Text Visualization

March 14, 2019

Overview

- Text Processing Steps

- Practice in Jupyter Notebook
- https://github.com/junyuanjun/TextVisualizationLab

Text processing Steps

1. Text Pre-processing

2. Feature Extraction

3. Machine Learning

4. Visualization

Step 1: Text Preprocessing

Data Preprocessing

Tokenization: convert sentences to words.

Removing unnecessary punctuation, tags, and stop words (frequent words such as “a”, “the”, “is”, etc.).

Lexicon Normalization:

- Stemming: words are reduced to a root by dropping unnecessary characters, usually a suffix.
- Lemmatization: smarter stemming, but more time consuming.

Difference between Stemming and Lemmatization

- Stemming

```
The stemmed form of studies is: studi

The stemmed form of studying is: study
```

- Lemmatization

```
The lemmatized form of studies is: study

The lemmatized form of studying is: study
```

Let’s practice!

Dataset - The 20 newsgroups text dataset

Dataset Homepage: http://qwone.com/~jason/20Newsgroups/

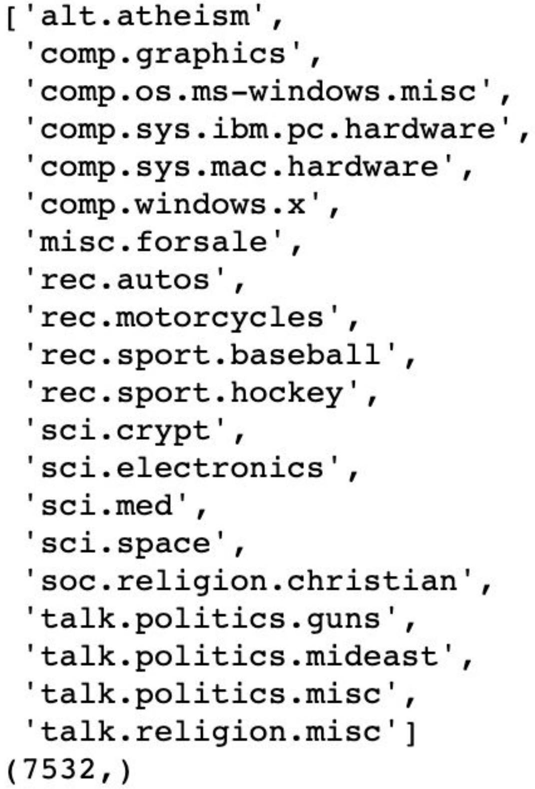
In this lab, we will use sklearn.datasets to load it:

https://scikit-learn.org/0.19/datasets/twenty_newsgroups.html

Loading Data

```
from sklearn.datasets import fetch_20newsgroups

from pprint import pprint



newsgroups_test = fetch_20newsgroups(subset='test')

pprint(list(newsgroups_test.target_names))

pprint(newsgroups_test.filenames.shape)
```

Loading Data



```
['alt.atheism',
 'comp.graphics',
 'comp.os.ms-windows.misc',
 'comp.sys.ibm.pc.hardware',
 'comp.sys.mac.hardware',
 'comp.windows.x',
 'misc.forsale',
 'rec.autos',
 'rec.motorcycles',
 'rec.sport.baseball',
 'rec.sport.hockey',
 'sci.crypt',
 'sci.electronics',
 'sci.med',
 'sci.space',
 'soc.religion.christian',
 'talk.politics.guns',
 'talk.politics.mideast',
 'talk.politics.misc',
 'talk.religion.misc']
(7532,)
```

We will get an output like this screenshot shows here.

We can also choose specific categories of text:

```
categories = ['rec.sport.baseball',
'talk.politics.guns', 'misc.forsale']

newsgroups = fetch_20newsgroups(shuffle=True,
random_state=1, remove=('headers', 'footers',
'quotes'), categories = categories)

data = newsgroups.data
```

Package Installation

Packages for topic modelling and natural language processing

- Pip install gensim
- Pip install spacy
- Pip install nltk

Text Preprocessing

```
import gensim, spacy

import gensim.corpora as corpora

from gensim.utils import simple_preprocess

from gensim.models import CoherenceModel

import nltk
```

Preprocessing: Tokenization

```
data_words = list(map(gensim.utils.simple_preprocess, data))
```

What does gensim.utils.simple_preprocess do?

Check the documents:

https://tedboy.github.io/nlps/generated/generated/gensim.utils.simple_preprocess.html

- Convert a document into a list of tokens.
- lowercases, tokenizes, de-accents (optional).

Text preprocessing: define stop words

```
# NLTK Stop words

# first time: nltk.download('stopwords')

# if you already have the stopwords, run the one blow

from nltk.corpus import stopwords

stop_words = stopwords.words('english')

stop_words.extend(['com', 'from', 'subject', 're', 'edu', 'use', 'not',
'would', 'say', 'could', '_', 'be', 'know', 'good', 'go', 'get', 'do', 'done',
'try', 'many', 'some', 'nice', 'thank', 'think', 'see', 'rather', 'easy',
'easily', 'lot', 'lack', 'make', 'want', 'seem', 'run', 'need', 'even',
'right', 'line', 'even', 'also', 'may', 'take', 'come'])
```

Text preprocessing: Remove stop words

```
texts = [[word for word in simple_preprocess(str(doc)) if
word not in stop_words] for doc in texts]
```

Text preprocessing: Lemmatization

Download an English library for lemmatization

Run the following command in terminal to link word Library

- python -m spacy download en
- (may need sudo)

More information can be found here: https://spacy.io/models/en

Text preprocessing: Lemmatization

```
data_ready = []

# Initialize spacy 'en' model, keeping only tagger component needed for lemmatization

nlp = spacy.load('en', disable=['parser', 'ner'])

allowed_postags=['NOUN', 'ADJ', 'VERB', 'ADV']

for sent in data_words:

    # Parse the sentence using the loaded 'en' model object `nlp`. Extract the lemma for each token and join

    doc = nlp(" ".join(sent))

    data_ready.append([token.lemma_ for token in doc if token.pos_ in allowed_postags])

# remove stopwords once more after lemmatization

data_ready = [[word for word in simple_preprocess(str(doc)) if word not in stop_words] for doc in data_ready]
```

More options for Lemmatization

We used the function provided by the library spaCy.

More options are mentioned in this article:

https://www.machinelearningplus.com/nlp/lemmatization-examples-python/

Step 2: Feature Extraction

mapping from textual data to real valued vectors

Step 3: Machine Learning

Feature Extraction

- Bag of Word (BoW): simple but lose the context information
  - Occurrence Count
  - Tf-idf
- Word Embedding
- …

The feature extraction methods depends on the model you are going to run and the application you are going to build.

Occurrence Count

- Assign each word a unique number (id)
- Encode a document as a fixed-length vector with the length of the vocabulary of known words
- Put a count or frequency of each word in the encoded document

Then we get a high-dimensional vector for a document, where each dimension represents a word, the value of each dimension represents the count of this word in this document.

Topic Modeling: Latent Dirichlet Allocation (LDA)

LDA is a model for discovering topics, it requires data in the form of integer counts.

We do not cover the algorithm details here, if you want to know more information, a few good materials to check here:

http://blog.echen.me/2011/08/22/introduction-to-latent-dirichlet-allocation/

https://medium.com/@lettier/how-does-lda-work-ill-explain-using-emoji-108abf40fa7d

LDA - practice

For more information about the library:

https://radimrehurek.com/gensim/models/ldamodel.html

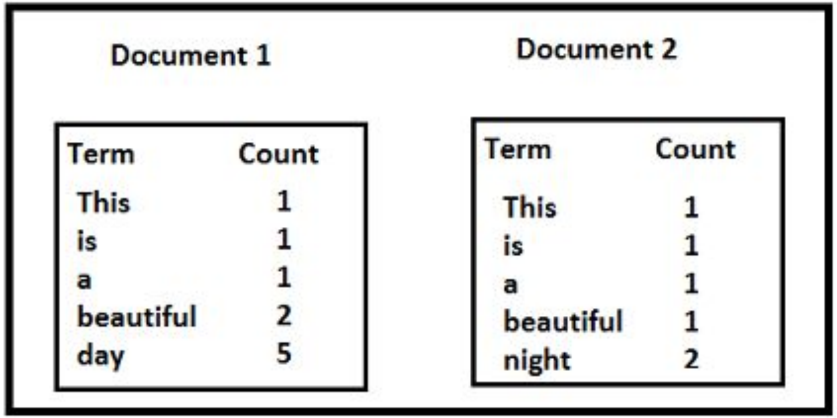
```
# Create Dictionary
id2word = corpora.Dictionary(data_ready)

# Create Corpus: Term Document Frequency
corpus = [id2word.doc2bow(text) for text in data_ready]

# Build LDA model
lda_model = gensim.models.ldamodel.LdaModel(corpus=corpus, id2word=id2word,
                                            num_topics=4, random_state=100, update_every=1, chunksize=10,
                                            passes=10,  alpha='symmetric', iterations=100, per_word_topics=True)

# Check the result
pprint(lda_model.print_topics())
```

Term Frequency-Inverse Document Frequency (TF-IDF)

One of the most important concepts to know for text mining.

- Frequency (TF) = (Number of times term t appears in a document)/(Number of terms in the document)
- Inverse Document Frequency (IDF) = **log(N/n)**, where, **N** is the number of documents and **n** is the number of documents a term t has appeared in. The IDF of a rare word is high, whereas the IDF of a frequent word is likely to be low. Thus having the effect of highlighting words that are distinct.
- We calculate TF-IDF value of a term as = TF * IDF

TF-IDF: example



Document 1

Term	Count
This	1
is	1
a	1
beautiful	2
day	5

Document 2

Term	Count
This	1
is	1
a	1
beautiful	1
night	2

TF('beautiful',Document1) = 2/10, IDF('beautiful')=log(2/2) = 0

TF(‘day’,Document1) = 5/10, IDF(‘day’)=log(2/1) = 0.30

TF-IDF(‘beautiful’, Document1) = (2/10)*0 = 0

TF-IDF(‘day’, Document1) = (5/10)*0.30 = 0.15

TF-IDF

Libraries:

- sklearn.feature_extraction.text.TfidfVectorizer: https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html
- gensim.models.tfidfmodel.TfidfModel: https://radimrehurek.com/gensim/models/tfidfmodel.html

Applications:

- Topic Modeling: https://medium.com/mlreview/topic-modeling-with-scikit-learn-e80d33668730
- Keyword extraction from a document
- Clustering Document

Word Embedding

BoW is simple and effective, but it ignore the context.

In order for the representation to be meaningful some empirical conditions should be satisfied:

- Similar words should be closer to each other in the vector space
- Allow word analogies: "King" - "Man" + "Woman" == "Queen"

A good tutorial here: https://nlpforhackers.io/word-embeddings/

Step 4: Visualization

Visualization Libraries

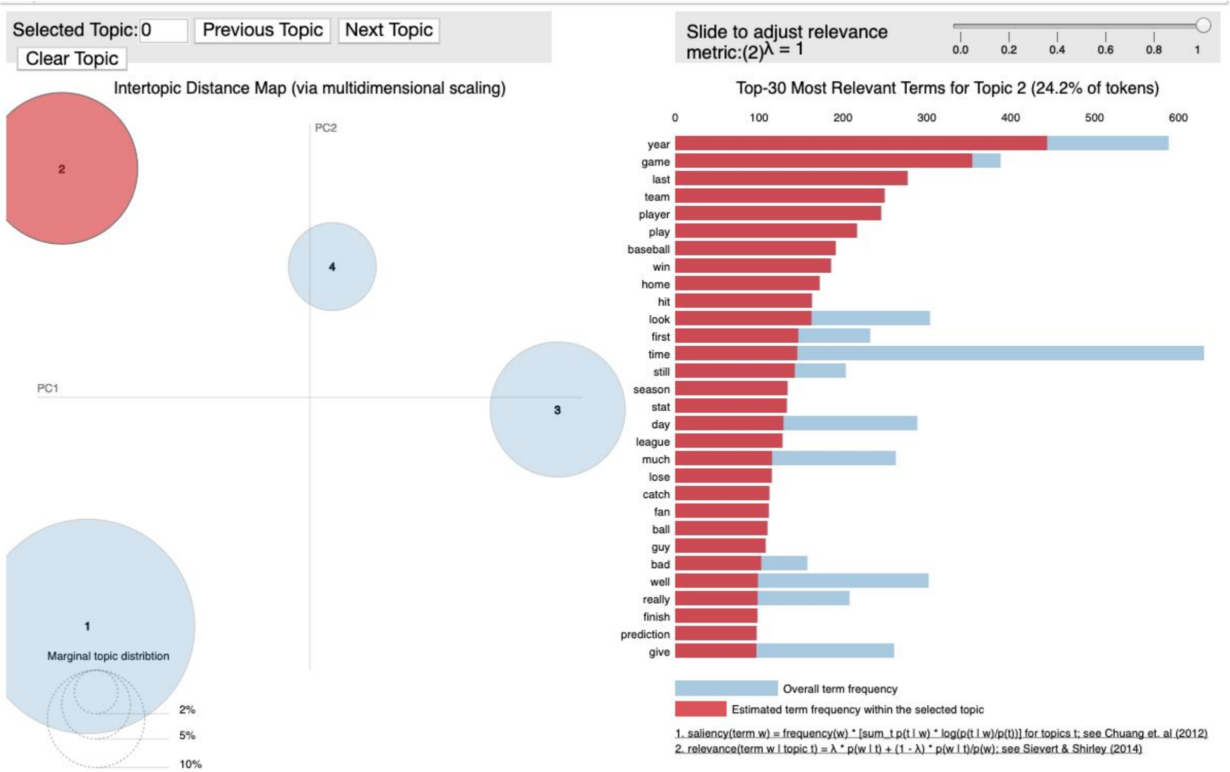
Packages for visualization

- Pip install pyLDAvis
- Pip install wordcloud

pyLDAvis

```
import pyLDAvis.gensim

pyLDAvis.enable_notebook()

vis = pyLDAvis.gensim.prepare(lda_model, corpus,
dictionary=lda_model.id2word)

display(vis)
```

pyLDAvis

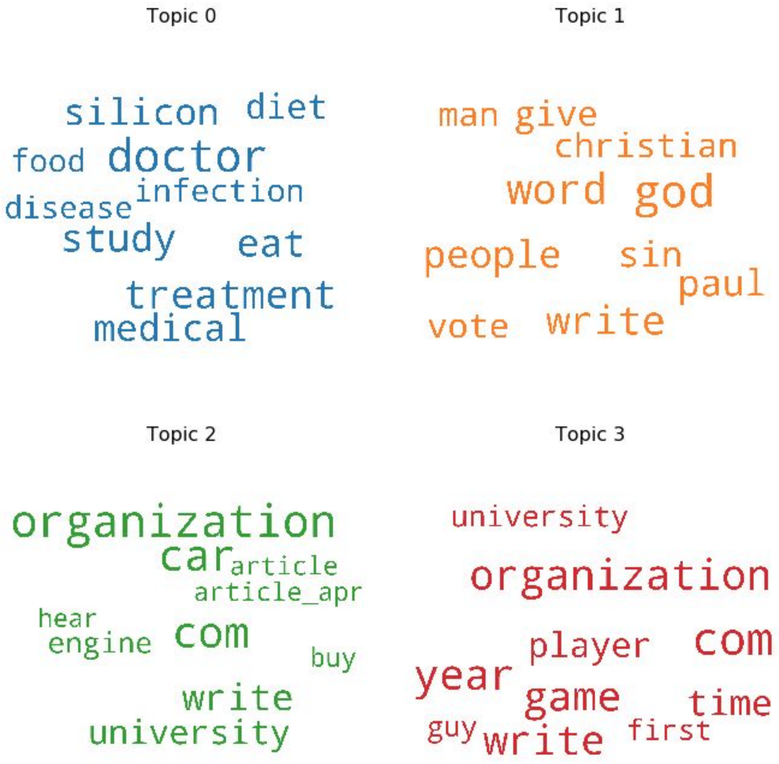


Selected Topic: 0

Slide to adjust relevance metric:(2) λ = 1

1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

wordcloud


Topic 0
silicon
diet
food
doctor
disease
infection
study
eat
treatment
medical
Topic 1
man
give
christian
word
god
people
sin
paul
vote
write
Topic 2
organization
car
article
article_apr
hear
engine
com
buy
write
university
Topic 3
university
organization
player
com
year
game
time
guy
write
first

Reference

- https://towardsdatascience.com/machine-learning-text-processing-1d5a2d638958
- http://www.davidsbatista.net/blog/2018/02/28/TfidfVectorizer/
- https://www.machinelearningplus.com/nlp/topic-modeling-visualization-how-to-present-results-lda-models/