Published in Towards Data Science

Susan Li Follow

May 31, 2018 · 5 min read · Listen

Topic Modeling and Latent Dirichlet Allocation (LDA) in Python

example of topic model and is used to classify text in a document to a particular topic. It builds a topic per document model and words per topic model, modeled as Dirichlet distributions.

Here we are going to apply LDA to a set of documents and split them into topics. Let's get started!

The Data

The data set we'll use is a list of over one million news headlines published over a period of 15 years and can be downloaded from Kaggle.

```
import pandas as pd

data = pd.read_csv('abcnews-date-text.csv', error_bad_lines=False);
data_text = data[['headline_text']]
data_text['index'] = data_text.index
documents = data_text
```

Take a peek of the data.

```
print(len(documents))
print(documents[:5])
```

1048575


0	aba decides against community broadcasting lic...	0
1	act fire witnesses must be aware of defamation	1
2	a g calls for infrastructure protection summit	2
3	air nz staff in aust strike for pay rise	3
4	air nz strike to affect australian travellers	4

Figure 1

Data Pre-processing

We will perform the following steps:

- **Tokenization:** Split the text into sentences and the sentences into words. Lowercase the words and remove punctuation.
- Words that have fewer than 3 characters are removed.
- All **stopwords** are removed.
- Words are **lemmatized** — words in third person are changed to first person and verbs in past and future tenses are changed into present.
- Words are **stemmed** — words are reduced to their root form.

Loading gensim and nltk libraries

```
import gensim
from gensim.utils import simple_preprocess
from gensim.parsing.preprocessing import STOPWORDS
from nltk.stem import WordNetLemmatizer, SnowballStemmer
from nltk.stem.porter import *
import numpy as np
```


[nltk_data] Downloading package wordnet to
[nltk_data] C:\Users\SusanLi\AppData\Roaming\nltk_data...
[nltk_data] Package wordnet is already up-to-date!

True

Write a function to perform lemmatize and stem preprocessing steps on the data set.

```
def lemmatize_stemming(text):
    return stemmer.stem(WordNetLemmatizer().lemmatize(text,
pos='v'))

def preprocess(text):
    result = []
    for token in gensim.utils.simple_preprocess(text):
        if token not in gensim.parsing.preprocessing.STOPWORDS and
len(token) > 3:
            result.append(lemmatize_stemming(token))
    return result
```

Select a document to preview after preprocessing.

```
doc_sample = documents[documents['index'] == 4310].values[0][0]

print('original document: ')
words = []
for word in doc_sample.split(' '):
    words.append(word)
print(words)
print('\n\n tokenized and lemmatized document: ')
print(preprocess(doc_sample))
```


tokenized and lemmatized document:

['rain', 'help', 'dampen', 'bushfir']

It worked!

Preprocess the headline text, saving the results as 'processed_docs'

```
processed_docs = documents['headline_text'].map(preprocess)
processed_docs[:10]
```

```
0              [decid, communiti, broadcast, licenc]
1                                [wit, awar, defam]
2            [call, infrastructur, protect, summit]
3                       [staff, aust, strike, rise]
4              [strike, affect, australian, travel]
5                [ambiti, olsson, win, tripl, jump]
6            [antic, delight, record, break, barca]
7    [aussi, qualifi, stosur, wast, memphi, match]
8             [aust, address, secur, council, iraq]
9                           [australia, lock, timet]
Name: headline_text, dtype: object
```

Figure 2

Bag of Words on the Data set

Create a dictionary from 'processed_docs' containing the number of times a word appears in the training set.

```
dictionary = gensim.corpora.Dictionary(processed_docs)
```


```
        break
```

0 broadcast

1 communiti

2 decid

3 licenc

4 awar

5 defam

6 wit

7 call

8 infrastructur

9 protect

10 summit

Gensim filter_extremes

Filter out tokens that appear in

- less than 15 documents (absolute number) or
- more than 0.5 documents (fraction of total corpus size, not absolute number).
- after the above two steps, keep only the first 100000 most frequent tokens.


For each document we create a dictionary reporting how many words and how many times those words appear. Save this to 'bow_corpus', then check our selected document earlier.

```
bow_corpus = [dictionary.doc2bow(doc) for doc in processed_docs]
bow_corpus[4310]
```

[(76, 1), (112, 1), (483, 1), (3998, 1)]

Preview Bag Of Words for our sample preprocessed document.

```
bow_doc_4310 = bow_corpus[4310]

for i in range(len(bow_doc_4310)):
    print("Word {} (\"{}\") appears {}
time.".format(bow_doc_4310[i][0],

dictionary[bow_doc_4310[i][0]],
bow_doc_4310[i][1]))
```

Word 76 ("bushfir") appears 1 time.

Word 112 ("help") appears 1 time.

Word 483 ("rain") appears 1 time.

Word 3998 ("dampen") appears 1 time.

TF-IDF

Create tf-idf model object using models.TfidfModel on 'bow_corpus' and save it to 'tfidf', then apply transformation to the entire corpus and call it 'corpus_tfidf'. Finally we preview TF-IDF scores for our first document.


```
tfidf = models.TfidfModel(bow_corpus)
corpus_tfidf = tfidf[bow_corpus]

from pprint import pprint

for doc in corpus_tfidf:
    pprint(doc)
    break
```

[(0, 0.5907943557842693),

(1, 0.3900924708457926),

(2, 0.49514546614015836),

(3, 0.5036078441840635)]

Running LDA using Bag of Words

Train our lda model using gensim.models.LdaMulticore and save it to 'lda_model'

```
lda_model = gensim.models.LdaMulticore(bow_corpus, num_topics=10,
id2word=dictionary, passes=2, workers=2)
```

For each topic, we will explore the words occuring in that topic and its relative weight.

```
for idx, topic in lda_model.print_topics(-1):
    print('Topic: {} \nWords: {}'.format(idx, topic))
```


```
11*"test" + 0.010*"australia" + 0.010*"hill"
Topic: 2
Words: 0.018*"rural" + 0.018*"council" + 0.015*"fund" + 0.014*"plan" + 0.013*"health" + 0.012*"chang" + 0.011*"nation" + 0.010
*"price" + 0.010*"servic" + 0.009*"say"
Topic: 3
Words: 0.025*"elect" + 0.022*"adelaid" + 0.012*"perth" + 0.011*"take" + 0.011*"say" + 0.010*"labor" + 0.010*"turnbul" + 0.009
*"vote" + 0.009*"royal" + 0.009*"time"
Topic: 4
Words: 0.032*"court" + 0.022*"face" + 0.020*"charg" + 0.020*"home" + 0.018*"tasmania" + 0.017*"murder" + 0.015*"trial" + 0.012
*"accus" + 0.012*"abus" + 0.012*"child"
Topic: 5
Words: 0.024*"countri" + 0.021*"hour" + 0.020*"australian" + 0.019*"warn" + 0.016*"live" + 0.013*"indigen" + 0.011*"call" + 0.0
09*"victorian" + 0.009*"campaign" + 0.008*"show"
Topic: 6
Words: 0.027*"south" + 0.024*"year" + 0.020*"interview" + 0.020*"north" + 0.019*"jail" + 0.018*"west" + 0.014*"island" + 0.013
*"australia" + 0.013*"victoria" + 0.010*"china"
Topic: 7
Words: 0.031*"queensland" + 0.029*"melbourn" + 0.018*"water" + 0.017*"claim" + 0.013*"hunter" + 0.012*"green" + 0.012*"resid" +
0.011*"darwin" + 0.010*"young" + 0.009*"plead"
Topic: 8
Words: 0.017*"attack" + 0.016*"kill" + 0.012*"victim" + 0.012*"violenc" + 0.010*"hobart" + 0.010*"rugbi" + 0.010*"secur" + 0.01
0*"say" + 0.009*"state" + 0.008*"domest"
Topic: 9
Words: 0.052*"polic" + 0.020*"crash" + 0.019*"death" + 0.017*"sydney" + 0.016*"miss" + 0.016*"woman" + 0.015*"die" + 0.015*"cha
```

Figure 3

Can you distinguish different topics using the words in each topic and their corresponding weights?

Running LDA using TF-IDF

```
lda_model_tfidf = gensim.models.LdaMulticore(corpus_tfidf,
num_topics=10, id2word=dictionary, passes=2, workers=4)

for idx, topic in lda_model_tfidf.print_topics(-1):
    print('Topic: {} Word: {}'.format(idx, topic))
```

```
Topic: 0 Word: 0.008*"octob" + 0.006*"search" + 0.006*"miss" + 0.006*"inquest" + 0.005*"stori" + 0.005*"jam" + 0.004*"john" +
0.004*"harvest" + 0.004*"australia" + 0.004*"world"
Topic: 1 Word: 0.006*"action" + 0.006*"violenc" + 0.006*"thursday" + 0.005*"domest" + 0.005*"cancer" + 0.005*"legal" + 0.005*"u
nion" + 0.005*"breakfast" + 0.005*"school" + 0.004*"student"
Topic: 2 Word: 0.023*"rural" + 0.018*"govern" + 0.013*"news" + 0.012*"podcast" + 0.008*"grandstand" + 0.008*"health" + 0.007*"b
udget" + 0.007*"busi" + 0.007*"nation" + 0.007*"fund"
Topic: 3 Word: 0.030*"countri" + 0.028*"hour" + 0.009*"sport" + 0.008*"septemb" + 0.008*"wednesday" + 0.007*"commiss" + 0.006
*"royal" + 0.006*"updat" + 0.006*"station" + 0.005*"bendigo"
Topic: 4 Word: 0.014*"south" + 0.009*"weather" + 0.009*"north" + 0.008*"west" + 0.008*"coast" + 0.008*"australia" + 0.006*"eas
t" + 0.006*"queensland" + 0.006*"storm" + 0.005*"season"
Topic: 5 Word: 0.008*"monday" + 0.008*"august" + 0.006*"babi" + 0.005*"shorten" + 0.005*"hobart" + 0.004*"victorian" + 0.004*"d
onald" + 0.004*"safe" + 0.004*"scott" + 0.004*"donat"
Topic: 6 Word: 0.022*"interview" + 0.013*"market" + 0.009*"share" + 0.008*"cattl" + 0.008*"trump" + 0.008*"turnbul" + 0.007*"no
vemb" + 0.007*"michael" + 0.006*"australian" + 0.006*"export"
Topic: 7 Word: 0.019*"crash" + 0.014*"kill" + 0.009*"fatal" + 0.009*"dead" + 0.007*"die" + 0.007*"truck" + 0.007*"polic" + 0.00
6*"attack" + 0.006*"injur" + 0.006*"bomb"
Topic: 8 Word: 0.008*"drum" + 0.007*"abbott" + 0.007*"farm" + 0.006*"dairi" + 0.006*"asylum" + 0.006*"tuesday" + 0.006*"water"
+ 0.006*"labor" + 0.006*"say" + 0.005*"plan"
```

Performance evaluation by classifying sample document using LDA Bag of Words model

We will check where our test document would be classified.

```
processed_docs[4310]
```

['rain', 'help', 'dampen', 'bushfir']

```
for index, score in sorted(lda_model[bow_corpus[4310]], key=lambda
tup: -1*tup[1]):
    print("\nScore: {}\t \nTopic: {}".format(score,
lda_model.print_topic(index, 10)))
```

```
Score: 0.41997694969177246
Topic: 0.017*"attack" + 0.016*"kill" + 0.012*"victim" + 0.012*"violenc" + 0.010*"hobart" + 0.010*"rugbi" + 0.010*"secur" + 0.01
0*"say" + 0.009*"state" + 0.008*"domest"

Score: 0.21999986469745636
Topic: 0.023*"world" + 0.014*"final" + 0.013*"record" + 0.012*"break" + 0.011*"lose" + 0.011*"australian" + 0.011*"leagu" + 0.0
11*"test" + 0.010*"australia" + 0.010*"hill"

Score: 0.21999594569206238
Topic: 0.027*"south" + 0.024*"year" + 0.020*"interview" + 0.020*"north" + 0.019*"jail" + 0.018*"west" + 0.014*"island" + 0.013
*"australia" + 0.013*"victoria" + 0.010*"china"

Score: 0.020009687170386314
Topic: 0.018*"rural" + 0.018*"council" + 0.015*"fund" + 0.014*"plan" + 0.013*"health" + 0.012*"chang" + 0.011*"nation" + 0.010
*"price" + 0.010*"servic" + 0.009*"say"

Score: 0.020008400082588196
Topic: 0.024*"countri" + 0.021*"hour" + 0.020*"australian" + 0.019*"warn" + 0.016*"live" + 0.013*"indigen" + 0.011*"call" + 0.0
09*"victorian" + 0.009*"campaign" + 0.008*"show"

Score: 0.02000494673848152
Topic: 0.031*"queensland" + 0.029*"melbourn" + 0.018*"water" + 0.017*"claim" + 0.013*"hunter" + 0.012*"green" + 0.012*"resid" +
0.011*"darwin" + 0.010*"young" + 0.009*"plead"

Score: 0.020004209131002426
Topic: 0.052*"polic" + 0.020*"crash" + 0.019*"death" + 0.017*"sydney" + 0.016*"miss" + 0.016*"woman" + 0.015*"die" + 0.015*"cha
rg" + 0.014*"shoot" + 0.013*"arrest"
```

Figure 5

Our test document has the highest probability to be part of the topic that our model

```
for index, score in sorted(lda_model_tfidf[bow_corpus[4310]],
key=lambda tup: -1*tup[1]):
    print("\nScore: {}\t \nTopic: {}".format(score,
lda_model_tfidf.print_topic(index, 10)))
```

```
Score: 0.44014573097229004
Topic: 0.014*"south" + 0.009*"weather" + 0.009*"north" + 0.008*"west" + 0.008*"coast" + 0.008*"australia" + 0.006*"east" + 0.00
6*"queensland" + 0.006*"storm" + 0.005*"season"

Score: 0.3998423218727112
Topic: 0.023*"rural" + 0.018*"govern" + 0.013*"news" + 0.012*"podcast" + 0.008*"grandstand" + 0.008*"health" + 0.007*"budget" +
0.007*"busi" + 0.007*"nation" + 0.007*"fund"

Score: 0.02000250481069088
Topic: 0.006*"action" + 0.006*"violenc" + 0.006*"thursday" + 0.005*"domest" + 0.005*"cancer" + 0.005*"legal" + 0.005*"union" +
0.005*"breakfast" + 0.005*"school" + 0.004*"student"

Score: 0.020002111792564392
Topic: 0.008*"drum" + 0.007*"abbott" + 0.007*"farm" + 0.006*"dairi" + 0.006*"asylum" + 0.006*"tuesday" + 0.006*"water" + 0.006
*"labor" + 0.006*"say" + 0.005*"plan"

Score: 0.020001791417598724
Topic: 0.030*"countri" + 0.028*"hour" + 0.009*"sport" + 0.008*"septemb" + 0.008*"wednesday" + 0.007*"commiss" + 0.006*"royal" +
0.006*"updat" + 0.006*"station" + 0.005*"bendigo"

Score: 0.02000163123011589
Topic: 0.008*"octob" + 0.006*"search" + 0.006*"miss" + 0.006*"inquest" + 0.005*"stori" + 0.005*"jam" + 0.004*"john" + 0.004*"ha
rvest" + 0.004*"australia" + 0.004*"world"

Score: 0.020001478493213654
Topic: 0.008*"monday" + 0.008*"august" + 0.006*"babi" + 0.005*"shorten" + 0.005*"hobart" + 0.004*"victorian" + 0.004*"donald" +
0.004*"safe" + 0.004*"scott" + 0.004*"donat"
```

Figure 6

Our test document has the highest probability to be part of the topic that our model assigned, which is the accurate classification.

Testing model on unseen document

```
unseen_document = 'How a Pentagon deal became an identity crisis for
Google'
bow_vector = dictionary.doc2bow(preprocess(unseen_document))

for index, score in sorted(lda_model[bow_vector], key=lambda tup:
-1*tup[1]):
    print("Score: {}\t Topic: {}".format(score,
lda_model.print_topic(index, 5)))
```


```
Score: 0.016669215634465218     Topic: 0.018*"rural" + 0.018*"council" + 0.015*"fund" + 0.014*"plan" + 0.013*"health"
Score: 0.016668815165758133     Topic: 0.024*"countri" + 0.021*"hour" + 0.020*"australian" + 0.019*"warn" + 0.016*"live"
Score: 0.01666823774576187      Topic: 0.031*"queensland" + 0.029*"melbourn" + 0.018*"water" + 0.017*"claim" + 0.013*"hunter"
Score: 0.016666674986481667     Topic: 0.032*"court" + 0.022*"face" + 0.020*"charg" + 0.020*"home" + 0.018*"tasmania"
Score: 0.01666666753590107      Topic: 0.023*"world" + 0.014*"final" + 0.013*"record" + 0.012*"break" + 0.011*"lose"
```

Figure 7

Source code can be found on Github. I look forward to hearing any feedback or questions.

Reference:

Udacity — NLP