92586 Computational Linguistics
Lesson 2. Tokens

Alberto Barrón-Cedeño

Alma Mater Studiorum-Università di Bologna
a.barron@unibo.it   @_albarron_

26/02/2021

Table of Contents

Words

Words
What is a word?

Speech The smallest sequence of phonemes that can be uttered in isolation with objective or practical meaning

Text Sequences of graphemes ("letters") [. . .] delimited by spaces [. . .] or by other graphical conventions

https://en.wikipedia.org/wiki/Word

Simplistic definition
A word is a sequence of characters surrounded by spaces

Arguable, as multiple scholars claim; in particular across languages (Bender, 2013; Haspelmath, 2011)

Words

Lexicon

The set of all tokens (words!) in document d (or a corpus c)

Words

Tokenisers

We have a tokeniser, kindly provided by Church (1994)

```
tokens = re.findall('[A-Za-z]+', txt)
```

Python provides a "similar" tool

```
tokens = txt.split()
```

What if `txt` is the following?[1]

```
txt = """Thomas Jefferson started building Monticello at
the age of 26."""
```

</> **Let us see it working**

[1]Example borrowed from Lane et al. (2019, p. 34)

Words

Tokenisers

A better regular expression[2]

```
tokens = re.split(r'([-\s.,;!?])+', txt)
```

What if we have the following text?

```
txt = "Monticello wasn't designated as UNESCO World Heritage
Site until 1987"
```

[2]Borrowed from Lane et al. (2019, p. 43)

Words

NLTK

- ▶ One of the leading platforms to work with human language data in python[3]
- ▶ Easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet
- ▶ Suite of text processing libraries for classification, **tokenization**, stemming, tagging, parsing [. . .]

http://www.nltk.org/

[3]See spacy, stanza, huggingface

Words
Installing NLTK

```
$ pip install --user -U nltk
$ pip install --user -U numpy
$ python
>>> import nltk
```

Using (one of) the NLTK tokenisers

```
from nltk.tokenize import TreebankWordTokenizer
tokenizer = TreebankWordTokenizer()
sentence = "Monticello wasn't designated as UNESCO World
Heritage Site until 1987"
tokenizer.tokenize(sentence)
```

</> **Let us see it working**

Normalisation

Normalisation
Case folding

Ignoring differences in the spelling of a word which involves only capitalisation (Lane et al., 2019, p. 54)

PROS Now TEA==tea; the vocabulary is smaller

CONS The Joker is not a character any longer

</> **Let us see it working**

Normalisation
Stemming

"Eliminate the small meaning differences of pluralisation or possessive endings of words or [. . .] verb form" (Lane et al., 2019, p. 57)

```
import re
def stem(phrase):
    return ' '.join([re.findall('^(.*ss|.*?)(s)?$',
        word)[0][0].strip("'") for word in phrase.lower()
        .split()])

stem('houses')
stem("Doctor House's calls")
stem("stress")
```

</> **Let us see it working**

Normalisation
Stemming: Porter and Snowball

Once again, people have developed (and released) more sophisticated stemming algorithms
`https://tartarus.org/martin/PorterStemmer/`
`http://snowball.tartarus.org/`

```
from nltk.stem.porter import PorterStemmer
stemmer = PorterStemmer()
' '.join([stemmer.stem(w).strip("'") for w in
 "dish washer's washed dishes".split()])
```

Normalisation
Lemmatisation

Associating several words down to their semantic common root (adapted from (Lane et al., 2019, p. 59))

PROS	Stemming might alter the meaning of a word
CONS	It is more expensive; it requires a knowledge base of synonyms and endings, and POS

Normalisation
Lemmatisation: re-use, re-use!

```
import nltk
nltk.download('wordnet')

from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()

lemmatizer.lemmatize("better")
lemmatizer.lemmatize("better", pos="a")
```

</> **Let us see it working**

Normalisation
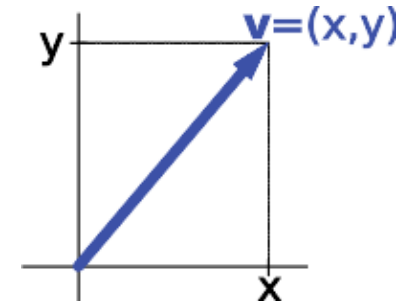
What kind of NLP are we using here?

rule-based or **statistical**

Representations

Representations

Vectors

An (Euclidean) vector is an entity endowed with a magnitude (the length of the line segment (A, B)) and a direction (the direction from A to B).

https://en.wikipedia.org/wiki/Vector_(mathematics_and_physics)
https://en.wikipedia.org/wiki/Vector_space

Representations

Bag of Words (BoW)

Turning words into numbers[4]

```
sentence = """Thomas Jefferson began building Monticello at
the age of 26."""

sentence_bow = {}
for token in sentence.split():
    sentence_bow[token] = 1
sorted(sentence_bow.items())
```

[4]From (Lane et al., 2019, p. 35)

Representations

Bag of Words (BoW)

Using **pandas** (data structures for data analysis, time series,and statistics)[5]

```
import pandas as pd

sentences = """Thomas Jefferson began building Monticello at
the age of 26.\n"""
sentences += """Construction was done mostly by local masons
and carpenters.\n"""
sentences += "He moved into the South Pavilion in 1770.\n"
sentences += """Turning Monticello into a neoclassical
masterpiece was Jefferson's obsession."""
corpus = {}
for i, sent in enumerate(sentences.split('\n')):
    corpus['sent{}'.format(i)] = dict((tok, 1) for tok in
        sent.split())
```

Representations

One-Hot Vectors

Turning words into numbers[6]

```python
import numpy as np
sentence = "Thomas Jefferson began building Monticello at
the age of 26."
token_sequence = str.split(sentence)
vocab = sorted(set(token_sequence))
', '.join(vocab)

num_tokens = len(token_sequence)
vocab_size = len(vocab)
onehot_vectors = np.zeros((num_tokens, vocab_size), int)
for i, word in enumerate(token_sequence):
    onehot_vectors[i, vocab.index(word)] = 1

' '.join(vocab)
onehot_vectors
```

[6]From (Lane et al., 2019, p. 35)

Representations

One-Hot Vectors

Turning words into numbers[7]

```python
import pandas as pd
pd.DataFrame(onehot_vectors, columns=vocab)
```

[7]From (Lane et al., 2019, p. 35)

References

Bender, E. M.
    2013. *Linguistic Fundamentals for Natural Language Processing:
    100 Essentials from Morphology and Syntax*. Morgan &
    Claypool Publishers.

Church, K.
    1994. UNIX for poets.

Haspelmath, M.
    2011. The indeterminacy of word segmentation and the nature
    of morphology and syntax. *Folia Linguistica*, 45.

Lane, H., C. Howard, and H. Hapkem
    2019. *Natural Language Processing in Action*. Shelter Island,
    NY: Manning Publication Co.