Workshop 2

COMP90051 Natural Language Processing

Semester 1, 2020

About me

- Jun Wang
- I was a tutor of last semester SML
- I'm tutoring SML and NLP this semester
- jun5@unimelb.edu.au

Materials

- Download files
  - Workshop-02.pdf
  - 01-preprocessing.ipynb
  - 02-bpe.ipynb
- From Canvas - Modules - Workshops - Materials

Learning Outcomes

- Discuss text preprocess
  - Tokenisation
  - Normalisation
- Byte-pair Encoding
  - Algorithm
  - Implementation exercise
- Porter Stemmer

Text processing application

1. Give some examples of text processing applications that you use on a daily basis.

Text processing application

1. Give some examples of text processing applications that you use on a daily basis.

- Google translate
- Grammarly
- Spelling correction
- Spam filter for emails

Word Tokenisation

2. What is **tokenisation** and why is it important?

   (a) What are **stemming** and **lemmatisation**, and how are they different? Give examples from the `01-preprocessing` iPython notebook.

Word Tokenisation

- What is tokenisation
-

Word Tokenisation

- What is tokenisation
  - Segmenting text into tokens(words)
  - Example

```
'Topics to be covered include part-of-speech tagging, n-gram language
modelling, syntactic parsing and deep learning.'
```


```
['Topics', 'to', 'be', 'covered', 'include', 'part-of-speech',
'tagging,', 'n-gram', 'language', '', 'modelling,', 'syntactic',
'parsing', 'and', 'deep', 'learning.']
```

Word Tokenisation

- What is tokenisation
  - Segmenting text into tokens(words)
- Why we need tokenisation

Word Tokenisation

- What is tokenisation
  - Segmenting text into tokens(words)
- Why we need tokenisation
  - A text contain too much information
  - Human can break it into individual components
  - Machine should do the same

Word Tokenisation

- What is tokenisation
  - Segmenting text into tokens(words)
- Why we need tokenisation
  - A text contain too much information
  - Human can break it into individual components
  - Machine should do the same

Word Tokenisation

- What is tokenisation
  - Segmenting text into tokens(words)
- Why we need tokenisation
  - A text contain too much information
  - Human can break it into individual components
  - Machine should do the same
- Subword tokenisation
  - Byte-pair encoding (BPE)
  - We have BPE implementation exercises (02-bpe.ipynb)

Token Normalisation

- What is word normalisation?

Token Normalisation

- What is word normalisation?
  - Putting word into a standard format

Token Normalisation

- What is word normalisation?
  - Putting word into a standard format
- What can you do when normalise?

Token Normalisation

- What is word normalisation?
  - Putting word into a standard format
- What can you do when normalise?
  - Case folding
  - Correct spelling
  - Expanding abbreviations
  - Removing morphology
    - Stemming
    - Lemmatisation

Why preprocess

- Example
  - Find the similar sentence for sentence “The cat eats the rat” from the following sentences
    A. Cats eat rats.
    B. The dog eats the meat.
    C. The rat eats cheese.

Why preprocess

- Find the similar sentence for sentence “The cat eats the rat” from the following sentences
    A. Cats eat rats.
    B. The dog eats the meat.
    C. The rat eats cheese.
- Let’s consider only the number of identical words in sentences. (usually we don’t do this).
- If we don’t preprocessed sentences.

Why preprocess

- Find the similar sentence for sentence “The cat eats the rat” from the following sentences
    A. Cats eat rats.
    B. The dog eats the meat.
    C. The rat eats cheese.
- Let’s consider only the number of identical words in sentences. (usually we don’t do this)
- If we don’t preprocessed sentences.
    - A: 0 identical words
    - B: 3 identical words
    - C: 3 identical words
- B and C?

Why preprocess

- If we lowercased and lemmatised tokens
  - Query sentence: the cat eat the rat
  - A: cat eat rat
  - B: the dog eat the meat
  - C: the rat eat cheese
- Now:
  - A: 3 identical tokens
  - B: 3 identical token
  - C: 3 identical tokens
- All of them?
- What else can we do?

Stop words

- A list of unwanted words
  - Closed-class or function words
  - High frequency words
- Not appropriate when sequence is important

Why preprocess

- If we lowercased and lemmatised tokens and remove stop words
  - Query sentence: cat eat rat
  - A: cat eat rat
  - B: dog eat meat
  - C: rat eat cheese
- You may can remove “eat” because it has high frequency.
- Now:
  - A: 3 identical tokens
  - B: 1 identical token
  - C: 2 identical tokens
- A!!!

Stemming and lemmatisation

- What are stemming and lemmatisation?

Stemming and lemmatisation

- What are stemming and lemmatisation?
  - Lemmatisation: removing any inflection to reach the uninflected form, the *lemma*
  - Stemming: strips off all suffixes, leaving a *stem*

Stemming and lemmatisation

- Similar and different between stemming and lemmatisation

	Stemming	Lemmatisation
Garbage token		
Remove derivational morphology		
Remove inflectional morphology		
Works with a lexicon		
Remove or replace affixes (primarily suffixes)		
Transform a token into a normalised form		

Stemming and lemmatisation

- What are stemming and lemmatisation?

	Stemming	Lemmatisation
Garbage token	✓	✗
Remove derivational morphology		
Remove inflectional morphology		
Works with a lexicon		
Remove or replace affixes (primarily suffixes)		
Transform a token into a normalised form		

Stemming and lemmatisation

- What are stemming and lemmatisation?

	Stemming	Lemmatisation
Garbage token	✓	✗
Remove derivational morphology	usually	Doesn't usually
Remove inflectional morphology		
Works with a lexicon		
Remove or replace affixes (primarily suffixes)		
Transform a token into a normalised form		

Stemming and lemmatisation

- What are stemming and lemmatisation?

	Stemming	Lemmatisation
Garbage token	✓	✗
Remove derivational morphology	usually	Doesn't usually
Remove inflectional morphology	✓	✓
Works with a lexicon		
Remove or replace affixes (primarily suffixes)		
Transform a token into a normalised form		

Stemming and lemmatisation

- What are stemming and lemmatisation?

	Stemming	Lemmatisation
Garbage token	✓	✗
Remove derivational morphology	usually	Doesn't usually
Remove inflectional morphology	✓	✓
Works with a lexicon	✗	✓
Remove or replace affixes (primarily suffixes)		
Transform a token into a normalised form		

Stemming and lemmatisation

- What are stemming and lemmatisation?

	Stemming	Lemmatisation
Garbage token	✓	✗
Remove derivational morphology	usually	Doesn't usually
Remove inflectional morphology	✓	✓
Works with a lexicon	✗	✓
Remove or replace affixes (primarily suffixes)	✓	✓
Transform a token into a normalised form		

Stemming and lemmatisation

- What are stemming and lemmatisation?

	Stemming	Lemmatisation
Garbage token	✓	✗
Remove derivational morphology	usually	Doesn't usually
Remove inflectional morphology	✓	✓
Works with a lexicon	✗	✓
Remove or replace affixes (primarily suffixes)	✓	✓
Transform a token into a normalised form	✓	✓

Inflectional morphology

- Inflectional morphology is the systematic process by which tokens are altered to conform to certain grammatical constraints

- Cat -> cats
- Eat -> eats, ate
- Teach -> teaching

Derivational morphology

- Derivational morphology is the (semi-)systematic process by which we transform terms of one **class** into a different class.

- Teach -> teacher
- Personal -> personally

Inflectional and derivational

- Computers
- -s: inflectional morpheme
- -er: derivational morpheme

- Lemmatisation: Computers -> Computer
- Stemming: Computers -> comput

Porter Stemmer

- c = a single character of consonant
  - s, d, g
- v = a single character of vowel
  - a, o
- C = a sequence of consonants
  - s, ss, tr
- V = a sequence of vowels
  - a, ao, oo

Porter Stemmer

- Measure
  - Word can be represented as: [C] $(VC)^m$ [V]
  - m = measure
  - Example: PRIVATE

PR	I	V	A	T	E
C	V	C	V	C	V

[C] VC^2 [V]

Porter Stemmer

- Rule format: (condition) S1 -> S2
- e.g. (m>1) EMENT -> null
  - Replacement -> Replac

Porter Stemmer

- Rule format: (condition) S1 -> S2
- e.g. (*v*) ING -> null (m=1 and *o) -> E
  - Filing -> File
  - Failing -> Fail

- *o - the stem ends cvc, where the second c is not W, X or Y (e.g. -WIL, -HOP).