NLP Assignment #6

Luke Schwenke

The data is a collection of tweets and a collection of news articles about one particular company. Use appropriate topic modeling technique to identify top N most important topics. To get quality results apply appropriate text cleaning methods.

1. Present top N most important topics in the news articles and tweets. For news articles, consider how to effectively combine information from the title and text of news article
2. Select N to identify relevant topics, but minimize duplication
3. Explain how you selected N

In [ ]:
```
#!pip install --upgrade pip wheel
```

In [ ]:
```
import os
import time
import math
import re
from pprint import pprint
from textblob import TextBlob
#import pandas as pd
import numpy as np

import nltk as nltk
from nltk.corpus import stopwords
# from nltk.stem.wordnet import WordNetLemmatizer
import spacy
import multiprocessing
import string

import gensim
from gensim import corpora, models
from gensim.models.ldamulticore import LdaMulticore
from gensim.utils import simple_preprocess
from gensim.models import CoherenceModel

# import pyLDAvis.gensim
import pyLDAvis
import pyLDAvis.gensim_models as gensimvis
pyLDAvis.enable_notebook()

pd.set_option('display.max_rows', 100)
pd.set_option('display.max_columns', None)
pd.set_option('display.max_colwidth', 500)
```

In [ ]:
```
#!pip install pandas==1.5.3
```

In [ ]:
```
import warnings
warnings.filterwarnings("ignore")
```

In [ ]:
```
import multiprocessing
num_processors = multiprocessing.cpu_count()
print(f'Available CPUs: {num_processors}')
```

```
Available CPUs: 8
```

In [ ]:
```
from pandarallel import pandarallel
pandarallel.initialize(nb_workers=num_processors-1, use_memory_fs=False, progress_bar=True)
```

```
INFO: Pandarallel will run on 7 workers.
INFO: Pandarallel will use standard multiprocessing data transfer (pipe) to transfer data between the main
process and workers.
```

In [ ]:
```
workers = num_processors-1
print(f'Using {workers} workers')
```

```
Using 7 workers
```

In [ ]:
```
start_time = time.time()
```

```
def tic():
    global start_time
    start_time = time.time()

def tac():
    t_sec = round(time.time() - start_time)
    (t_min, t_sec) = divmod(t_sec,60)
    (t_hour,t_min) = divmod(t_min,60)
    print(f'Execution time to calculate for topic {k}: {t_hour}hour:{t_min}min:{t_sec}sec'.format(t_hour,t
```

Read news data

In [ ]:
```
news_path = 'https://storage.googleapis.com/msca-bdp-data-open/news/nlp_a_6_news.json'
news_df = pd.read_json(news_path, orient='records', lines=True)

print(f'Sample contains {news_df.shape[0]:,.0f} news articles')
news_df.head(2)
```

```
Sample contains 9,962 news articles
```

Out[ ]:

	url	date	language	title	
0	http://oaklandnewsnow.com/breaking-bts-announces-las-vegas-us-concert-date-in-2022/	2022-02-24	en	BREAKING: BTS Announces LAS VEGAS, US Concert Date in 2022! | Oakland News Now - Oakland News, SF Bay Area, East Bay, California, World	BREAKING: BTS Announces in 2022! | Oakland News Now East Bay, California, W Disabled! To see this page a enable your Javascript!BF VEGAS, US Concert Date i Oakland News, SF Bay Area, I to contentMenuSearch fc Oakland News, SF
1	http://www.newsdzezimbabwe.co.uk/2022/04/mai-tt-weds.html	2022-04-09	en	MAI TT WEDS newsdzeZimbabweNewsdzeZimbabwe	MAI TT WEDS newsdzeZim sidebarHomeAboutContactAd NewsHomeNewsBusinessI 2022MAI TT ' NewsdzeZimbabwe 0 @NyamayaroArrc pic.twitter.com/MsrhcFXUJj– 2022 Posted in: Share to T PostHome0com

Read Tweets data

In [ ]:
```
tweets_path = 'https://storage.googleapis.com/msca-bdp-data-open/tweets/nlp_a_6_tweets.json'
tweets_df = pd.read_json(tweets_path, orient='records', lines=True)
print(f'Sample contains {tweets_df.shape[0]:,.0f} tweets')
tweets_df.head(2)
```

```
Sample contains 9,941 tweets
```

Out[ ]:

	id	lang	date	name	retweeted	text
0	1484553027222741001	en	2022-01-21	Dylan Green	RT	*Microsoft has entered the chat* https://t.co/Uz3pZrk6B3
1	1505486305102557184	en	2022-03-20	Rahim Rajwani		"I actually use an @Android phone. Some #Android manufacturers pre-install @Microsoft software in a way that makes it easy for me. They're more flexible about how the software connects up with the OS. So that's what I ended up getting used to."\nhttps://t.co/C0VjfS9PUO

Data Cleaning & N-Grams & Lemmatizing

In [ ]:
```
nltk.download('stopwords')
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))

nltk.download('words')
from nltk.corpus import words

english_words = set(words.words())
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/lmschwenke/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package words to
[nltk_data]     /Users/lmschwenke/nltk_data...
[nltk_data]   Package words is already up-to-date!
```

In [ ]:

```
import re

def clean_text(text):
    # Remove mentions
    text = re.sub(r'@[A-Za-z0-9_]+', '', text)
    # Remove hashtags (but keep the text after #)
    text = re.sub(r'#', '', text)
    # Remove RT (retweet symbol)
    text = re.sub(r'RT[\s]+', '', text)
    # Remove hyperlinks
    text = re.sub(r'https?:\/\/\S+', '', text)
    # Remove newline characters
    text = re.sub(r'\n', ' ', text)
    # Remove carriage return characters
    text = re.sub(r'\r', '', text)
    # Remove "&amp;"
    text = re.sub(r'&amp;', '', text)
    # Remove other special characters and numbers
    text = re.sub(r'[^A-Za-z\s]', '', text)
    # Convert multiple spaces to a single space
    text = re.sub(r'\s+', ' ', text)
    # Optionally, convert to lowercase
    # text = text.lower()
    # Remove stopwords
    text = ' '.join([word for word in text.split() if word not in stop_words])
    # Remove non-English words
    text = ' '.join([word for word in text.split() if word.lower() in english_words])

    return text.strip()
```

In [ ]:

```
tweets_df['tweets_clean'] = tweets_df['text'].parallel_apply(clean_text)
news_df['text_clean'] = news_df['text'].parallel_apply(clean_text)
news_df['title_clean'] = news_df['title'].parallel_apply(clean_text)
```

```
VBox(children=(HBox(children=(IntProgress(value=0, description='0.00%', max=1421), Label(value='0 / 142
1'))), …
VBox(children=(HBox(children=(IntProgress(value=0, description='0.00%', max=1424), Label(value='0 / 142
4'))), …
VBox(children=(HBox(children=(IntProgress(value=0, description='0.00%', max=1424), Label(value='0 / 142
4'))), …
```

In [ ]:

```
def sent_to_words(sentences):
    for sentence in sentences:
        yield(gensim.utils.simple_preprocess(str(sentence), deacc=False))  # deacc=True removes punctuatio
```

In [ ]:

```
def make_bi_and_tri_grams(texts):

    bigram = gensim.models.Phrases(texts, min_count=1, threshold=1)
    bigram_mod = gensim.models.phrases.Phraser(bigram)

    trigram = gensim.models.Phrases(bigram[texts], threshold=1)
    trigram_mod = gensim.models.phrases.Phraser(trigram)

    return [trigram_mod[bigram_mod[doc]] for doc in texts]
```

In [ ]:

```
nlp = spacy.load("en_core_web_lg", disable=['parser', 'ner'])
```

In [ ]:

```
def lemmatization(texts, allowed_postags=['NOUN', 'ADJ', 'VERB', 'ADV']):
    """https://spacy.io/api/annotation"""
    texts_out = []
    for sent in texts:
        doc = nlp(" ".join(sent))
        texts_out.append([token.lemma_ for token in doc if token.pos_ in allowed_postags])
    return texts_out
```

In [ ]:

```
def prepare_data(column):

    # Tokenize text into words (other punctuation and elements have been cleaned in earlier function)
    data_list = column.tolist()
```

```
    data_tokens = list(sent_to_words(data_list))

    # Make n-grams
    data_words_trigrams = make_bi_and_tri_grams(data_tokens)

    # Lemmatize text keeping only noun, adj, vb, adv
    data_lemmatized = lemmatization(data_words_trigrams, allowed_postags=['NOUN', 'ADJ', 'VERB', 'ADV'])

    return data_lemmatized
```

News Articles - Topic Modeling

In [ ]:
```
# Combine the cleaned article titles with the text body
news_df['title_and_text'] = news_df['title_clean'] + ' ' + news_df['text_clean']
```

In [ ]:
```
news_df[['title_clean', 'text_clean', 'title_and_text']].head(1)
```

Out[ ]:

	title_clean	text_clean	title_and_text
0	BREAKING LAS US Concert Date News Now News Bay Area East Bay World	BREAKING LAS US Concert Date News Now News Bay Area East Bay Disabled To see page meant appear please enable LAS US Concert Date News Now News Bay Area East Bay News Now News Bay Area East Bay News Now News News Aggregator Home About News Lake News The Alley Cat Grand Lake Theater Law Channel Page Comic Con Las Draft Draft Page The Eagle Offense City Council Privacy BREAKING LAS Concert Entertainment News Concert Date LAS US Concert Date News Now video made channel upper left hand corner ori...	BREAKING LAS US Concert Date News Now News Bay Area East Bay World BREAKING LAS US Concert Date News Now News Bay Area East Bay Disabled To see page meant appear please enable LAS US Concert Date News Now News Bay Area East Bay News Now News Bay Area East Bay News Now News News Aggregator Home About News Lake News The Alley Cat Grand Lake Theater Law Channel Page Comic Con Las Draft Draft Page The Eagle Offense City Council Privacy BREAKING LAS Concert Entertainment News Concert Date LAS US ...

In [ ]:
```
lemmatized_titles_and_articles = prepare_data(news_df['title_and_text'])
```

In [ ]:
```
dictionary = corpora.Dictionary(lemmatized_titles_and_articles)

# Converting list of documents (corpus) into Document Term Matrix using dictionary prepared above.
doc_term_matrix = [dictionary.doc2bow(doc) for doc in lemmatized_titles_and_articles]
```

In [ ]:
```
def compute_coherence_values(corpus, dictionary, k, a, b, texts_lemmatized):

    lda_model = LdaMulticore(corpus=doc_term_matrix,
                   id2word=dictionary,
                   num_topics=k,
                   random_state=100,
                   passes=10,
                   alpha=a,
                   eta=b,
                   workers=workers)

    coherence_model_lda = CoherenceModel(model=lda_model,
                                 texts=texts_lemmatized,
                                 dictionary=dictionary,
                                 coherence='c_v')

    return coherence_model_lda.get_coherence()
```

In [ ]:
```
%%time

grid = {}
grid['Validation_Set'] = {}
# Topics range
min_topics = 2
max_topics = 8
step_size = 1
topics_range = range(min_topics, max_topics+1, step_size)

# Validation sets
num_of_docs = len(doc_term_matrix)
corpus_sets = [# gensim.utils.ClippedCorpus(doc_term_matrix, num_of_docs*0.25),
               # gensim.utils.ClippedCorpus(doc_term_matrix, num_of_docs*0.5),
#                gensim.utils.ClippedCorpus(doc_term_matrix, num_of_docs*0.75),
               doc_term_matrix]
```

```
corpus_title = ['100% Corpus']
model_results = {
                 'Topics': [],
                 'Coherence': []
                }

itr = 0
itr_total = len(topics_range)*len(corpus_title) #len(beta)*len(alpha)*
print(f'LDA will execute {itr_total} iterations')

# iterate through hyperparameters
for i in range(len(corpus_sets)):
    # iterate through number of topics
    for k in topics_range:
        tic()
        itr += 1
        cv = compute_coherence_values(corpus=corpus_sets[i],
                                      dictionary=dictionary,
                                      k=k,
                                      texts_lemmatized=lemmatized_titles_and_articles,
                                      a=0.5,
                                      b=0.5) # Updated
        # Save the model results
        model_results['Topics'].append(k)
                #model_results['Alpha'].append(a)
                #model_results['Beta'].append(b)
        model_results['Coherence'].append(cv)
        pct_completed = round((itr / itr_total * 100),1)
#                 print(f'Completed Percent: {pct_completed}%, Corpus: {corpus_title[i]}, Topics: {k}, Alp
        print(f'ARTICLES - Completed model based on {k} LDA topics. Finished {pct_completed}% of LDA runs'
        tac()

lda_tuning_articles = pd.DataFrame(model_results)
```

```
LDA will execute 7 iterations
ARTICLES - Completed model based on 2 LDA topics. Finished 14.3% of LDA runs
Execution time to calculate for topic 2: 0hour:0min:27sec
ARTICLES - Completed model based on 3 LDA topics. Finished 28.6% of LDA runs
Execution time to calculate for topic 3: 0hour:0min:59sec
ARTICLES - Completed model based on 4 LDA topics. Finished 42.9% of LDA runs
Execution time to calculate for topic 4: 0hour:2min:47sec
ARTICLES - Completed model based on 5 LDA topics. Finished 57.1% of LDA runs
Execution time to calculate for topic 5: 0hour:4min:15sec
ARTICLES - Completed model based on 6 LDA topics. Finished 71.4% of LDA runs
Execution time to calculate for topic 6: 0hour:20min:31sec
ARTICLES - Completed model based on 7 LDA topics. Finished 85.7% of LDA runs
Execution time to calculate for topic 7: 0hour:34min:35sec
ARTICLES - Completed model based on 8 LDA topics. Finished 100.0% of LDA runs
Execution time to calculate for topic 8: 0hour:7min:12sec
CPU times: user 7min 42s, sys: 2min 31s, total: 10min 14s
Wall time: 1h 10min 46s
```

In [ ]:
```
lda_tuning_articles
```

Out[ ]:

	Topics	Coherence
0	2	0.409476
1	3	0.401805
2	4	0.375256
3	5	0.391131
4	6	0.390823
5	7	0.396331
6	8	0.377064

Articles & Titles: After iterating over varying levels of *n* topics with the LDA topic model, the highest coherence value indicated was 2 topics. For this reason I will continue with *n=2* for the model and analysis

In [ ]:
```
%%time

lda_model_titles_and_articles_2_topics = LdaMulticore(corpus=doc_term_matrix,
```

```
                                                             id2word=dictionary,
                                                             num_topics=2,
                                                             random_state=456,
                                                             passes=10,
                                                             eta='auto',
                                                             workers=workers)
```

```
CPU times: user 12 s, sys: 1.24 s, total: 13.3 s
Wall time: 22.3 s
```

In [ ]:

```
# Print the keywords in the 2 topics
pprint(lda_model_titles_and_articles_2_topics.print_topics())
```

```
[(0,
  '0.007*"new" + 0.005*"say" + 0.004*"day" + 0.003*"also" + 0.003*"get" + '
  '0.003*"make" + 0.003*"use" + 0.003*"go" + 0.003*"see" + 0.003*"work"'),
 (1,
  '0.005*"official_music_video" + 0.004*"new" + 0.003*"news" + 0.002*"stock" + '
  '0.002*"say" + 0.002*"day" + 0.002*"get" + 0.002*"company" + 0.002*"video" + '
  '0.001*"make"')]
```

In [ ]:

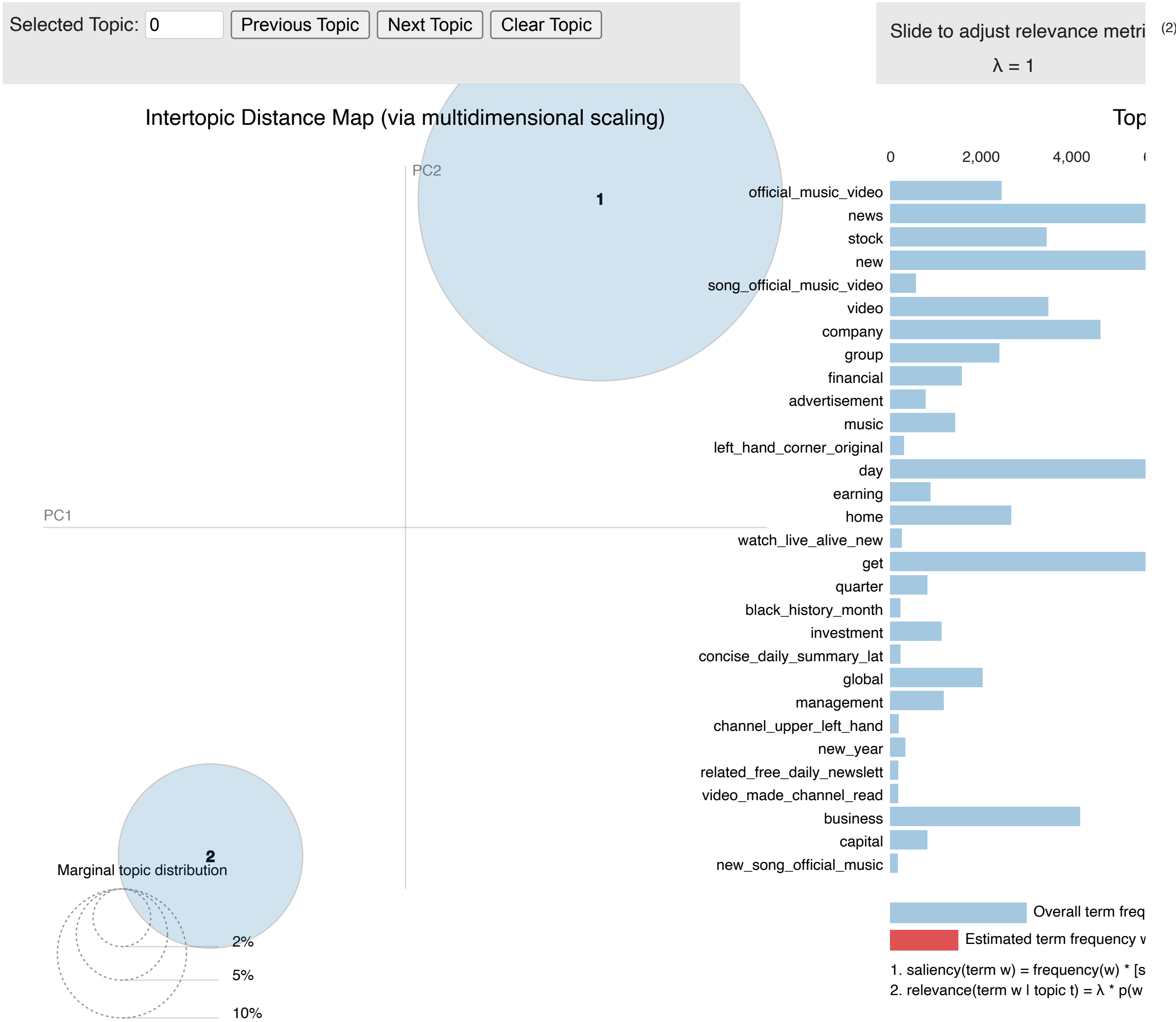
```
coherence_model_lda = CoherenceModel(model=lda_model_titles_and_articles_2_topics,
                                     texts=lemmatized_titles_and_articles,
                                     dictionary=dictionary,
                                     coherence='c_v')

coherence_lda = coherence_model_lda.get_coherence()
print('\nCoherence Score for n=2 for the Titles and Articles: ', coherence_lda)
```

```
Coherence Score for n=2 for the Titles and Articles:  0.41954264932134977
```

In [ ]:

```
lda_display = gensimvis.prepare(lda_model_titles_and_articles_2_topics,
                                doc_term_matrix,
                                dictionary,
                                sort_topics=False,
                                mds='mmds')
pyLDAvis.display(lda_display)
```

Out[ ]:



Tweets - Topic Modeling

In [ ]:
```
tweets_df['tweets_clean'][1]
```

Out[ ]:
```
'I actually use phone Some Android preinstall way easy Theyre flexible OS So thats I ended getting used'
```

In [ ]:
```
lemmatized_tweets = prepare_data(tweets_df['tweets_clean'])
```

In [ ]:
```
dictionary = corpora.Dictionary(lemmatized_tweets)
# Converting list of documents (corpus) into Document Term Matrix using dictionary prepared above.
doc_term_matrix = [dictionary.doc2bow(doc) for doc in lemmatized_tweets]
```

In [ ]:
```
%%time

grid = {}
grid['Validation_Set'] = {}
# Topics range
min_topics = 2
max_topics = 8
step_size = 1
topics_range = range(min_topics, max_topics+1, step_size)
```

```
# Validation sets
num_of_docs = len(doc_term_matrix)
corpus_sets = [# gensim.utils.ClippedCorpus(doc_term_matrix, num_of_docs*0.25),
               # gensim.utils.ClippedCorpus(doc_term_matrix, num_of_docs*0.5),
#                gensim.utils.ClippedCorpus(doc_term_matrix, num_of_docs*0.75),
               doc_term_matrix]

corpus_title = ['100% Corpus']
model_results = {
                 'Topics': [],
                 'Coherence': []
                }

itr = 0
itr_total = len(topics_range)*len(corpus_title) #len(beta)*len(alpha)*
print(f'LDA will execute {itr_total} iterations')

# iterate through hyperparameters
for i in range(len(corpus_sets)):
    # iterate through number of topics
    for k in topics_range:
        tic()
        itr += 1
        cv = compute_coherence_values(corpus=corpus_sets[i],
                                      dictionary=dictionary,
                                      k=k,
                                      texts_lemmatized=lemmatized_tweets,
                                      a=0.5,
                                      b=0.5) # Updated
        # Save the model results
        model_results['Topics'].append(k)
                #model_results['Alpha'].append(a)
                #model_results['Beta'].append(b)
        model_results['Coherence'].append(cv)
        pct_completed = round((itr / itr_total * 100),1)
        # print(f'Completed Percent: {pct_completed}%, Corpus: {corpus_title[i]}, Topics: {k}, Alpha: {a},
        print(f'ARTICLES - Completed model based on {k} LDA topics. Finished {pct_completed}% of LDA runs'
        tac()

lda_tuning_tweets= pd.DataFrame(model_results)
```

```
LDA will execute 7 iterations
ARTICLES - Completed model based on 2 LDA topics. Finished 14.3% of LDA runs
Execution time to calculate for topic 2: 0hour:0min:10sec
ARTICLES - Completed model based on 3 LDA topics. Finished 28.6% of LDA runs
Execution time to calculate for topic 3: 0hour:0min:10sec
ARTICLES - Completed model based on 4 LDA topics. Finished 42.9% of LDA runs
Execution time to calculate for topic 4: 0hour:0min:10sec
ARTICLES - Completed model based on 5 LDA topics. Finished 57.1% of LDA runs
Execution time to calculate for topic 5: 0hour:0min:10sec
ARTICLES - Completed model based on 6 LDA topics. Finished 71.4% of LDA runs
Execution time to calculate for topic 6: 0hour:0min:10sec
ARTICLES - Completed model based on 7 LDA topics. Finished 85.7% of LDA runs
Execution time to calculate for topic 7: 0hour:0min:10sec
ARTICLES - Completed model based on 8 LDA topics. Finished 100.0% of LDA runs
Execution time to calculate for topic 8: 0hour:0min:10sec
CPU times: user 15.6 s, sys: 2.18 s, total: 17.8 s
Wall time: 1min 11s
```

In [ ]:
```
lda_tuning_tweets
```

Out[ ]:

	Topics	Coherence
0	2	0.405639
1	3	0.397374
2	4	0.398100
3	5	0.381171
4	6	0.364934
5	7	0.403216
6	8	0.416361

Tweets: After iterating over varying levels of *n* topics with the LDA topic model, the highest coherence value indicated was 8 topics. For this reason I will continue with *n=8* for the model and analysis

In [ ]:
```
%%time

lda_model_tweets_8_topics = LdaMulticore(corpus=doc_term_matrix,
                                         id2word=dictionary,
                                         num_topics=8,
                                         random_state=456,
                                         passes=10,
                                         eta='auto',
                                         workers=workers)
```

```
CPU times: user 1.56 s, sys: 184 ms, total: 1.74 s
Wall time: 5.98 s
```

In [ ]:
```
# Print the keywords in the 2 topics
pprint(lda_model_tweets_8_topics.print_topics())
```

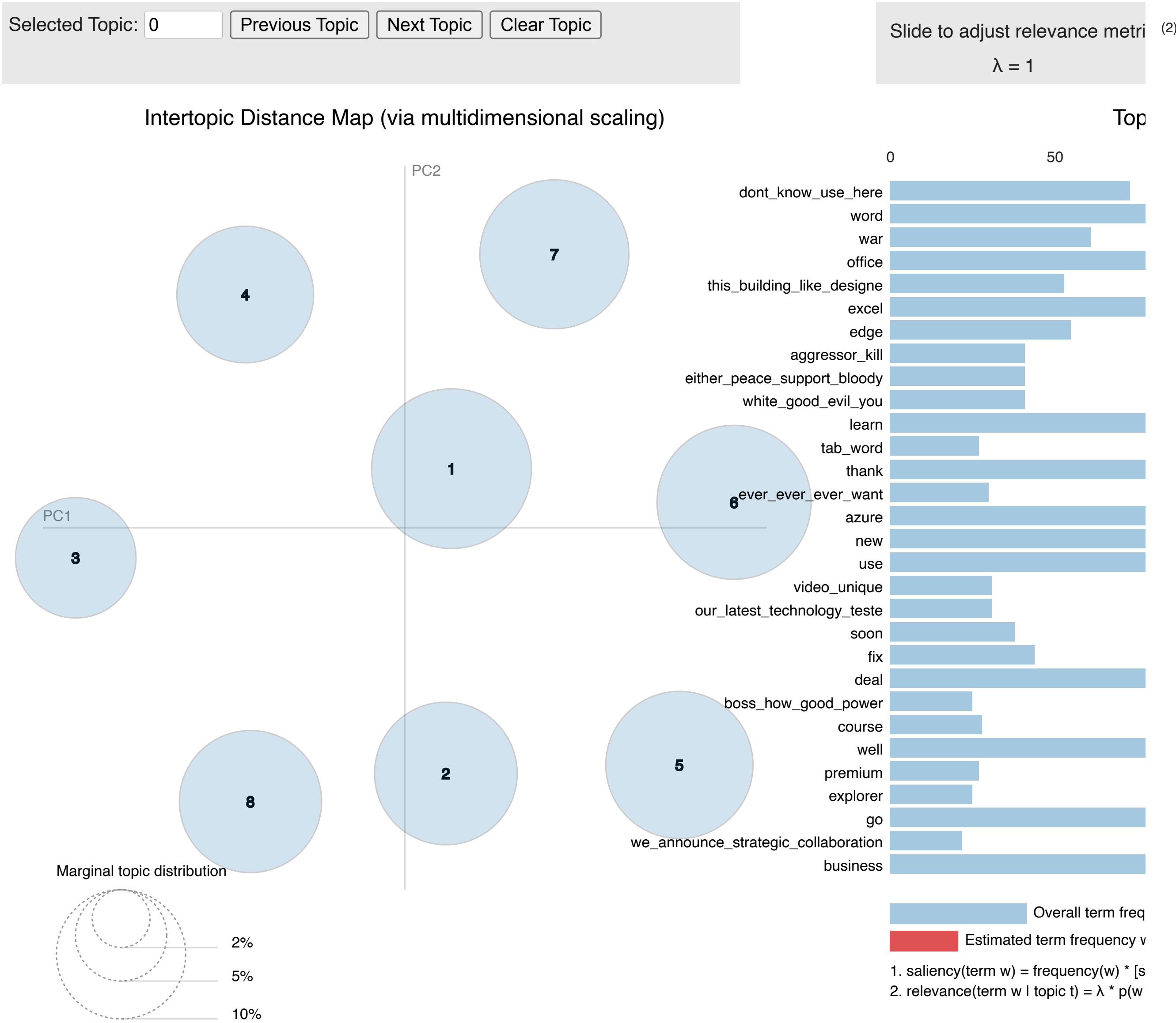
```
[(0,
  '0.010*"office" + 0.010*"get" + 0.009*"work" + 0.006*"see" + 0.005*"s" + '
  '0.005*"business" + 0.005*"never" + 0.005*"excel" + 0.004*"build" + '
  '0.004*"tech"'),
 (1,
  '0.012*"dont_know_use_here" + 0.009*"azure" + 0.007*"deal" + 0.007*"time" + '
  '0.006*"make" + 0.006*"well" + 0.006*"new" + 0.006*"get" + 0.005*"learn" + '
  '0.005*"use"'),
 (2,
  '0.006*"tab_word" + 0.006*"soon" + 0.004*"azure" + 0.004*"team" + '
  '0.004*"free" + 0.004*"make" + 0.004*"learn" + 0.004*"excel_smoke" + '
  '0.004*"whoever_designe" + 0.004*"little_wee"'),
 (3,
  '0.012*"learn" + 0.005*"work" + 0.005*"ever_ever_ever_want" + '
  '0.005*"company" + 0.005*"go" + 0.005*"say" + 0.004*"course" + 0.004*"make" '
  '+ 0.004*"watch" + 0.004*"see"'),
 (4,
  '0.007*"buy" + 0.007*"go" + 0.006*"word" + 0.006*"business" + 0.006*"make" + '
  '0.005*"think" + 0.005*"take" + 0.005*"say" + 0.004*"office" + 0.004*"use"'),
 (5,
  '0.015*"word" + 0.008*"war" + 0.008*"work" + 0.008*"go" + 0.008*"get" + '
  '0.008*"this_building_like_designe" + 0.007*"new" + 0.007*"apple" + '
  '0.007*"thank" + 0.006*"aggressor_kill"'),
 (6,
  '0.010*"excel" + 0.010*"new" + 0.006*"game" + 0.006*"make" + 0.005*"good" + '
  '0.005*"get" + 0.005*"video_unique" + 0.005*"our_latest_technology_teste" + '
  '0.005*"datum" + 0.004*"look"'),
 (7,
  '0.012*"use" + 0.008*"get" + 0.008*"edge" + 0.005*"fix" + 0.005*"today" + '
  '0.005*"come" + 0.004*"explorer" + 0.004*"thank" + '
  '0.004*"we_announce_strategic_collaboration" + 0.004*"top"')]
```

In [ ]:
```
coherence_model_lda = CoherenceModel(model=lda_model_tweets_8_topics,
                                     texts=lemmatized_tweets,
                                     dictionary=dictionary,
                                     coherence='c_v')

coherence_lda = coherence_model_lda.get_coherence()
print('\nCoherence Score for n=8 for the Tweets: ', coherence_lda)
```

```
Coherence Score for n=8 for the Tweets:  0.45664423263293846
```

In [ ]:
```
lda_display = gensimvis.prepare(lda_model_tweets_8_topics,
                                doc_term_matrix,
                                dictionary,
                                sort_topics=False,
                                mds='mmds')
pyLDAvis.display(lda_display)
```

Out[ ]:



Conclusions

Both articles and tweets were cleaned, lemmatized, and had n-grams created before the dictionary corpus and document term matrix were made. Due to resource constraints, the grid search functions only iterated over varying values of *n* topics; more compute resoure could allow further tuning of LDA alpha and beta values as well as more topics (*n*). After running both LDA models for the corresonding ideal topics, the diagram indicated a distinct spread between the topic clusters (no overlap), meaning the topics are distinct from each other for both Tweets and Articles.

Articles:

The article dataset was created by appending the article title to the front of the article text body. The grid search indicated number of topics set to *n=2* was most appropriate. The coherence value for the corresponding LDA model was 0.4195. Examining the diagram for these 2 news article topics indiates the following:

1. Topic #1 (finance) - companies, markets, busines
2. Topic #2 (music and videos) - videos, music, music videos

Tweets:

The grid search indicated number of topics set to *n=8* was most appropriate. The coherence value for the corresponding LDA model was 0.4566. Examining the digram across these 8 tweet topics indicates the following:

1. Topic #1 (Work & Tech) - office, work, technology
2. Topic #2 (IT) - azure, support, security
3. Topic #3 (Cloud & Programming) - team, cloud, python
4. Topic #4 (Finanial Markets) - work, company, sell, buy
5. Topic #5 (Business) - buy, business, premium, office, acquisition
6. Topic #6 (War/Conflict) - agressor, evil, war
7. Topic #7 (Gaming) - game, video, technology
8. Topic #8 (Microsoft Products) - explorer, zoom, edge, update, product, version