# Information Retrieval

Weike Pan

The slides are **adapted from those provided by Prof. Hinrich Schütze** at University of Munich (http://www.cis.lmu.de/~hs/teach/14s/ir/).

# Chapter 2 The term vocabulary & postings lists

- 2.1 Document delineation and character sequence decoding
- 2.2 Determining the vocabulary of terms
- 2.3 Faster postings list intersection via skip pointers
- 2.4 Positional postings and phrase queries
- 2.5 References and further reading

# Outline

- 2.1 Document delineation (文档分析) and character sequence decoding
- 2.2 Determining the vocabulary of terms
- 2.3 Faster postings list intersection via skip pointers
- 2.4 Positional postings and phrase queries
- 2.5 References and further reading

# 2.1 Document delineation and character sequence decoding

**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.

# 2.1 Document delineation and character sequence decoding

**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 (see Chapter 13).
- Alternative: use heuristics

# 2.1 Document delineation and character sequence decoding

**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

# 2.1 Document delineation and character sequence decoding

- What is the document unit for indexing?
  - A file?
  - An email?
  - An email with 5 attachments?
  - A group of files (PPT or Latex, HTML)?

- Answering the question "what is a document?" is not trivial and requires some design decisions.

# Outline

- 2.1 Document delineation and character sequence decoding
- 2.2 Determining the vocabulary of terms
- 2.3 Faster postings list intersection via skip pointers
- 2.4 Positional postings and phrase queries
- 2.5 References and further reading

# 2.2 Determining the vocabulary of terms

**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.

# 2.2 Determining the vocabulary of terms

**Normalization (1/3)**

- 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

# 2.2 Determining the vocabulary of terms

**Normalization (2/3)**

- Alternatively: do asymmetric (非对称的) expansion
  - window -> window, windows
  - windows -> Windows, windows
  - Windows (no expansion)

- More powerful, but less efficient

- Question: Why don't you want to put window, Window, windows, and Windows in the same equivalence class?

# 2.2 Determining the vocabulary of terms

**Normalization (3/3)**

- Normalization and language detection interact.
  - PETER WILL NICHT MIT. -> MIT = mit
  - He got his PhD from MIT. -> MIT is for Massachusetts Institute of Technology (麻省理工学院)

# 2.2 Determining the vocabulary of terms

**Tokenization**

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

# 2.2 Determining the vocabulary of terms

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

- Hewlett-Packard (惠普)
- State-of-the-art
- co-education
- data base
- San Francisco
- Los Angeles-based company
- cheap San Francisco-Los Angeles fares
- York University vs. New York University

# 2.2 Determining the vocabulary of terms

**Tokenization: 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

# 2.2 Determining the vocabulary of terms

**Tokenization: No whitespace in Chinese**

- 《信息检索》是一门本科生的专业选修课。

# 2.2 Determining the vocabulary of terms

**Tokenization: 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’.

# 2.2 Determining the vocabulary of terms

**Tokenization: Other cases of "no whitespace"**

- Compounds in Dutch (荷兰语), German (德语), Swedish (瑞典语)
  - Computerlinguistik -> Computer + Linguistik
  - Lebensversicherungsgesellschaftsangestellter -> leben + versicherung + gesellschaft + angestellter

- Inuit (因纽特语): tusaatsiarunnanngittualuujunga (I can't hear very well.)

- Many other languages with segmentation difficulties: Finnish (芬兰语), Urdu (乌尔都语), ...

# 2.2 Determining the vocabulary of terms

**Tokenization: Japanese**

- 4 different “alphabets”
  - Chinese characters
  - Hiragana (平假名) syllabary for inflectional endings and function words
  - Katakana (片假名) syllabary for transcription of foreign words and other uses
  - Latin
- No spaces (as in Chinese).

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。

# 2.2 Determining the vocabulary of terms

**Tokenization: Bidirectionality in Arabic script**

- ‘Algeria achieved its independence in 1962 after 132 years of French occupation.’

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

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

# 2.2 Determining the vocabulary of terms

**Tokenization: Accents (重音) and diacritics (变音符号)**

- Accents: resume (simple omission of accent)
- Umlauts (变音): 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, e.g., Polish (波兰语).
  - Question: Why?

# 2.2 Determining the vocabulary of terms

**English: Case folding**

- Reduce all letters to lower case
- Even though case can be semantically meaningful
  - E.g., 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.

# 2.2 Determining the vocabulary of terms

**English: 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“ (丹麦)

- Most web search engines index stop words.

# 2.2 Determining the vocabulary of terms

**English: More equivalence classing**

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

# 2.2 Determining the vocabulary of terms

**English: Lemmatization (词形归并)**

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

- Lemmatization implies doing "proper" reduction to dictionary headword form (the lemma).

- Inflectional morphology (曲折形态学): cutting -> cut
- Derivational morphology (衍生形态学): destruction -> destroy; automate, automatic, automation -> automat

# 2.2 Determining the vocabulary of terms

**English: Stemming (词干还原)**

- Definition of stemming: Crude heuristic process that chops off the ends of words in the hope of achieving what "principled"

- Lemmatization (词形归并) attempts to do with a lot of linguistic knowledge. It is language dependent.

# 2.2 Determining the vocabulary of terms

**English: Porter algorithm (1/2)**

- Most common algorithm for stemming English (http://tartarus.org/~martin/PorterStemmer/)

- Results suggest that it is at least as good as other stemming options
  - Conventions (约定) + 5 phases of reductions (约简)
    - Phases are applied sequentially
    - Each phase consists of a set of commands

- Convention (约定): Of the rules in a compound command, select the one that applies to the longest suffix.

# 2.2 Determining the vocabulary of terms

**English: Porter algorithm (2/2)**

- **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

# 2.2 Determining the vocabulary of terms

**English: 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] (格子呢毛衣), [sightseeing tour san francisco] (旧金山观光旅游)

- Queries where stemming hurts: [operational AND research] (运筹学), [operating AND system] (操作系统), [operative AND dentistry] (牙科手术)

# 2.2 Determining the vocabulary of terms

**English: What does Google do?**

- Stop words
- Normalization
- Tokenization
- Lowercasing
- Stemming (词干还原)
- Non-latin alphabets
- Umlauts (变音)
- Compounds (复合词)
- Numbers

# Outline

- 2.1 Document delineation and character sequence decoding
- 2.2 Determining the vocabulary of terms
- 2.3 Faster postings list intersection via skip pointers
- 2.4 Positional postings and phrase queries
- 2.5 References and further reading

# 2.3 Faster postings list intersection via skip pointers

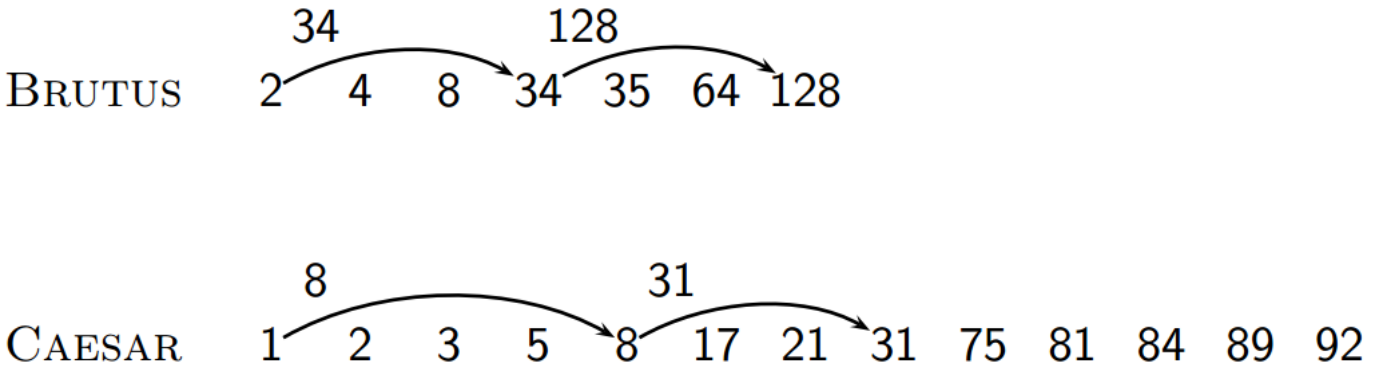
**Skip points**

- 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 intersection results are correct?

# 2.3 Faster postings list intersection via skip pointers

**Basic idea**



# 2.3 Faster postings list intersection via skip pointers

**Intersecting with skip pointers**

```
INTERSECTWITHSKIPS(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 if hasSkip(p1) and (docID(skip(p1)) ≤ docID(p2))
 9                     then while hasSkip(p1) and (docID(skip(p1)) ≤ docID(p2))
10                          do p1 ← skip(p1)
11                     else p1 ← next(p1)
12                else if hasSkip(p2) and (docID(skip(p2)) ≤ docID(p1))
13                     then while hasSkip(p2) and (docID(skip(p2)) ≤ docID(p1))
14                          do p2 ← skip(p2)
15                     else p2 ← next(p2)
16  return answer
```

# 2.3 Faster postings list intersection via skip pointers

**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.

- **Simple heuristic**: for postings list of length P, use P^0.5 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.

# 2.3 Faster postings list intersection via skip pointers

- 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

- 2.1 Document delineation and character sequence decoding
- 2.2 Determining the vocabulary of terms
- 2.3 Faster postings list intersection via skip pointers
- 2.4 Positional postings and phrase queries
- 2.5 References and further reading

# 2.4 Positional postings and phrase queries

**Phrase queries (1/2)**

- 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).

# 2.4 Positional postings and phrase queries

**Phrase queries (2/2)**

- 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

# 2.4 Positional postings and phrase queries

**Biword indexes (1/3)**

- 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

# 2.4 Positional postings and phrase queries

**Biword indexes (2/3)**

- 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.

# 2.4 Positional postings and phrase queries

**Biword indexes (3/3)**

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

# 2.4 Positional postings and phrase queries

**Positional indexes (1/2)**

- 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

# 2.4 Positional postings and phrase queries

**Positional indexes (2/2)**

- Query: “to be or not to be”
- Document 4 is a match!

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⟩; . . . ⟩

# 2.4 Positional postings and phrase queries

**Proximity search (1/2)**

- 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.

# 2.4 Positional postings and phrase queries

**Proximity search (2/2)**

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

# 2.4 Positional postings and phrase queries

**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.

# 2.4 Positional postings and phrase queries

**"Positional" queries on Google**

- For web search engines, positional queries are much more expensive than regular Boolean queries.
  - Why are they more expensive than regular Boolean queries?
  - Can you demonstrate on Google that phrase queries are more expensive than Boolean queries?

# Summary

- 2.1 Document delineation and character sequence decoding
- 2.2 Determining the vocabulary of terms
- 2.3 Faster postings list intersection via skip pointers
- 2.4 Positional postings and phrase queries
- 2.5 References and further reading