CS918: LECTURE 2

Text Preprocessing

Arkaitz Zubiaga, 8th October, 2018

RECAP: REGULAR EXPRESSIONS

- A formal language for specifying **text patterns**: for **searching or replacing** text.

- For instance:
  - Find URLs in text.
  - Find all numbers in document.

    65,640,000 people live in the UK’s 4 nations.

    [1-9][0-9]{0,2}(,[0-9]{3})*

RECAP: REGULAR EXPRESSIONS

- ELIZA: Early application using **regular expressions** to develop a **chatbot for psychotherapy**.

RECAP: REGULAR EXPRESSIONS

- **Originally**, they were used for **generating linguistic rules**.
  - e.g. to “You are X”, chatbot replies with “why am I X?” for any X = [A-Za-z]*
  - This however **doesn’t scale** for large collections.

- **Nowadays**, mostly used for **text preprocessing**:
  - **pattern matching and replacement** (URLs, numbers,...)
  - **tokenisation**.
  - etc.

LECTURE 2: CONTENTS

- Text preprocessing: why?
  - Word tokenisation.
  - Text normalisation.
  - Stopword removal.
  - Lemmatisation and stemming.
  - Sentence segmentation.

TEXT PREPROCESSING: WHY?

WARWICK

- Inconsistent use of words:

  Welcome to the UK

  Welcome to the U.K.

  Welcome to the uk

  Welcome to the u.k.

TEXT PREPROCESSING: WHY?

WARWICK

- Linguistic variations with similar meanings.

  I am **happy** today.

  I am **happier** than yesterday.

For sentiment analysis, maybe all we need to know is that both have the word “happy”, irrespective of the variation.

TEXT PREPROCESSING

- Every NLP task needs to do text preprocessing:
  - Stopword removal.
  - Segmenting/tokenising words in running text.
  - Normalising word formats.
  - Segmenting sentences in running text.

STOP WORD REMOVAL

- Some words are more **meaningful** than others.

I am excited to be a member of Team GB!

Words like **“to”**, **“be”**, **“a”**, **“of” are not very meaningful** for some analyses.

STOP WORD REMOVAL

- Words like **“to”, “be”, “a”, “of” are not very meaningful** for some analyses.
  - We call them “stop words”, i.e. most commonly used words that are not informative for some tasks.
  - If they’re not useful for our task, we may remove them.
  - NLTK provides a list of stop words.

CORPORA

- **Corpus:** collection or dataset of text or speech. One or more documents, e.g. corpus with all of Shakespeare’s works.

- We can split each document/text into **sentences**, and these sentences into **words**.

SENTENCES

- **Sentences:** shortest sequence of words that are grouped together to convey some grammatically correct self-contained meaning.

- How do we about splitting a text into sentences?

  - For practical purposes, sequence between full stops or “?|!|:|;”.

  - We can do more advance segmentation, e.g. break long sentences with conjunctions like “and” or “or”.

HOW MANY WORDS IN A SENTENCE?

- It can be as simple as **counting** the elements we get after **splitting a text by spaces**, but it depends.

- How many words in the following?

  My *cat is different from other cats.*

WORDS AND LEMMAS

- My **cat** is different from other **cats**.

- Cat and cats both have the same **lemma** (cat), but two different **wordforms**:
  - Cat: cat (lemma)
  - Cats: cat (lemma) + s (suffix)

WARWICK

HOW MANY WORDS?

- **Type:** a unique element of the vocabulary.
- **Token:** an instance of that type in the running text.

HOW MANY WORDS?

WARWICK

- **Type:** a unique element of the vocabulary.
- **Token:** an instance of that type in the running text.

The house on the hill is the best

8 tokens.

6 types: the, house, on, hill, is, best.

(3 of the tokens belong the same type, 'the')

If we remove stop words (on, the), these numbers will decrease.

WARWICK

CORPORA

- **N** = number of tokens
- **V** = vocabulary = set of types
  - **|V|** is the size of the vocabulary

	Tokens = N	**Types = |V|**
Switchboard phone conversations	2.4 million	20 thousand
Shakespeare	884,000	31 thousand
Google N-grams	1 trillion	13 million

WORD TOKENISATION

- The Complete Works of William Shakespeare (shakes.txt, available on module website):

Open ▾ shakes.txt ~/Desktop

```
The Project Gutenberg EBook of The Complete Works of William Shakespeare, by
William Shakespeare

This eBook is for the use of anyone anywhere at no cost and with
almost no restrictions whatsoever.  You may copy it, give it away or
re-use it under the terms of the Project Gutenberg License included
with this eBook or online at www.gutenberg.org

** This is a COPYRIGHTED Project Gutenberg eBook, Details Below **
**     Please follow the copyright guidelines in this file.     **

Title: The Complete Works of William Shakespeare

Author: William Shakespeare

Posting Date: September 1, 2011 [EBook #100]
Release Date: January, 1994

Language: English


*** START OF THIS PROJECT GUTENBERG EBOOK COMPLETE WORKS--WILLIAM SHAKESPEARE ***




Produced by World Library, Inc., from their Library of the Future
```

SIMPLE TOKENISATION IN UNIX

```
> tr -sc 'A-Za-z' '\n' < shakes.txt
```

- Having shakes.txt as input (< shakes.txt)
- Convert all non-alphabetic characters (-sc ‘A-Za-z’)
- Into new lines (‘\n’)

STEP 1: TOKENISING

```
> tr -sc 'A-Za-z' '\n' < shakes.txt | head
The
Project
Gutenberg
EBook
of
The
Complete
Works
of
...
```

STEP 2: SORTING

```
> tr -sc 'A-Za-z' '\n' < shakes.txt | sort | head
a
a
a
a
a
a
a
a
a
...
```

STEP 3: COUNTING

```
> tr -sc 'A-Za-z' '\n' < shakes.txt | sort | uniq -c | head
  12851 a
   1949 A
     25 Aaron
     72 AARON
      1 abaissiez
     10 abandon
      2 abandoned
      2 abase
      1 abash
...
```

STEP 3: SORT BY COUNT

```
> tr -sc 'A-Za-z' '\n' < shakes.txt | sort | uniq -c | sort -r -n | head
  23455 the
  22225 I
  18715 and
  16433 to
  15830 of
  12851 a
  12236 you
  10840 my
  10074 in
   8954 d  → what happened here?
...
```

STEP 4: LOWERCASING TEXT

```
> tr -sc 'A-Za-z' '\n' < shakes.txt | tr 'A-Z' 'a-z' | sort | uniq -c |
sort -r -n | head
  27843 the
  26847 and
  22538 i
  19882 to
  18307 of
  14800 a
  13928 you
  12490 my
  11563 that
  11183 in
...
```

ISSUES IN TOKENISATION

- England’s capital → England, Englands, England’s
- what’re, I’m, isn’t → What are, I am, is not
- Hewlett-Packard → Hewlett Packard
- state-of-the-art → state of the art ?
- lower-case, lowercase, lower case
- Leamington Spa → one token or two?
- U.K./UK, U.S.A./USA

PENN TREEBANK TOKENISATION

WARWICK

- “The Leamington Spa-based restaurant,” they said, “doesn’t charge £10”.

  “ The Leamington Spa-based restaurant , “ they said , “ does n’t charge £ 10 “ .

  Uses regular expressions:
  ftp://ftp.cis.upenn.edu/pub/treebank/public_html/tokenization.html

TOKENISATION: LANGUAGE ISSUES

- French:
  - **L'ensemble** → one token or two?
  - L ? L’ ? Le ?
  - We want l’ensemble to match other instances of ensemble

- German noun compounds are not segmented:
  - **Lebensversicherungsgesellschaftsangestellter**
  - ‘life insurance company employee’
  - German information retrieval needs compound splitter

TOKENISATION IN CHINESE

WARWICK

- Also called **Word Segmentation**.

- Chinese words are composed of characters:
  - Characters are generally 1 syllable and 1 morpheme.
  - Average word is 2.4 characters long.

- Standard baseline segmentation algorithm:
  - **Maximum Matching** (also called Greedy)

MAXIMUM MATCHING ALGORITHM

WARWICK

- Given a wordlist (dictionary) of Chinese, and a string as input:

1) Start a pointer at the beginning of the string.
2) Find the longest word in dictionary that matches the string starting at pointer.
3) Move the pointer over the word in string.
4) Go to 2.

MAXIMUM MATCHING: EXAMPLE IN ENGLISH

WARWICK

- Thetabledownthere

  Longest dictionary word from the beginning is ‘theta’, but we wanted ‘the’.

  We get ‘Theta bled own there’

  We probably wanted ‘The table down there’ though!

It’s quite a **bad algorithm for English**!

MAXIMUM MATCHING: EXAMPLE IN CHINESE

WARWICK

- But it’s actually very good for Chinese:
  - 莎拉波娃现在居住在美国东南部的佛罗里达。
  - 莎拉波娃 现在 居住 在 美国 东南部 的 佛罗里达

WORD NORMALISATION AND STEMMING

NORMALISATION

- 2 types of normalisation. Let’s think of a Google search query.
  - Symmetric normalisation:
    - User searching for ‘U.S.A.’ or ‘USA’ most likely looking for the same.
  - Assymetric normalisation:
    - User enters ‘Windows’, we give results for ‘Windows’ (operating system)
    - User enters ‘windows’, do we give results for both ‘Windows’ and ‘windows’?

CASE FOLDING

- Often the case is not meaningful, e.g. ‘the’ vs ‘The’.
    - We may reduce all to **lowercase**.
    - Possible exception: upper case in mid-sentence?
        - e.g., **General Motors**
- But the case is sometimes important!
    - e.g. **US vs us**

LEMMATISATION AND STEMMING

- In both cases, we aim to **reduce vocabulary size**.
    - e.g. ‘cars’ and ‘car’ will both become ‘car’.

- **Lemmatisation:** finding dictionary headword form.

- **Stemming:** finding the stem by stripping off suffixes, usually using regular expressions.

LEMMATISATION

- Reduce inflections or variant forms to headword form:
  - am, are, is → be
  - car, cars, car's, cars' → car
  - those cars are really beautiful → those car be really beautiful

- PRO: we end up getting dictionary words.
- CON: costly, need to infer the meaning of each word.
  - Reading (verb) → read
  - Reading (city) → Reading

STEMMING

WARWICK

- Reduce by **following certain rules** (e.g. regular expressions):
  - am, are, is → am, ar, is
  - car, cars, car's, cars' → car, car, car ' s, car '
  - those cars are really beautiful → those car ar realli beauti

- CON: It does shorten words and reduce vocabulary, however **not always leading to dictionary words**!
- PRO: It’s **faster than a lemmatiser**.

PORTER: BEST-KNOWN ENGLISH STEMMER

Step 1a

```
sses → ss    posesses → posess
ies  → i     ponies   → poni
ss   → ss    posess   → posess
s    → ø    cats      → cat
```

Step 1b

```
(*v*)ing → ø  walking  → walk
               sing     →  sing
(*v*)ed  → ø  plastered → plaster
```

…

https://tartarus.org/martin/PorterStemmer/

PORTER: BEST-KNOWN ENGLISH STEMMER

WARWICK

- (*v*)ing → ø
- having → hav, living → liv, studying → study 
- king → ø, sing → ø, thing → ø 
- something → someth, morning → morn 

WARWICK

LEMMATISER VS STEMMER

for example compressed and compression are both accepted as equivalent to compress.

STEMMER:
for exampl compress and compress ar both accept as equival to compress

LEMMATISER:
for example compress and compress be both accept as equivalent to compress

SENTENCE SEGMENTATION

SENTENCE SEGMENTATION

- !, ? are relatively unambiguous

- But period "." is much more ambiguous
  - Sentence boundary
  - Abbreviations like Inc., etc. or PhD.
  - Numbers like .02% or 4.3

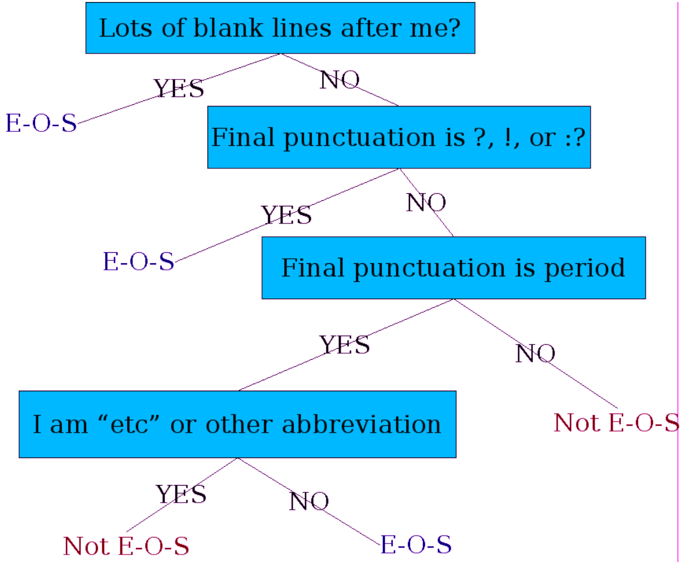
SENTENCE SEGMENTATION

- So how do we deal with this ambiguity?

- We can build a binary classifier:
  - Look at occurrences of '.'
  - Classifies EndOfSentence vs NotEndOfSentence. How?
    - Hand-written rules (if-then).
    - Regular expressions.
    - Machine learning.

SENTENCE SEGMENTATION: A DECISION TREE

- Deciding if a word is at the end of a sentence.



SEGMENTATION: MORE FEATURES

- Is the word following the period uppercased?
- What is the length of the word preceding the period?
- Are there more periods following? e.g. an ellipsis.
- Is there a space after the period?

ASSOCIATED READING

WARWICK

- Jurafsky, Daniel, and James H. Martin. 2009. Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics. 3rd edition. **Chapter 2.2-2.5**.

- Bird Steven, Ewan Klein, and Edward Loper. Natural Language Processing with Python. O'Reilly Media, Inc., 2009. **Chapters 1-3**.