7. Topic Modeling

March 30, 2022

[1]:
```
# Topic Modeling by Latent Dirichlet Alloction (LDA) algorithm
# LDA is a probabilistic topic model that assumes documents are a mixture of␣
↪topics and that
# each word in the document is attributable to the document's topics.
# Other algorithm for topic modeling are Non-negative Matrix Factorization (NMF)
```

[2]:
```
#!pip install gensim
```

[3]:
```
import numpy as np
import pandas as pd
import nltk
import re
import os
import string
from sklearn import feature_extraction
#import three lists: titles, links and wikipedia synopses
titles = open('title_list.txt').read().split('\n')
#ensures that only the first 100 are read in
titles = titles[:100]
synopsis = open('synopsis_list.txt').read().split('\n BREAKS HERE')
synopsis = synopsis[:100]
```

[4]:
```
# load nltk's English stopwords as variable called 'stopwords'
stopwords = nltk.corpus.stopwords.words('english')
# load nltk's SnowballStemmer as variabled 'stemmer'
from nltk.stem.snowball import SnowballStemmer
stemmer = SnowballStemmer("english")
```

[5]:
```
#strip any proper names from a text
def strip_proppers(text):
    # first tokenize by sentence, then by word
    tokens = [word for sent in nltk.sent_tokenize(text) \
              for word in nltk.word_tokenize(sent) if word.islower()]
    return "".join([" "+i if not i.startswith("'") \
    and i not in string.punctuation else i for i in tokens]).strip()

def tokenize_and_stem(text):
    # first tokenize by sentence, then by word
    tokens = [word for sent in nltk.sent_tokenize(text) \
              for word in nltk.word_tokenize(sent)]
    filtered_tokens = []
    # filter out any tokens not containing letters
    for token in tokens:
        if re.search('[a-zA-Z]', token):
            filtered_tokens.append(token)
    stems = [stemmer.stem(t) for t in filtered_tokens]
    return stems
```

[6]:
```
#Latent Dirichlet Allocation implementation with Gensim
from gensim import corpora, models, similarities
#remove proper names
preprocess = [strip_proppers(doc) for doc in synopsis]
tokenized_text = [tokenize_and_stem(text) for text in preprocess]
texts = [[word for word in text if word not in stopwords] \
         for text in tokenized_text]
print(len(texts[0]))

dictionary = corpora.Dictionary(texts)
dictionary.filter_extremes(no_below=1, no_above=0.8)
corpus = [dictionary.doc2bow(text) for text in texts]
len(corpus)
```

```
1234
```

[6]:
```
100
```

[7]:
```
lda = models.LdaModel(corpus, num_topics=5, id2word=dictionary, update_every=5,␣
 ↪chunksize=10000, passes=100)
print(lda[corpus[0]])
```

```
[(1, 0.9993052)]
```

[8]:
```
topics = lda.print_topics(5, num_words=20)
topics
```

[8]:
```
[(0,
  '0.008*"shark" + 0.007*"ship" + 0.005*"water" + 0.005*"tell" + 0.004*"take" +
0.004*"one" + 0.004*"find" + 0.004*"destroy" + 0.003*"escap" + 0.003*"order" +
0.003*"two" + 0.003*"man" + 0.003*"station" + 0.003*"see" + 0.003*"droid" +
0.003*"boat" + 0.003*"befor" + 0.003*"base" + 0.003*"back" + 0.003*"troop"'),
 (1,
  '0.012*"tell" + 0.006*"get" + 0.006*"take" + 0.006*"n\'t" + 0.005*"leav" +
0.005*"ask" + 0.005*"man" + 0.005*"see" + 0.005*"say" + 0.005*"back" +
0.004*"one" + 0.004*"kill" + 0.004*"go" + 0.004*"men" + 0.004*"tri" +
0.004*"find" + 0.004*"come" + 0.004*"meet" + 0.004*"goe" + 0.004*"call"'),
 (2,
  '0.008*"car" + 0.006*"get" + 0.006*"take" + 0.006*"go" + 0.005*"back" +
0.005*"tell" + 0.005*"talk" + 0.005*"one" + 0.004*"friend" + 0.004*"come" +
0.004*"n\'t" + 0.004*"explain" + 0.003*"befor" + 0.003*"say" + 0.003*"pod" +
0.003*"find" + 0.003*"onli" + 0.003*"see" + 0.003*"claim" + 0.003*"girl"'),
 (3,
  '0.006*"famili" + 0.005*"find" + 0.005*"take" + 0.005*"back" + 0.004*"leav" +
0.004*"return" + 0.004*"two" + 0.004*"see" + 0.004*"one" + 0.004*"day" +
0.003*"man" + 0.003*"kill" + 0.003*"work" + 0.003*"home" + 0.003*"befor" +
0.003*"tell" + 0.003*"go" + 0.003*"get" + 0.003*"friend" + 0.003*"onli"'),
 (4,
  '0.005*"mountain" + 0.005*"son" + 0.004*"appear" + 0.004*"one" + 0.004*"peopl"
+ 0.004*"befor" + 0.004*"two" + 0.004*"gladiat" + 0.004*"fli" + 0.004*"onli" +
0.003*"strang" + 0.003*"find" + 0.003*"light" + 0.003*"sever" + 0.003*"juri" +
0.003*"guilti" + 0.003*"juror" + 0.003*"number" + 0.003*"see" + 0.002*"die"')]
```

[9]:
```
topics_matrix = lda.show_topics(formatted=False, num_words=20)
topics_matrix
```

[9]:
```
[(0,
  [('shark', 0.008267318),
   ('ship', 0.006609732),
   ('water', 0.0054516946),
   ('tell', 0.005251137),
   ('take', 0.004272602),
   ('one', 0.0042589493),
   ('find', 0.0039768717),
   ('destroy', 0.0036508336),
   ('escap', 0.0033190951),
   ('order', 0.0031625612),
   ('two', 0.0031478007),
   ('man', 0.0030502705),
   ('station', 0.0030317947),
   ('see', 0.0029843543),
   ('droid', 0.0029716925),
   ('boat', 0.002968863),
   ('befor', 0.0028547016),
   ('base', 0.0028245468),
   ('back', 0.0027762323),
   ('troop', 0.0027191208)]),
 (1,
  [('tell', 0.01185876),
   ('get', 0.0063111256),
   ('take', 0.005928878),
   ("n't", 0.005686244),
   ('leav', 0.0054641385),
  ('ask', 0.005463069),
  ('man', 0.00531964),
  ('see', 0.0052484614),
  ('say', 0.0051322603),
  ('back', 0.0045040115),
  ('one', 0.0044282833),
  ('kill', 0.004315245),
  ('go', 0.0043122508),
  ('men', 0.004250982),
  ('tri', 0.0042365068),
  ('find', 0.004035435),
  ('come', 0.0039638523),
  ('meet', 0.003807375),
  ('goe', 0.0037176204),
  ('call', 0.0036115733)]),
(2,
 [('car', 0.008375065),
  ('get', 0.0062691034),
  ('take', 0.0061161043),
  ('go', 0.0055828257),
  ('back', 0.0053755646),
  ('tell', 0.0049896073),
  ('talk', 0.0047794282),
  ('one', 0.004508482),
  ('friend', 0.004277147),
  ('come', 0.0042443713),
  ("n't", 0.0041522267),
  ('explain', 0.003744715),
  ('befor', 0.0034794612),
  ('say', 0.0033705062),
  ('pod', 0.0032895892),
  ('find', 0.003183639),
  ('onli', 0.0031324232),
  ('see', 0.0030691344),
  ('claim', 0.0029722096),
  ('girl', 0.0029505352)]),
(3,
 [('famili', 0.0055331667),
  ('find', 0.0054167304),
  ('take', 0.005032659),
  ('back', 0.004579791),
  ('leav', 0.0044060466),
  ('return', 0.004265712),
  ('two', 0.004193646),
  ('see', 0.0040978766),
  ('one', 0.0036854965),
  ('day', 0.0036183822),
   ('man', 0.0034487178),
   ('kill', 0.0034404572),
   ('work', 0.0034327821),
   ('home', 0.003366991),
   ('befor', 0.0033624668),
   ('tell', 0.0033599187),
   ('go', 0.003270913),
   ('get', 0.0032305643),
   ('friend', 0.0031701121),
   ('onli', 0.0031324653)]),
 (4,
  [('mountain', 0.0047614262),
   ('son', 0.00464404),
   ('appear', 0.0044572637),
   ('one', 0.0042808964),
   ('peopl', 0.0042396383),
   ('befor', 0.0039281086),
   ('two', 0.0038373915),
   ('gladiat', 0.0037793368),
   ('fli', 0.0036354794),
   ('onli', 0.0035699152),
   ('strang', 0.003489191),
   ('find', 0.0029251496),
   ('light', 0.0027839895),
   ('sever', 0.0027140547),
   ('juri', 0.0026686343),
   ('guilti', 0.0026682864),
   ('juror', 0.0026677295),
   ('number', 0.0025950358),
   ('see', 0.0025866146),
   ('die', 0.0024017585)])]
```

[10]:
```
for i,topic in enumerate(lda.print_topics(num_topics=50, num_words=10)):
    words = topic[1].split("+")
    print(words,"\n")
```

```
['0.008*"shark" ', ' 0.007*"ship" ', ' 0.005*"water" ', ' 0.005*"tell" ', '
0.004*"take" ', ' 0.004*"one" ', ' 0.004*"find" ', ' 0.004*"destroy" ', '
0.003*"escap" ', ' 0.003*"order"']

['0.012*"tell" ', ' 0.006*"get" ', ' 0.006*"take" ', ' 0.006*"n\'t" ', '
0.005*"leav" ', ' 0.005*"ask" ', ' 0.005*"man" ', ' 0.005*"see" ', ' 0.005*"say"
', ' 0.005*"back"']

['0.008*"car" ', ' 0.006*"get" ', ' 0.006*"take" ', ' 0.006*"go" ', '
0.005*"back" ', ' 0.005*"tell" ', ' 0.005*"talk" ', ' 0.005*"one" ', '
0.004*"friend" ', ' 0.004*"come"']
```

```
['0.006*"famili" ', ' 0.005*"find" ', ' 0.005*"take" ', ' 0.005*"back" ', '
0.004*"leav" ', ' 0.004*"return" ', ' 0.004*"two" ', ' 0.004*"see" ', '
0.004*"one" ', ' 0.004*"day"']

['0.005*"mountain" ', ' 0.005*"son" ', ' 0.004*"appear" ', ' 0.004*"one" ', '
0.004*"peopl" ', ' 0.004*"befor" ', ' 0.004*"two" ', ' 0.004*"gladiat" ', '
0.004*"fli" ', ' 0.004*"onli"']
```

```
[ ]:
```