Document Processing

Facts:

• Documents may belong to various languages. Web: ~ 60% in English

• A given document may have foreign language terms and phrases.

• Skewed term frequency distribution
Outline

– Tokenizing single terms
– Stop terms
– Special terms
– Normalization of tokens
– Phrasing
– Stemming
– n-grams
– links

Parsing Single Terms

• Splitting on white spaces
  – “parsing single terms”
    “Parsing”, “single”, “terms”

  Problem:
  – “whitespaces” or “white spaces”
  – month day, year “Aug 28, 2008”
  – “Washington DC”

• Each language has somewhat its own conventions as to word boundaries.
  – Some languages use a compound splitter or segmentation software.
**Stop Words**

- Terms that occur too many times in a collection and hence are not discriminating:
  - to, a, the, of, from, ….
  - Evaluate the stop terms for a domain
  - Stop word lists are maintained
    - Reduces the index size
    - Problem: some searches are not successful: “to be or not to be”
    - It is a lossy compression.
  - General trend in IR has been to reduce the size of stop word list or eliminate their use.
    - Using a good index compression
    - Weighting stop terms accordingly for query processing (query-based)

**Special Tokens**

- Dates 2005; Oct 10, 2005; 10/10/2005; 10/10/05
- Digit-alphabet 1-hour
- Alphabet-digit F-16; I-20
- Hyphenation co-existence; black-tie party
- All caps CNN, BBC
- Cap period (initial) N.
- Digit, digit 8.00
- Digit, digit 8,000
- Currency symbol $, ….
- Cultural known names M*A*S*H
- Email address mouse@hotmail.com
- URLs http://www.cnn.com
- IP address 123.67.65.870
- Names New York; Los Angeles (Los Angeles-New York flights ???)
Normalization of Tokens

• Using equivalence class of terms. Example rules:
  – Ph.D → Phd
  – U.S.A. → USA
  – 10/10/2005 → Oct 10, 2005
  – F-16 → F16
  – Variations of Umlaut words in German
  – ..................

• What about these rules?
  – Windows → window                  (what if one is OS and one is a window???)
  – C.A.D. → cad                        (different meaning???)

Normalization of Tokens (cont’d)

• Case folding - reduces term index by ~17%, but a lossy compression
  – Convert all to lower case (most practical); or some to lower case

• Spelling variations (neighbor vs. neighbour; a foreign name)

• Accents on letters (naïve vs. naive; many foreign language terms)

• Variant transliteration (Den-Haag vs. The Hague)
  – Use Soundex algorithm!

More on normalization under Stemming....
Phrase processing

- Phrase recognition is based on the goal of indexing meaningful phrases like
  - “Lincoln Town Car”
  - “San Francisco”
  - “apple pie”
- Doing this would use word order to assist with effectiveness -- otherwise we are assuming the query and documents are just a “bag of words”
- ~ 10% of web queries are explicit phrase queries

Phrase processing

- Add phrase terms to the query just like other terms
  - This really violates independence assumptions but a lot of people do it anyway
- Give phrase terms a different weight than query terms
Constructing Phrases
using n-gram words

• Using bigrams, trigrams
• Start with all 2-word pairs that are not separated by punctuation, stop words, or special characters
• Only store those that occur more than $x$ times
  – Example: New York; Apple Pie;…

Constructing Phrases
using term positions

• Store the term positions
• Identify phrases at the query processing time
• Good flexibility for various window sizes
• May be too slow for large collections
Constructing Phrases
using Part-of-Speech Tagging

Can take advantage of NLP techniques:
• Using part-of-Speech tagging to identify key components of a sentence (S-V-OBJ, …)
  – store all noun phrases “Republic of China”, or
  – store adjective followed by noun “Red Carpet”
• Problem: too slow!

Constructing Phrases
Using Named Entity Tagging

• Finding structured data within an unstructured document
  – People’s names, organizations, locations, amounts, etc.
Phrase Processing Summary

• Pro
  – Often found to improve effectiveness by 10%

• Con
  – Dramatically increases size of term dictionary and the size of the index

Parser Generators

• Goal is to allow users to specify parsing rules as grammars.
• Grammars provide a very flexible means of expressing all valid strings in a language.
Some useful regular expressions

Acronym: (["A"-"Z"])(["A"-"Z"])* Ex: NCR, IBM, etc.

Abbreviation: (["A"-"Z"] "")* Ex: U.S.A.

Model: ["a"-"z","A"-"Z"] ",-" (["0"-"9"])* Ex: F-16, C-25

Word: ["a"-"z","A"-"Z"] (["a"-"z","A"-"Z" ])* Ex: hippo, Hippo

Integer: ["0"-"9"] (["0"-"9"])* Ex: 123

Decimal: (["0"-"9"])*. (["0"-"9"]) Ex: 123.45

Stemming

- Goal of stemming (Conflation) is to reduce variations of each word due to inflection or derivation to a common stem.
- Improves effectiveness by providing a better match between query and a relevant document.
- User who is searching for “swimming” might be interested in documents with “swim”.
- Reduces the term index by ~17%
- It is a lossy compression.
Stemming

• @ indexing time
  – Storing only the stems
    • Reduces the flexibility for certain context, improves for some other
    • Reduces index size
  – Storing both stems and non-stemmed terms

• @ Query processing time
  • Increases the flexibility of not stemming the Q terms
  • Must expand the Q to all term variations (slow)

Stemming Algorithms

• Stemming algorithms generate stem classes.

  – Rule-Based
    • Porter (1980)
    • Lovins (1968)
  – Dictionary-based
    • K-stem (1989, 1993)
  – Corpus/Co-Occurrence-Based (1994)
Porter Stemmer

- An incoming word is cleaned up in the initialization phase, one prefix trimming phase then takes place and then five suffix trimming phases occur.
- Note: The entire algorithm will not be covered -- we will leave out some obscure rules.

Initialization

- First the word is cleaned up. Converted to lower case only letters or digits are kept.
- F-16 is converted to f16.
Porter Stemming

• Remove prefixes:
"kilo", "micro", "milli", "intra", "ultra", "mega", "nano", "pico", "pseudo"

So megabyte, kilobyte all become “byte”.

Porter Step 1

• Replace “ing” with “e”, if number of consonant-vowels switches, called measure, is greater then 3.
  – liberating --> liberate, facilitating--> facilitate
• Remove “es” from words that end in “sses” or “ies”
  – passes --> pass,cries --> cri
• Remove “s” from words whose next to last letter is not an “s”
  – runs --> run, fuss --> fuss
• If word has a vowel and ends with “eed” remove the “ed”
  – agreed --> AGRE, freed --> freed
• Remove “ed” and “ing” from words that have other vowel
  – dreaded --> dread, red --> red, bother --> bother, bring --> bring
• Remove “d” if word has a vowel and ends with “ated” or “bled”
  – enabled --> enable, generated --> generate
• Replace trailing “y” with an “I” if word has a vowel
  – satisfy --> satisfy, fly --> fly
Porter Step 2

• With what is left, replace any suffix on the left with suffix on the right - only if the consonant-vowels measure >0

... tional  tion  conditional --> condition
ization  ize  nationalization --> nationalize
iveness  ive  effectiveness --> effective
fulness  ful  usefulness --> useful
ousness  ous  nervousness --> nervous
ousli  ous  nervously --> nervous
entli  ent  fervently --> fervent
iveness  ive  inventiveness --> inventive
bility  ble  sensibility --> sensible

Step 3

• With what is left, replace any suffix on the left with suffix on the right

... icate  ic  fabricate --> fabric (*Think about this one*)
ative  --  combativ --> comb (*another good one*)
alize  al  nationalize --> national
icit  ic  sensibility --> sensible
ical  ic  tropical --> tropic
ful  --  faithful --> faith
iveness  ive  inventiveness --> inventive
ness  --  harness --> har
Step 4

• Remove remaining standard suffixes
  al, ance, ence, er, ic, able, ible, ant, ement,
  ment, ent, sion, tion, ou, ism, ate, iti, ous, ive,
  ize, ise

Step 5

• Remove trailing “e” if word does not end in a vowel
  – hinge --> hing
  – free --> free
Porter Summary

• Con
  – many words with different meanings have common stems (e.g.; fabricate and fabric)
  – a lot of stems are not words

Dictionary based approaches (K-Stem)

• Using dictionaries to ensure that the generated stem is a valid word.
  – Develop some candidate words by removing the endings
  – Find the longest word that is in the dictionary that matches one of the candidates.
• Pro: This eliminates the Porter problem that many stems are not words.
• Con: Language dependent approach
Corpus-based Co-Occurrence

- Use Porter or other stemmer to stem terms
- Place words in potential classes
- Measure the frequency of co-occurrence of terms in the class
- Eliminate words from a class with a low co-occurrence
- Remaining classes form stemming rules

Corpus-based Co-Occurrence

- Pro
  - Language independent (no need of dictionary)
  - Based on assumption that terms in a class will co-occur with other terms “hippo” will co-occur with “hippos”
  - Improves effectiveness
- Con
  - Computationally expensive to build co-occurrence matrix (but you only do it every now and then)
N-grams

• Noise such as OCR (Optical Character Recognition) errors or misspelling lower the query processing accuracy in a term-based search.
• The premise is:
  – Substrings of a term may help to find a match in the noise cases
• Replace terms with n-grams
• Language-independent -- no stemming or stop word removal needed

5-Gram Example

• Q: What technique works on noise and misspelled words?
• D₁: N-grams work on noisy misspelled text.

| _work | spell | 8 terms are matched |
| _on_no | pelle | No stemming of work, noise |
| on_noi | elled | Partial match of misspelled word |
| n_nois | lled_ | |

33

34
N-gram Summary

• Pro
  – Language independent
  – Works on garbled text (OCR, etc.)

• Con
  – There can be a LOT of n-grams, dictionary may not fit in memory anymore \textit{(thus, only some are kept)}
  – Query processing requires more resources

Links

• Web documents contain \textit{link} information that is parsed and used for query processing and ranking (ex: pageRank,…).
  – \textit{Anchor text}
  – \textit{Inlinks} and \textit{outlinks}
Token Processing Summary

- Token Processing can make a difference in effectiveness
- It is often overlooked
- Language independence approach is preferred