let's look at the example of phrase queries.
- Why are they more expensive than regular Boolean queries?

"Positional" queries on Google

- For web search engines, positional queries are much more expensive than regular Boolean queries.
- Let's look at the example of phrase queries.
- Why are they more expensive than regular Boolean queries?
- Can you demonstrate on Google that phrase queries are more expensive than Boolean queries?

Take-away

- Understanding of the basic unit of classical information retrieval systems: words and documents: What is a document, what is a term?
- Tokenization: how to get from raw text to words (or tokens)
- More complex indexes: skip pointers and phrases

Resources

- Chapter 2 of IIR
- Resources at `http://cislmu.org`
  - Porter stemmer
  - A fun number search on Google