مبانی بازیابی اطلاعات و جستجوی وب

۳- The term vocabulary and postings lists

Original slides by: Hinrich Schütze and Christina Lioma

Overview

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

Outline

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

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?
- هرکدام از این موارد یک مسئله رده بندی است که بعدا به آن می پردازیم (فصل ۱۳).
- راه جایگزین: استفاده از روش های اکتشافی

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)?
- بنابراین پاسخ سوال سند چیست، بدیهی نبوده و نیازمند تصمیمات طراحی است.
- Also: XML

Outline

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

Definitions

- Word – A delimited string of characters as it appears in the text.
- Term – A “normalized” word (case, morphology, spelling etc); یک کلاس هم ارز از کلمات.
- 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.
  - In June, the dog likes to chase the cat in the barn.
  - 12 word tokens, 9 word types

Recall: Inverted index construction

- Input:

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

. . .

- Output:

friend	roman	countryman	so

. . .

- هر توکن گزینه ای برای درج در دیکشنری است.
- کدام توکن ها مناسب این کار هستند؟

Problems in tokenization

- What are the delimiters? Space? Apostrophe? Hyphen?
  - هرکدام از اینها گاهی جداکننده هستند و گاهی خیر
- No whitespace in many languages! (e.g., Chinese)
- No whitespace in Dutch, German, Swedish compounds (*Lebensversicherungsgesellschaftsangestellter*)

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
- 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.

Outline

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

Stop words

- stop words = کلمات بسیار پرتکرار که ارزش ناچیزی در کمک به انتخاب اسناد مرتبط با نیاز کاربر دارند.
- 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*
- حذف کلمات ایست در سیستم های بازیابی قدیمی تر یک رویه استاندارد بود.
- But you need stop words for phrase queries, e.g. “King of Denmark”
- Most web search engines index stop words.

Normalization

- Need to "normalize" terms 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: Case folding

- Reduce all letters to lower case
- Possible exceptions: 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.

Lemmatization

- 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 (*cutting* → *cut)* vs. derivational morphology (*destruction* → *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.
- Language dependent
- Often inflectional and derivational
- Example for derivational: *automate, automatic, automation* all reduce to *automat*

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.

Optional: 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 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.
- Porter Stemmer equivalence class *oper* contains all of *operate operating operates operation operative operatives operational.*

Exercise: What does Google do?

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

Outline

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

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

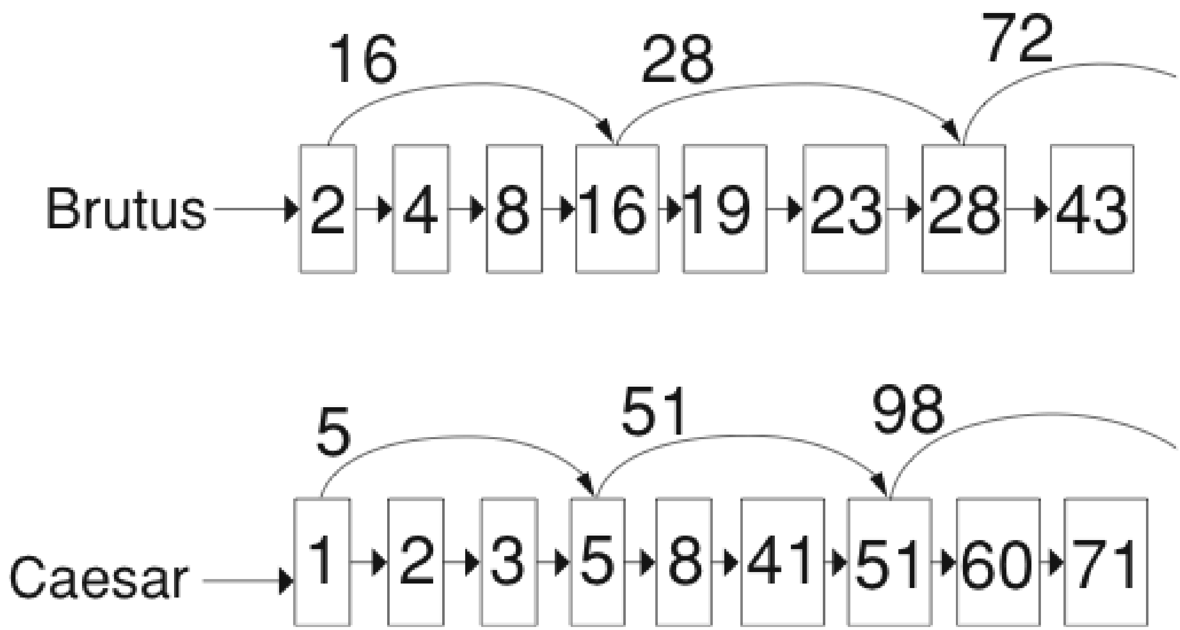
Intersection $\Longrightarrow$ $\boxed{2} \rightarrow \boxed{31}$

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

Skip pointers

- اشاره گرهای پرش امکان رد شدن از اسنادی (postings) که در نتیجه جستجو وجود ندارد را فراهم می کند.
- این امر موجب کاراتر شدن فرایند اشتراک گیری خواهد شد.
- گاها لیست اسناد (postings) شامل چندین میلیون مدخل است، بنابراین کارایی حتی به صورت خطی یک مسئله است.
- اشاره گرهای پرش را کجا قرار دهیم؟
- چگونه مطمئن شویم که نتیجه اشتراک گیری صحیح است؟

Skip lists



Optional: Intersection 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: تعداد ایتم های پرش شده در مقایسه با فرکانس پرش ها
- 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.
- 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

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

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.
- About 10% of web queries are phrase queries.
- 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.
- 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"
- We need to do post-filtering of hits to identify subset that actually contains the 4-word phrase.

Optional: Extended biwords

- Parse each document and perform part-of-speech tagging
- Bucket the terms into (say) nouns (N) and articles/prepositions (X)
- Now deem any string of terms of the form NX*N to be an *extended biword*
- Examples: catcher in the rye

  N X X N

  king of Denmark

  N X N
- Include extended biwords in the term vocabulary
- Queries are processed accordingly

Positional indexes

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

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›; . . . ›

Document 4 is a match!

Optional: “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.
- Many biwords are extremely frequent: Shahrood University, information retrieval, 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.

Optional: “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?

منابع

- فصل دوم کتاب An introduction to information retrieval