Dynamic_Topic_Modeling_Loading_&_Evaluating_Models_(Hw_2_Part_

March 26, 2021

1 Hw2 Part 2:

1.1 DTM

1.2 Loading & Evaluating Prior Models

[ ]:
```
# setting up our imports
import pandas as pd
import numpy

# Gensim
import gensim
from gensim.utils import simple_preprocess
import gensim.corpora as corpora
from gensim.models import ldaseqmodel
from gensim.models.wrappers import DtmModel
from gensim.corpora import Dictionary, bleicorpus
from gensim.matutils import hellinger
from gensim.models import ldaseqmodel
from gensim.corpora import Dictionary, bleicorpus
import numpy
from gensim.matutils import hellinger
import pickle
from gensim.test.utils import datapath
from gensim.corpora import Dictionary, bleicorpus
from gensim.matutils import hellinger
from gensim import corpora, models, similarities
from gensim.models.coherencemodel import CoherenceModel
from gensim.models.wrappers.dtmmodel import DtmModel

from gensim.models import callbacks
from gensim.models.callbacks import CallbackAny2Vec
from gensim.test.utils import get_tmpfile

# NLTK
import nltk
```

```
from nltk.corpus import stopwords
nltk.download('stopwords')

# spacy
import spacy

# Plotting tools
import pyLDAvis
import pyLDAvis.gensim
import matplotlib.pyplot as plt
from pprint import pprint
%matplotlib inline

#!pip install playsound
from playsound import playsound

import warnings
warnings.filterwarnings("ignore",category=DeprecationWarning)
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
<ipython-input-1-cf1871ecb900> in <module>()
     35
     36 # Plotting tools
---> 37 import pyLDAvis
     38 import pyLDAvis.gensim
     39 import matplotlib.pyplot as plt

ModuleNotFoundError: No module named 'pyLDAvis'

---------------------------------------------------------------------------
NOTE: If your import is failing due to a missing package, you can
manually install dependencies using either !pip or !apt.
To view examples of installing some common dependencies, click the
"Open Examples" button below.
---------------------------------------------------------------------------
```

1.2.1 Data Preprocessing

[ ]:
```
df = pd.read_excel('Index_fund_all.xlsx')

df1 = df[['filing_year','principal_strategies']]    # dtype('O') w. 6171 rows

df1 = df1.dropna(axis=0)    # dtype('O') w. 5761 rows
```

```
df1.sort_values(by='filing_year')

# creating time slice
x = df1.filing_year.value_counts().sort_values()
time_slice = x.tolist()

df2 = df1.principal_strategies
tl = df2.values.tolist()    # list of strings w. 5761 strings of txt

# lowercases, tokenizes, de-accents each sentence
def sentence_to_words(sentences):
    for sentence in sentences:
        yield(gensim.utils.simple_preprocess(sentence, deacc=True))  #␣
 ↪deacc=True removes punctuations

wl = list(sentence_to_words(tl))    # list of strings w. 5761 strings of␣
 ↪tokenized words

# removing stop-words
stop_words = stopwords.words('english')
stop_words.extend(['from', 'subject', 're', 'edu', 'use','index','fund'])

def remove_stopwords(texts):
    return [[word for word in simple_preprocess(str(doc)) if word not in␣
 ↪stop_words] for doc in texts]

wl1 = remove_stopwords(wl)

# lemmatization - keeping only noun, adj, vb, adv
nlp = spacy.load('en', disable=['parser', 'ner'])

def lemmatization(texts, allowed_postags=['NOUN', 'ADJ', 'VERB', 'ADV']):
    texts_out = []
    for sentence in texts:
        doc = nlp(" ".join(sentence))
        texts_out.append([token.lemma_ for token in doc if token.pos_ in␣
 ↪allowed_postags])
    return texts_out

wl2 = lemmatization(wl1, allowed_postags=['NOUN', 'ADJ', 'VERB', 'ADV'])

# Create Dictionary
dict = corpora.Dictionary(wl2)

# Create Corpus
corpus = [dict.doc2bow(word) for word in wl2]
```

```
print('Number of unique tokens: %d' % len(dict))
print('Number of documents: %d' % len(corpus))
```

```
Number of unique tokens: 3820
Number of documents: 5761
```

1.2.2 Saving Corpus & Dictionary

[ ]:
```
revNo = 1

#model_list.save('hw2p2_dtm_rev{}.gensim'.format(revNo))

corpora.MmCorpus.serialize('hw2p2_dtm_corpus_rev{}.gensim'.format(revNo),␣
↪corpus)

dict.save('hw2p2_dtm_dictionary_rev{}.gensim'.format(revNo))

#playsound('COD.mp3',True)
```

```
WARNING:smart_open.smart_open_lib:this function is deprecated, use
smart_open.open instead
WARNING:smart_open.smart_open_lib:this function is deprecated, use
smart_open.open instead
WARNING:smart_open.smart_open_lib:this function is deprecated, use
smart_open.open instead
```

1.2.3 Loading Model, Corpus, & Dictionary

[ ]:
```
loaded_model =  models.LdaModel.load('hw2p2_dtm_model_rev1.gensim')
loaded_dict = corpora.Dictionary.load('hw2p2_dtm_dictionary_rev1.gensim')
loaded_corpus = corpora.MmCorpus('hw2p2_dtm_corpus_rev1.gensim')
```

```
WARNING:smart_open.smart_open_lib:this function is deprecated, use
smart_open.open instead
WARNING:root:random_state not set so using default value
WARNING:smart_open.smart_open_lib:this function is deprecated, use
smart_open.open instead
WARNING:root:failed to load state from hw2p2_dtm_model_rev1.gensim.state: [Errno
2] No such file or directory: 'hw2p2_dtm_model_rev1.gensim.state'
WARNING:smart_open.smart_open_lib:this function is deprecated, use
smart_open.open instead
WARNING:smart_open.smart_open_lib:this function is deprecated, use
smart_open.open instead
WARNING:smart_open.smart_open_lib:this function is deprecated, use
smart_open.open instead
```

1.2.4 Coherence Scores

[ ]:
```
topics_wrapper = loaded_model.dtm_coherence(time=0)
#texts = pickle.load(loaded_corpus)

cm_wrapper = CoherenceModel(topics=topics_wrapper,
                            texts=loaded_corpus,
                            dictionary=loaded_dict,
                            coherence='c_v')

print ("C_v topic coherence")
print ("Wrapper coherence is ", cm_wrapper.get_coherence())
```

```
C_v topic coherence

WARNING:smart_open.smart_open_lib:this function is deprecated, use
smart_open.open instead
C:\Users\vamsi\Anaconda3\lib\site-
packages\gensim\topic_coherence\direct_confirmation_measure.py:193:
RuntimeWarning: invalid value encountered in true_divide
  numerator = (co_occur_count / num_docs) + EPSILON
C:\Users\vamsi\Anaconda3\lib\site-
packages\gensim\topic_coherence\direct_confirmation_measure.py:194:
RuntimeWarning: invalid value encountered in true_divide
  denominator = (w_prime_count / num_docs) * (w_star_count / num_docs)
C:\Users\vamsi\Anaconda3\lib\site-
packages\gensim\topic_coherence\direct_confirmation_measure.py:189:
RuntimeWarning: invalid value encountered in true_divide
  co_doc_prob = co_occur_count / num_docs

Wrapper coherence is  nan
```

1.2.5 Test Metrics

[ ]:
```
pprint(loaded_model.print_topics(num_topics=2,num_words=1))
```

```
C:\Users\vamsi\Anaconda3\lib\site-
packages\gensim\models\wrappers\dtmmodel.py:529: UserWarning: The parameter
`num_words` is deprecated, will be removed in 4.0.0, use `topn` instead.
  warnings.warn("The parameter `num_words` is deprecated, will be removed in
4.0.0, use `topn` instead.")

['0.080*fund',
 '0.113*index',
 '0.080*fund',
 '0.114*index',
 '0.079*fund',
 '0.121*index',
 '0.079*fund',
 '0.132*index',
```

```
 '0.078*fund',
 '0.140*index']
```

[ ]:
```
c = loaded_model.dtm_coherence(time=8,num_words=3)
c1 = loaded_model.dtm_coherence(time=7,num_words=3)
c2 = pd.DataFrame(c)
print(c2)
```

```
            0           1           2
0     expense        fund         fee
1       index       stock        make
2       index       month      market
3       index  investment        fund
4       index        fund    security
5    dividend       value        fund
6       share        fund      invest
7        fund       index  investment
8        fund       index    underlie
9        fund      invest        rate
10       fund    security      invest
11      index        fund    security
12      index        fund    security
13      index        fund     company
14   underlie       index    security
15       fund    security        cost
16        cap       index        fund
17      index        fund         day
18      index    underlie        fund
19      index      market       stock
20       fund        risk      market
21       gold      mining     foreign
22      index        fund        bond
23      index    underlie  investment
24       fund  investment   commodity
25  leveraged     inverse       stock
26       real      estate       index
27       fund       index  investment
28   security        fund       asset
29      index    security      factor
30      index        fund     company
31     market      emerge     country
32      index    currency        fund
33      index        fund       stock
34    company  investment        fund
35     market     economy      emerge
36    company         try       index
37       fund      invest    security
38      index        fund     company
```

```
39      index    underlie      equity
40       fund       index     manager
41      index        fund    security
42      index    underlie       stock
43       bond       index       score
44        oil         gas     natural
45      index        fund  investment
46       bond       index    underlie
47       fund       index    position
48       fund     company      sector
49     option        fund       index
```

[ ]:
```
slice = 8
doc_topic, topic_term, doc_lengths, term_frequency, vocab = loaded_model.
 ↪dtm_vis(loaded_corpus,time=slice)

pyLDAvis.enable_notebook()
vis_dtm = pyLDAvis.prepare(topic_term_dists=topic_term,
                           doc_topic_dists=doc_topic,
                           doc_lengths=doc_lengths,
                           vocab=vocab,
                           term_frequency=term_frequency)
print('slice = {}'.format(slice))
pyLDAvis.display(vis_dtm)
```

```
WARNING:smart_open.smart_open_lib:this function is deprecated, use
smart_open.open instead
WARNING:smart_open.smart_open_lib:this function is deprecated, use
smart_open.open instead
slice = 8
```

[ ]:
```
<IPython.core.display.HTML object>
```

[ ]:
```
slice = 7
doc_topic, topic_term, doc_lengths, term_frequency, vocab = loaded_model.
 ↪dtm_vis(loaded_corpus,time=slice)

pyLDAvis.enable_notebook()
vis_dtm = pyLDAvis.prepare(topic_term_dists=topic_term,
                           doc_topic_dists=doc_topic,
                           doc_lengths=doc_lengths,
                           vocab=vocab,
                           term_frequency=term_frequency)
print('slice = {}'.format(slice))
pyLDAvis.display(vis_dtm)
```

```
WARNING:smart_open.smart_open_lib:this function is deprecated, use
```

```
smart_open.open instead
WARNING:smart_open.smart_open_lib:this function is deprecated, use
smart_open.open instead

slice = 7
```

[ ]:
```
<IPython.core.display.HTML object>
```

[ ]:
```
slice = 6
doc_topic, topic_term, doc_lengths, term_frequency, vocab = loaded_model.
 ↪dtm_vis(loaded_corpus,time=slice)

pyLDAvis.enable_notebook()
vis_dtm = pyLDAvis.prepare(topic_term_dists=topic_term,
                           doc_topic_dists=doc_topic,
                           doc_lengths=doc_lengths,
                           vocab=vocab,
                           term_frequency=term_frequency)
print('slice = {}'.format(slice))
pyLDAvis.display(vis_dtm)
```

```
WARNING:smart_open.smart_open_lib:this function is deprecated, use
smart_open.open instead
WARNING:smart_open.smart_open_lib:this function is deprecated, use
smart_open.open instead

slice = 6
```

[ ]:
```
<IPython.core.display.HTML object>
```

[ ]:
```
slice = 5
doc_topic, topic_term, doc_lengths, term_frequency, vocab = loaded_model.
 ↪dtm_vis(loaded_corpus,time=slice)

pyLDAvis.enable_notebook()
vis_dtm = pyLDAvis.prepare(topic_term_dists=topic_term,
                           doc_topic_dists=doc_topic,
                           doc_lengths=doc_lengths,
                           vocab=vocab,
                           term_frequency=term_frequency)
print('slice = {}'.format(slice))
pyLDAvis.display(vis_dtm)
```

```
WARNING:smart_open.smart_open_lib:this function is deprecated, use
smart_open.open instead
WARNING:smart_open.smart_open_lib:this function is deprecated, use
smart_open.open instead

slice = 5
```

[ ]: <IPython.core.display.HTML object>

[ ]:
```
slice = 4
doc_topic, topic_term, doc_lengths, term_frequency, vocab = loaded_model.
 ↪dtm_vis(loaded_corpus,time=slice)

pyLDAvis.enable_notebook()
vis_dtm = pyLDAvis.prepare(topic_term_dists=topic_term,
                           doc_topic_dists=doc_topic,
                           doc_lengths=doc_lengths,
                           vocab=vocab,
                           term_frequency=term_frequency)
print('slice = {}'.format(slice))
pyLDAvis.display(vis_dtm)
```

```
WARNING:smart_open.smart_open_lib:this function is deprecated, use
smart_open.open instead
WARNING:smart_open.smart_open_lib:this function is deprecated, use
smart_open.open instead
```

```
slice = 4
```

[ ]: <IPython.core.display.HTML object>

[ ]:
```
slice = 3
doc_topic, topic_term, doc_lengths, term_frequency, vocab = loaded_model.
 ↪dtm_vis(loaded_corpus,time=slice)

pyLDAvis.enable_notebook()
vis_dtm = pyLDAvis.prepare(topic_term_dists=topic_term,
                           doc_topic_dists=doc_topic,
                           doc_lengths=doc_lengths,
                           vocab=vocab,
                           term_frequency=term_frequency)
print('slice = {}'.format(slice))
pyLDAvis.display(vis_dtm)
```

```
WARNING:smart_open.smart_open_lib:this function is deprecated, use
smart_open.open instead
WARNING:smart_open.smart_open_lib:this function is deprecated, use
smart_open.open instead
```

```
slice = 3
```

[ ]: <IPython.core.display.HTML object>

[ ]:
```
slice = 2
```

```
doc_topic, topic_term, doc_lengths, term_frequency, vocab = loaded_model.
 ↪dtm_vis(loaded_corpus,time=slice)

pyLDAvis.enable_notebook()
vis_dtm = pyLDAvis.prepare(topic_term_dists=topic_term,
                           doc_topic_dists=doc_topic,
                           doc_lengths=doc_lengths,
                           vocab=vocab,
                           term_frequency=term_frequency)
print('slice = {}'.format(slice))
pyLDAvis.display(vis_dtm)
```

```
WARNING:smart_open.smart_open_lib:this function is deprecated, use
smart_open.open instead
WARNING:smart_open.smart_open_lib:this function is deprecated, use
smart_open.open instead
```

```
slice = 2
```

[ ]:
```
<IPython.core.display.HTML object>
```

[ ]:
```
slice = 1
doc_topic, topic_term, doc_lengths, term_frequency, vocab = loaded_model.
 ↪dtm_vis(loaded_corpus,time=slice)

pyLDAvis.enable_notebook()
vis_dtm = pyLDAvis.prepare(topic_term_dists=topic_term,
                           doc_topic_dists=doc_topic,
                           doc_lengths=doc_lengths,
                           vocab=vocab,
                           term_frequency=term_frequency)
print('slice = {}'.format(slice))
pyLDAvis.display(vis_dtm)
```

```
WARNING:smart_open.smart_open_lib:this function is deprecated, use
smart_open.open instead
WARNING:smart_open.smart_open_lib:this function is deprecated, use
smart_open.open instead
C:\Users\vamsi\Anaconda3\lib\site-packages\pyLDAvis\_prepare.py:257:
FutureWarning: Sorting because non-concatenation axis is not aligned. A future
version
of pandas will change to not sort by default.

To accept the future behavior, pass 'sort=False'.

To retain the current behavior and silence the warning, pass 'sort=True'.

  return pd.concat([default_term_info] + list(topic_dfs))
```

```
slice = 1
```

[ ]: <IPython.core.display.HTML object>

[ ]:
```
slice = 0
doc_topic, topic_term, doc_lengths, term_frequency, vocab = loaded_model.
 ↪dtm_vis(loaded_corpus,time=slice)

pyLDAvis.enable_notebook()
vis_dtm = pyLDAvis.prepare(topic_term_dists=topic_term,
                           doc_topic_dists=doc_topic,
                           doc_lengths=doc_lengths,
                           vocab=vocab,
                           term_frequency=term_frequency)
print('slice = {}'.format(slice))
pyLDAvis.display(vis_dtm)
```

```
WARNING:smart_open.smart_open_lib:this function is deprecated, use
smart_open.open instead
WARNING:smart_open.smart_open_lib:this function is deprecated, use
smart_open.open instead
slice = 0
```

[ ]: <IPython.core.display.HTML object>

[ ]: