ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA
CAMPUS DI FORLÌ

91258 / B0385
Natural Language Processing

Lesson 2. Tokens

Alberto Barrón-Cedeño
a.barron@unibo.it

01/10/2024
2024/2025

Table of Contents

1. Words

2. Normalisation

3. Representations

Words

Words

What is a word?

The basic element of language that carries an objective or practical meaning, can be used on its own, and is uninterruptible

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

Text Sequence of graphemes (“letters”) [. . .] delimited by spaces [. . .] or by other graphical conventions

`https://en.wikipedia.org/wiki/Word` (old version)

Simplistic operational 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)[1]

[1] In general, lowercase symbols represent single instances; uppercase ones represent collections

Words

Tokenisers

We can use a simple tokeniser, kindly provided by Church (1994)[2]

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

Python provides a "similar" tool

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

What if `txt` is the following?[3]

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

</> Let us see it working

[2] Refer to PBR / APS / P4P

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

Words

Tokenisers

Building a better regular expression[4]

```
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"
```

</> Let us see it working

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

Words

The NLTK library

- One of the leading platforms to work with human language data in python
- 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

Words

The Spacy library

- “Industrial-strength Natural Language Processing”
- Support for 66+ languages
- Pre-trained word vectors and modules for tokenisation, lemmatisation, tagging, parsing [. . .]

https://spacy.io

Words

Installing NLTK and spacy

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

```
$ pip install --user -U spacy
$ python
>>> import spacy
```

Words

Using (one of the) spacy tokenisers

```
# loading the library
import spacy

# downloading the model
import spacy.cli
spacy.cli.download("en_core_web_sm")
```

```
nlp = spacy.load("en_core_web_sm")
doc = nlp(txt)
print([token.text for token in doc])
```

</> Let us see it work

Words

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)
```

Normalisation

Normalisation

Case folding

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

```
# We know how to deal with this, don't we?
```

PROS `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()])
```

</> Let us see it working

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 part-of-speech tags

Normalisation

Lemmatisation: re-use, re-use!

The NLTK way

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

from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()

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

The spacy way

```
doc = nlp("better")
print([token.lemma_ for token in doc])
```

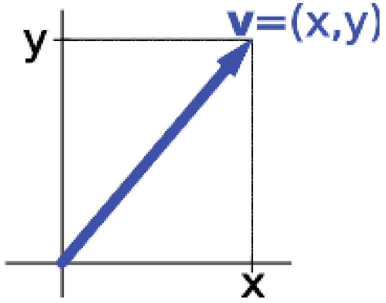
</> Let us see them working

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[5]

```
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())
```

</> Let us see it working

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

Representations

Bag of Words (BoW)

Using pandas (data structures for data analysis, time series, statistics)[6]

```
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())
df = pd.DataFrame.from_records(corpus).fillna(0).astype(int)
                                                .T
df[df.columns[:10]]
```

</> Let us see it working

[6] From (Lane et al., 2019, p. 41)

Representations

One-Hot Vectors

Turning words into numbers[7]

```
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))
print(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
```

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

Representations

One-Hot Vectors

Turning words into numbers[8]

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

[8] 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.