Text Technologies for Data Science
INFR11145

Preprocessing

Instructor:
Youssef Al Hariri
Lecture Objectives

• Learn about and implement
• Standard text pre-processing steps:
  • Tokenisation
  • Stopping
  • Normalisation
  • Stemming
Indexing Process

Documents acquisition

Index creation

Document data store

document → unique ID
what can you store?
disk space? rights?
compression?

Index

Pre-processing

Text transformation

format conversion international?
which part contains “meaning”?
word units? stopping? stemming?

web-crawling provider feeds
RSS “feeds”
desktop/email

what data do we want?

© Addison Wesley, 2008
Preprocessing

Find the best text transformation technique (preprocessing) that will lead to better match between different forms of words in document and query
Getting ready for indexing?

• BOW, what is a word?

• In IR, we refer to word-elements as “terms”
  • word “preprocessing”
  • part of a word “pre”
  • number / code “INFR11145”

• Pre-processing steps before indexing:
  • Tokenisation
  • Stopping
  • Stemming

• Objective → identify the optimal form of the term to be indexed to achieve the best retrieval performance
Tokenisation

- **Input**: “Enemies, Romans; and Countrymen!”
- **Output**: Tokens
  - Friends
  - Romans
  - and
  - Countrymen

- Sentence → tokenization (splitting) → tokens
- A **token** is an **instance** of a sequence of characters
- **Typical technique**: split at non-letter characters
- Each such token is now a candidate for an index entry (term), after further processing
Issues in Tokenisation

• “Finland’s” capital → Finland? Finlands? Finland’s?

• Hewlett-Packard → one token or two?
  • state-of-the-art: break up hyphenated sequence.
  • co-education
  • lowercase, lower-case, lower case?
  • It can be effective to get the user to put in possible hyphens

• Numbers?
  • 3/20/91 vs. Mar. 20, 1991 vs. 20/3/91
  • This course code is INFR11145
  • (800) 234-2333
Issues in Tokenisation

• **URLs:**
  - http://www.bbc.co.uk

• **Social Media**
  - Black lives matter
  - #Black_lives_matter
  - #BlackLivesMatter
  - #blacklivesmatter
  - @blacklivesmatter

• **San Francisco**: one token or two?
  - How do you decide it is one token?
Tokenisation for different languages

• French → *L'ensemble* → one token or two?
  • *L ? L’ ? Le?*
  • Want *l’ensemble* to match with *un ensemble*
  • Until at least 2003, it didn’t on Google

• German → compounds
  • *Lebensversicherungsgesellschaftsangestellter*
    ‘life insurance company employee’
  • German retrieval systems benefit greatly from a *compound splitter*
    module → Can give a 15% performance boost for German

• Chinese and Japanese → no spaces between words:
  • 莎拉波娃现在居住在美国东南部的佛罗里达
  • Tokenisation → Segmentation
Tokenisation: common practice

• Just split at non-letter characters
• Add special cases if required
• Some applications have special setup
  • Social media: hashtags/mentions handled differently
  • URLs: no split, split at domain only, remove entirely!
  • Medical: protein & diseases names
Stopping (stop words removal)

• This is a very exciting lecture on the technologies of text

• Stop words: the most common words in collection
  → the, a, is, he, she, I, him, for, on, to, very, ...

• There are a lot of them ≈ 30-40% of text

• New stop words appear in specific domains
  • Tweets: RT → “RT @realDonaldTrump Mexico will …”
  • Patents: said, claim → “a said method that extracts ....”

• Stop words
  • influence on sentence structure
  • less influence on topic (aboutness)
Stopping: always apply?

- Sometimes very important:
  - Phrase queries: “Let it be”, “To be or not to be”
  - Relational queries:
    - flights to London from Edinburgh
    - flights from London to Edinburgh

- In Web search, trend is to keep them:
  - Good compression techniques means the space for including stop words in a system is very small
  - Good query optimization techniques mean you pay little at query time for including stop words.
  - Probabilistic retrieval models give them low weight.
Stopping: stop words

• Common practice in many applications → remove stop words

• There are common stop words list for each language
  • NLTK (python)
  • [http://members.unine.ch/jacques.savoy/clef/index.html](http://members.unine.ch/jacques.savoy/clef/index.html)

• There are special stop words list for some applications

• How to create your list:
  • Sort all terms in a collection by frequency
  • Manually select the possible stop words from top $N$ terms
Normalisation

• **Objective** → make words with different surface forms look the same

• Document: “this is my CAR!!”
  Query: “car” should “car” match “CAR”?

• Sentence → **tokenisation** → **tokens** → **normalisation** → **terms** to be indexed

• Same tokenisation/normalisation steps should be applied to documents & queries
Case folding and equivalents

- “A” & “a” are different strings for computers
- Case folding: convert all letters to lower case
  - CAR, Car, caR → car
  - Windows → windows, should we do that?

- Diacritics/Accents removal
  - French: Château → chateau
  - German: Tübingen → tuebingen
  - Arabic: كتاب → كتاب
Equivalence Classes

- U.S.A. → USA
- Ph.D. → PhD
- 92.3 → 923? 92 3?
- multi-disciplinary → multidisplinary ← multi disciplinary

- The most important criteria:
  - Be consistent between documents & queries
  - Try to follow users’ most common behaviour
Stemming

• Search for: “play” should it match: “played”, “playing”, “player”?

• Many morphological variations of words
  • inflectional (plurals, tenses)
  • derivational (making verbs, nouns, etc.)

• In most cases, aboutness does not change

• Stemmers attempt to reduce morphological variations of words to a common stem
  • usually involves removing suffixes (in English)

• Can be done at indexing time or as part of query processing (like stopwords)
Stemming

- Usually, it achieves 5-10% improvement in retrieval effectiveness, e.g. English
- For highly inflected languages, it is more critical:
  - 30% improvement in Finnish IR
  - 50% improvement in Arabic IR

They are Peter’s children
The children behaved well
Her children are cute
My children are funny
We have to save our children
Patents and children are happy
He loves his children
His children loves him

هؤلاء أبناء بيتر
الأبناء تصرفوا جيدا
أبناءها لطاف
أبنائي ظرفاء
علينا أن نحمي أبناءنا
الأباء والأبناء سعداء
هو يحب أبناءه
أبناؤه يحبونه
Stemming

• Two basic types
  • Dictionary-based: uses lists of related words
  • Algorithmic: uses program to determine related words

• Algorithmic stemmers
  • suffix-s: remove ‘s’ endings assuming plural
  • e.g., cats $\rightarrow$ cat, lakes $\rightarrow$ lake, windows $\rightarrow$ window
  • Many false negatives: supplies $\rightarrow$ supplie
  • Some false positives: James $\rightarrow$ Jame
Porter Stemmer

• Most common algorithm for stemming English
• Conventions + 5 phases of reductions
  • phases applied sequentially
  • each phase consists of a set of commands
  • sample convention:
    of the rules in a compound command, select the one that applies to the longest suffix.
• Example rules in Porter stemmer
  • sses → ss  (processes → process)
  • y → i  (reply → repli)
  • ies → i  (replies → repli)
  • ement → null  (replacement → replac)
Stemmed words are misspelled!!

- repli, replac, suppli, inform retriev, anim
- These are not words anymore, these are terms
- These terms are not seen by the user, but just used by the IR system (search engine)
- These represent the optimal form for a better match between different surface forms of a term
  - e.g. replac $\rightarrow$ replace, replaces, replaced, replacing, replacer, replacers, replacement, replacements.
Pre-processing: Common practice

• Tokenisation: split at non-letter characters
  • Basic regular expression
    → process \w and neglect anything else
  • For tweets, you might want to keep “#” and “@”

• Remove stop words
  • find a common list, and filter these words out

• Apply case folding
  • One command in Perl or Python: lc($string)

• Apply Porter stemmer
  • Other stemmers are available, but Porter is the most famous with many implementations available in different programming languages
Limitations

• Irregular verbs:
  • saw → see
  • went → go

• Different spellings
  • colour vs. color
  • tokenisation vs. tokenization
  • Television vs. TV

• Synonyms
  • car vs. vehicle
  • UK vs. Britain

• Solution → Query expansion …
Asymmetric Expansion

• Maintains relations between unnormalised tokens
• An alternative to equivalence classing
• An example of where this may be useful
  • query: window search: window, windows
  • query: windows search: windows, Windows
  • query: Windows search: Windows

• Potentially more powerful, but less efficient
  • More vocabulary, longer query

• Can be less effective:
  • Inaccurate stats on terms (“car” ≠ “Car”)

Youssef Al Hariri, TTDS 2023/2024
Summary

- Text pre-processing before IR:
  - Tokenisation → Stopping → Stemming

This is an example sentence of how the pre-processing is applied to text in information retrieval. It includes: Tokenization, Stop Word Removal, and Stemming.
## Practical

<table>
<thead>
<tr>
<th>Collection</th>
<th>Original</th>
<th></th>
<th>After Pre-processing</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td># words</td>
<td>File size</td>
<td># words</td>
<td>File size</td>
</tr>
<tr>
<td>Bible</td>
<td>824,054</td>
<td>4.24 MB</td>
<td>358,112</td>
<td>2.05 MB</td>
</tr>
<tr>
<td>Wiki abstracts</td>
<td>78,137,597</td>
<td>472 MB</td>
<td>47,741,065</td>
<td>309 MB</td>
</tr>
</tbody>
</table>
Resources

• Text book 1: Intro to IR, Chapter 2 → 2.2.4
• Text book 2: IR in Practice, chapter 4

• Lab 1 → Implement what learnt in these two lectures
  START NOW, support on PIAZZA

• Optional reading:
  
  *if you think English pre-processing is hard!
  
  - Arabic Information Retrieval. Darwish & Magdy
Next lecture

• Indexing:
  How to build an index!

• Assignment 1 announcement:
  • Build indexing components
  • Today: build your pre-processing module!
  • Next time: build the index