11 - Natural Language Processing

Data Science Practicum 2021/22, Lesson 11

Marko Tkalčič

Univerza na Primorskem

Table of Contents

Natural Language Processing

- Human-generated text -> machine usable information (features)
- Word: delimited string
- Term: a normalized word (usually used in IR indexes)
- Token: an instance of a word or a term
  - *Hello world, I am happy to be here*
    - Hello
    - world
    - I
    - am
    - happy
    - to
    - be
    - here

NLP

- Tokenization
  - C++, C#, B-52, `M*A*S*H`, O'Neill, aren't
- Stop-words removal: dropping common terms (they don't carry much information)
  - a, an, and, are, as, he, has, is, it, of, on etc.
- Frequency Distribution
- Normalization: connection, connected, connecting -> connect
  - Language-dependent
  - Stemming: reduce terms to their roots (chops off the derivational affixes)
  - Lemmatization: reduces words to their base word, which is linguistically correct lemmas.
    - more sophisticated than stemmer
    - better -> good (the lemma)

Tokenization

- Simple tokenizer

```
def simple_tokenizer(text)
    return text.split()
```

- Beware
  - C++, C#, B-52, M*A*S*H, O'Neill, aren't

NLTK

- NLTK package contains libraries for
  - tokenization
  - stemming
  - tagging
  - parsing

Tokenization

```
import nltk
#nltk.download('punkt')

def nltk_tokenizer(text):
    tokens = nltk.word_tokenize(text)
    return tokens

def simple_tokenizer(text):
    return text.split()

text = "Haven't you seen yesterday's episode of M*A*S*H?"
print(simple_tokenizer(text))
print(nltk_tokenizer(text))
```

```
["Haven't", 'you', 'seen', "yesterday's", 'episode', 'of', 'M*A*S*H?']
['Have', "n't", 'you', 'seen', 'yesterday', "'s", 'episode', 'of', 'M', '*', 'A', '*', 'S', '*', 'H', '?']
```

Sentence-based tokenization

```
from nltk.tokenize import sent_tokenize
token_text = sent_tokenize(text)
```

Stop-words removal

```
import nltk
#nltk.download('stopwords')

stop = nltk.corpus.stopwords.words("english")
stop
```

```
['i',
 'me',
 'my',
 'myself',
 'we',
 'our',
 'ours',
 'ourselves',
 'you',
 "you're",
 ...
```

Stop-words removal

```
import nltk
#nltk.download('stopwords')

stop = nltk.corpus.stopwords.words("english")

def nltk_tokenizer(text):
    tokens = nltk.word_tokenize(text)
    return tokens


def stopw_rem(tokens):
    out = []
    for tok in tokens:
        if tok not in stop:
            out.append(tok)
    return out

text = "Haven't you seen yesterday's episode of M*A*S*H?"
tokenized = nltk_tokenizer(text)
removed = stopw_rem(tokenized)
print(removed)
```

```
['Have', "n't", 'seen', 'yesterday', "'s", 'episode', 'M', '*', 'A', '*', 'S', '*', 'H', '?']
```

Stop-words removal

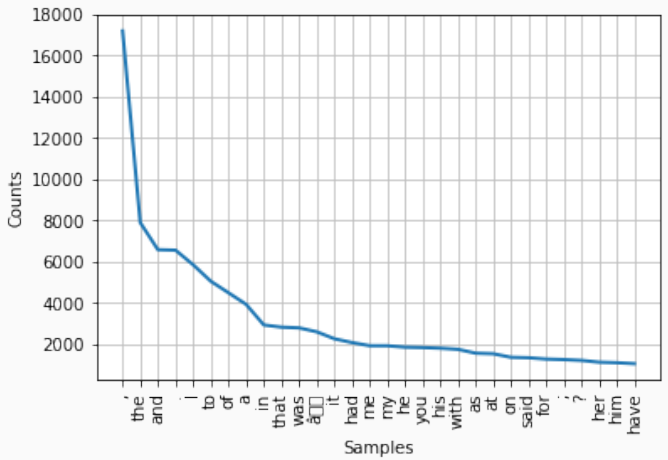
```
def stopw_rem(tokens):
    out = []
    for tok in tokens:
        if tok not in stop:
            out.append(tok)
    return out
```

```
def stopw_rem(tokens):
    out = [tok for tok in tokens if tok not in stop]
    return out
```

Frequency Distribution

```
# Frequency Distribution
import requests
from nltk.probability import FreqDist
import matplotlib.pyplot as plt

url = "http://www.gutenberg.org/files/1400/1400-0.txt"
request = requests.get(url)
great_expectations = request.text

cp_tokens = nltk.word_tokenize(great_expectations)

fdist = FreqDist(cp_tokens)
print(fdist)
print(fdist.most_common(5))
fdist.plot(30,cumulative=False)
plt.show()
```

```
<FreqDist with 14608 samples and 218931 outcomes>
[(',', 17197), ('the', 7901), ('and', 6585), ('.', 6563), ('I', 5853)]
```



Stemming

- Process of linguistic normalization, which reduces words to their word root word or chops off the derivational affixes
- connection, connected, connecting word reduce to a common word “connect”.

```
# Stemming
import nltk
from nltk.stem import PorterStemmer
from nltk.tokenize import sent_tokenize, word_tokenize

ps = PorterStemmer()
stop = nltk.corpus.stopwords.words("english")

text="""Hello Mr. Smith, how are you doing today? The weather is great, and city is awesome.
The sky is pinkish-blue. You shouldn't eat cardboard"""
tokenized = nltk.word_tokenize(text)
filtered = [tok for tok in tokenized if tok not in stop]
stemmed_words=[]
for w in filtered:
    stemmed_words.append(ps.stem(w))

print("Filtered Sentence:",filtered)
print("Stemmed Sentence:",stemmed_words)
```

```
Filtered Sentence: ['Hello', 'Mr.', 'Smith', ',', 'today', '?', 'The', 'weather', 'great', ',', 'city', 'awesome',
'.', 'The', 'sky', 'pinkish-blue', '.', 'You', "n't", 'eat', 'cardboard']
Stemmed Sentence: ['hello', 'mr.', 'smith', ',', 'today', '?', 'the', 'weather', 'great', ',', 'citi', 'awesom',
'.', 'the', 'sky', 'pinkish-blu', '.', 'you', "n't", 'eat', 'cardboard']
```

Lemmatization

- Stemming may create non-real words: Example:
  - are -> ar
  - thus -> thu
  - example -> exampl
- Lemmatization reduces words to their base word, which is a linguistically correct lemma
  - better -> good

Lemmatization

```
from nltk.stem.wordnet import WordNetLemmatizer
from nltk.stem.porter import PorterStemmer
import nltk
from nltk.stem import PorterStemmer
from nltk.tokenize import sent_tokenize, word_tokenize

#nltk.download('wordnet')

lem = WordNetLemmatizer()
stem = PorterStemmer()
ps = PorterStemmer()
stop = nltk.corpus.stopwords.words("english")

text = "The boy's cars are of different colors"
tokenized = nltk.word_tokenize(text)
filtered = tokenized
stemmed_words=[]
for w in filtered:
    stemmed_words.append(ps.stem(w))
lemmatized_words=[]
for w in filtered:
    lemmatized_words.append(lem.lemmatize(w,"v"))

word = "flying"
print("Lemmatized Words:",lemmatized_words)
print("Stemmed Word:",stemmed_words)
```

```
Lemmatized Words: ['The', 'boy', "'s", 'cars', 'be', 'of', 'different', 'color']
Stemmed Word: ['the', 'boy', "'s", 'car', 'are', 'of', 'differ', 'color']
```

Table of Contents

Exercise

- Write a Python NLTK program to split the text sentence/paragraph into a list of words.
- Make up a text with 4 sentences

Exercise

- Write a Python NLTK program to split the text sentence/paragraph into a list of words.
- Make up a text with 4 sentences

```
text = '''Joe waited for the train. The train was late. Mary and Samantha took the bus.
I looked for Mary and Samantha at the bus station.
'''
print("\nOriginal string:")
print(text)
from nltk.tokenize import sent_tokenize
token_text = sent_tokenize(text)
print("\nSentence-tokenized copy in a list:")
print(token_text)
print("\nRead the list:")
for s in token_text:
    print(s)
```

```
Original string:

Joe waited for the train. The train was late.
Mary and Samantha took the bus.
I looked for Mary and Samantha at the bus station.

Sentence-tokenized copy in a list:
['\nJoe waited for the train.', 'The train was late.', 'Mary and Samantha took the bus.', 'I looked for Mary and Samantha at the b

Read the list:

Joe waited for the train.
The train was late.
Mary and Samantha took the bus.
I looked for Mary and Samantha at the bus station.
```

Exercise

- tokenize and stem the sixth paragraph of https://en.wikipedia.org/wiki/Charles_Darwin
- use nltk
- print unstemmed and stemmed tokens
- count the occurrences with FreqDist()
- print the words and frequencies

Exercise

- tokenize and stem the sixth paragraph of https://en.wikipedia.org/wiki/Charles_Darwin
- use nltk
- print unstemmed and stemmed tokens
- count the occurrences with FreqDist()
- print the words and frequencies

```
import nltk
import requests
from bs4 import BeautifulSoup

url = "https://en.wikipedia.org/wiki/Charles_Darwin"
response = requests.get(url)
html_soup = BeautifulSoup(response.text, 'html.parser')
reviews = html_soup.findAll('p')
tokens = nltk.word_tokenize(reviews[5].getText())
porter = nltk.stem.porter.PorterStemmer()
tokens_stemmed = [porter.stem(tok) for tok in tokens]
print(tokens)
print(tokens_stemmed)
fdist = nltk.FreqDist(tokens_stemmed)
print(fdist)
for item,count in fdist.items():
    print(item,count)
```

```
['Darwin', "'s", 'early', 'interest', 'in', 'nature', 'led', 'him', 'to', 'neglect', 'his', 'medical', 'education',
['darwin', "'s", 'earli', 'interest', 'in', 'natur', 'led', 'him', 'to', 'neglect', 'hi', 'medic', 'educ',
<FreqDist with 69 samples and 94 outcomes>
darwin 1
's 3
earli 1
```

Exercise

- Write a Python NLTK program to tokenize a twitter text.
  - Manually copy/paste a tweet
  - hint: `from nltk.tokenize import TweetTokenizer`

Exercise

- Write a Python NLTK program to tokenize a twitter text.
  - Manually copy/paste a tweet
  - hint: `from nltk.tokenize import TweetTokenizer`

```
from nltk.tokenize import TweetTokenizer
tknzr = TweetTokenizer(strip_handles=True, reduce_len=True)
tweet_text = "NoSQL introduction - w3resource http://bit.ly/1ngHC5F  #nosql #database #webdev"
print("\nOriginal Tweet:")
print(tweet_text)
result = tknzr.tokenize(tweet_text)
print("\nTokenize a twitter text:")
print(result)
```

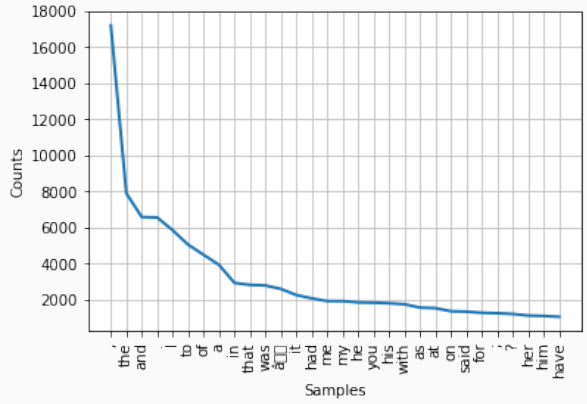
```
Original Tweet:
NoSQL introduction - w3resource http://bit.ly/1ngHC5F  #nosql #database #webdev

Tokenize a twitter text:
['NoSQL', 'introduction', '-', 'w3resource', 'http://bit.ly/1ngHC5F', '#nosql', '#database', '#webdev']
```

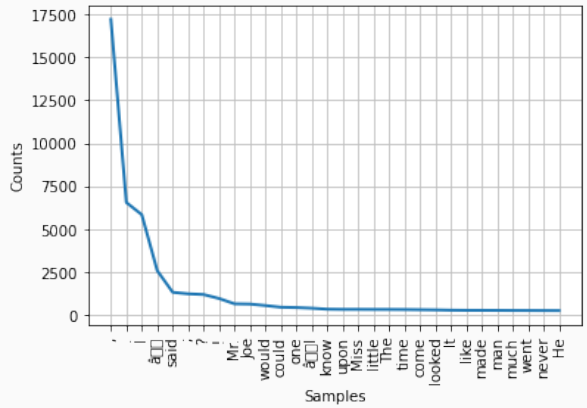
Exercise

- Pick a long text from project gutenberg (requests)
- Plot the frequency of words
- Apply stop-words removal
- Plot the frequency of words
- Apply lemmatization
- Plot frequency of words

Exercise

```
import requests
import nltk
from nltk.probability import FreqDist
import matplotlib.pyplot as plt
from nltk.stem.wordnet import WordNetLemmatizer


url = "http://www.gutenberg.org/files/1400/1400-0.txt"
request = requests.get(url)
great_expectations = request.text

#1
cp_tokens = nltk.word_tokenize(great_expectations)
fdist = FreqDist(cp_tokens)
fdist.plot(30,cumulative=False)
plt.show()

#2
stop = nltk.corpus.stopwords.words("english")
filtered = [tok for tok in cp_tokens if tok not in stop]
fdist = FreqDist(filtered)
fdist.plot(30,cumulative=False)
plt.show()

#3
lem = WordNetLemmatizer()
lemmatized_words=[]
for w in filtered:
    lemmatized_words.append(lem.lemmatize(w,"v"))
fdist = FreqDist(lemmatized_words)
fdist.plot(30,cumulative=False)
plt.show()
```





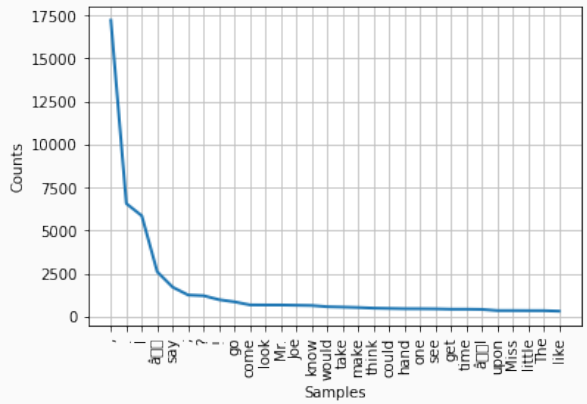


Table of Contents

Assignment

- url: `https://www.imdb.com/title/tt0068646/reviews?ref_=tt_ql_3`
- retrieve the reviews (requests, BeautifulSoup)
- load the file emotionWords.csv into a variable
- apply tokenization and stemming to emotion words and reviews
- for the first 10 reviews:
  - count the occurrences of each word from emotionWords in all the reviews
  - plot a histogram for the occurrences of each word in all the reviews (such as the one below)

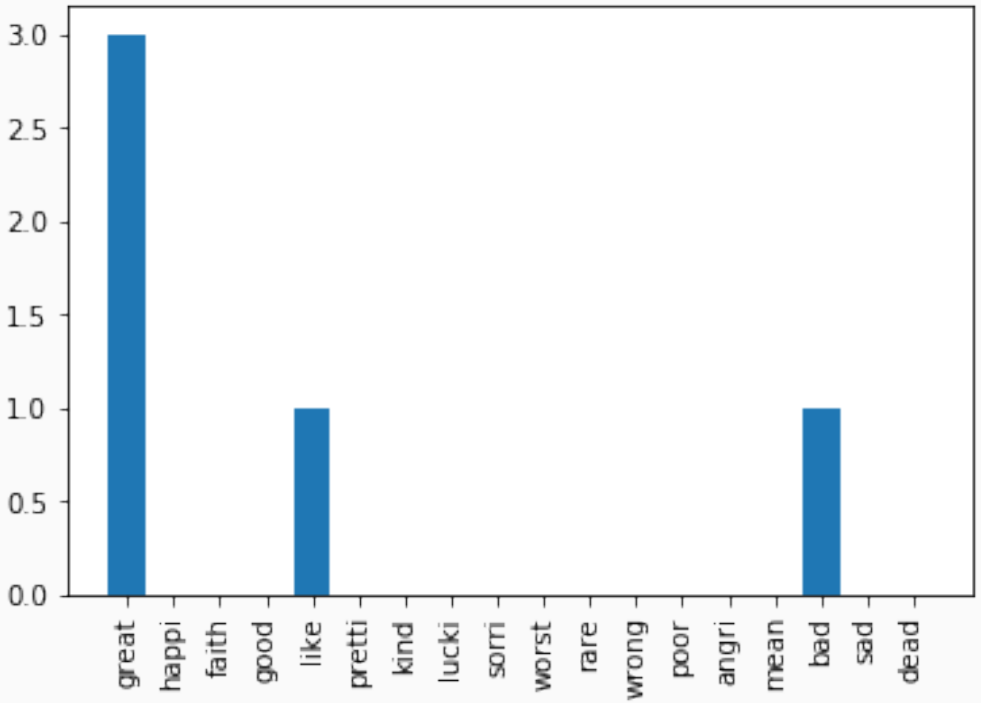


Table of Contents

References

Part of the material has been taken from the following sources. The usage of the referenced copyrighted work is in line with fair use since it is for nonprofit educational purposes.

- https://www.datacamp.com/community/tutorials/text-analytics-beginners-nltk
-