Text Technologies for Data Science
INFR11145

Preprocessing

Instructor:
Walid Magdy

Lecture Objectives

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

Documents acquisition → Index creation → Index

- What data do we want?
- Document → unique ID
- What can you store?
- Disk space? rights? compression?
- a lookup table for quickly finding all docs containing a word

Text transformation

- Format conversion: international?
- Which part contains "meaning"?
- Word units? stopping? stemming?

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**: “Friends, 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?
  - 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: stop words**

- Common practice in many applications “remove them”
- 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 wait.
- You need them for:
  - Phrase queries: “King of Denmark”, “Let it be”, “To be or not to be”
  - “Relational” queries: “flights to London”

**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 \(\rightarrow\) car
  • Windows \(\rightarrow\) windows, should we do that?
  • Usually yes, users are so lazy
• Diacritics/Accents removal
  • French: Château \(\rightarrow\) chateau
  • German: Tuebingen vs. tuebingen
  • Arabic: كُتُب \(\rightarrow\) كتب

Equivalence Classes

• U.S.A. \(\rightarrow\) USA
• Ph.D. \(\rightarrow\) PhD
• 92.3 \(\rightarrow\) 923? 92 3?
  • multi-disciplinary \(\rightarrow\) multidisciplinary \(\leftarrow\) 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 طفلا
- علينا أن نحمي أبنائنا
- الآباء والأبناء سعداء
- هو يحب أبناءه
- أبناءه 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 → cat, lakes → lake, windows → window
  - Many false negatives: supplies → supplie
  - Some false positives: James → 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)
Pre-processing: Common practice

- Tokenisation: split at non-letter characters
  - One line of code in Perl
    → 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: 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 unnormalized 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”)

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.

example sentence pre process appli text inform retriev includ Token Stop Word Remov Stem
**Resources**

- Text book 1: Intro to IR, Chapter 2 → 2.2.4
- Text book 2: IR in Practice, chapter 4
- Stop words list for different languages: [http://members.unine.ch/jacques.savoy/clef/index.html](http://members.unine.ch/jacques.savoy/clef/index.html)
- 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