LDA

April 23, 2022

1 Topic Modeling: Latent Dirichlet Allocation (LDA)

Topic modeling is a type of statistical modeling for discovering the abstract “topics” that occur in a collection of documents, and Latent Dirichlet Allocation (LDA) is one of the method. LDA is a generative probabilistic model that assumes each topic is a mixture over an underlying set of words, and each document is a mixture of over a set of topic probabilities.

Main idea - Every documents is a mixture of topics. e.g. Document 1 is 90% topic A and 10% topic B, while Document 2 is 30% topic A and 70% topic B.

- Every topic is a mixture of words. e.g. two-topic model of American news, with one topic for “politics” and one for “entertainment.” The most common words in the politics topic might be “President”, “Congress”, and “government”, while the entertainment topic may be made up of words such as “movies”, “television”, and “actor”.

Three main parts of LDA

1. Dimensionality Reduction: Rather than representing a text T in its feature space as `{Word_i: count(Word_i, T) for Word_i in Vocabulary}`, you can represent it in a topic space as `{Topic_i: Weight(Topic_i, T) for Topic_i in Topics}`.
2. Unsupervised Learning: By doing topic modeling, we build clusters of words rather than clusters of texts. A text is thus a mixture of all the topics, each having a specific weight.
3. Tagging: abstract “topics” that occur in a collection of documents that best represents the information in them.

- psi, the distribution of words for each topic K
- phi, the distribution of topics for each document i
- α: parameter is Dirichlet prior concentration parameter that represents document-topic density — with a higher alpha, documents are assumed to be made up of more topics and result in more specific topic distribution per document.
- β: is the same prior concentration parameter that represents topic-word density — with high beta, topics are assumed to made of up most of the words and result in a more specific word distribution per topic.

2 Import data

```
[ ]: import numpy as np
     import pandas as pd
     import sklearn
     import seaborn as sns
     import matplotlib.pyplot as plt
```

```
[ ]: df = pd.read_csv('processed_response.zip')
```

- what is lambda?
- how to intperpret the result of the topics?

2.1 Running LDA using bags of words

```
[ ]: import gensim
     import gensim.corpora as corpora
```

```
[ ]: from ast import literal_eval

     #convert the response to a list
     df['clean_responses'] = df['clean_responses'].apply(literal_eval)
```

```
[ ]: # Create Dictionary
     id2word = corpora.Dictionary(df['clean_responses'])

     # Create Corpus
     texts = df['clean_responses']

     # Term Document Frequency
     corpus = [id2word.doc2bow(text) for text in texts]

     print(corpus[0])
```

```
[(0, 1), (1, 1), (2, 1), (3, 1), (4, 1), (5, 1), (6, 1), (7, 2), (8, 1), (9, 1),
(10, 1), (11, 1), (12, 1), (13, 1), (14, 1), (15, 1), (16, 1), (17, 1), (18, 1),
(19, 1), (20, 1), (21, 1), (22, 1), (23, 1)]
```

```
[ ]: #times a word appears
     corpus_0 = corpus[0]
     for i in range(len(corpus_0)):
         print("Word {} (\"{}\") appears {} time.".format(corpus_0[i][0],␣
      ↪id2word[corpus_0[i][0]], corpus_0[i][1]))
```

```
Word 0 ("abstract") appears 1 time.
Word 1 ("answer") appears 1 time.
Word 2 ("approach") appears 1 time.
Word 3 ("attack") appears 1 time.
Word 4 ("construct") appears 1 time.
Word 5 ("focus") appears 1 time.
Word 6 ("form") appears 1 time.
Word 7 ("framework") appears 2 time.
Word 8 ("good") appears 1 time.
Word 9 ("hard") appears 1 time.
Word 10 ("instead") appears 1 time.
Word 11 ("knowledg") appears 1 time.
Word 12 ("matt") appears 1 time.
Word 13 ("moral") appears 1 time.
Word 14 ("plato") appears 1 time.
Word 15 ("poll") appears 1 time.
Word 16 ("preparatori") appears 1 time.
Word 17 ("set") appears 1 time.
Word 18 ("specif") appears 1 time.
Word 19 ("strength") appears 1 time.
Word 20 ("veri") appears 1 time.
Word 21 ("weak") appears 1 time.
Word 22 ("whole") appears 1 time.
Word 23 ("would") appears 1 time.
```

In `corpus`, every tuple represent (index of the word, frequency), which is shown above.

[ ]:
```python
from pprint import pprint
# number of topics
num_topics = 10
# Build LDA model
lda_model = gensim.models.LdaMulticore(corpus=corpus,
                                       id2word=id2word,
                                       num_topics=num_topics)
# Print the Keyword in the 10 topics
pprint(lda_model.print_topics())
```

```
[(0,
  '0.013*"would" + 0.011*"becaus" + 0.009*"one" + 0.007*"use" + 0.007*"make" + '
  '0.006*"like" + 0.006*"chang" + 0.006*"differ" + 0.005*"data" + '
  '0.005*"increas"'),
 (1,
  '0.016*"system" + 0.011*"would" + 0.008*"level" + 0.007*"think" + '
  '0.007*"individu" + 0.007*"understand" + 0.007*"peopl" + 0.007*"one" + '
  '0.007*"differ" + 0.006*"exampl"'),
 (2,
  '0.012*"would" + 0.011*"model" + 0.009*"one" + 0.009*"use" + 0.009*"valu" + '
  '0.008*"n" + 0.008*"p" + 0.008*"time" + 0.008*"becaus" + 0.007*"differ"'),
 (3,
  '0.014*"use" + 0.013*"x" + 0.010*"studi" + 0.009*"would" + 0.009*"one" + '
  '0.008*"becaus" + 0.007*"research" + 0.007*"understand" + 0.007*"also" + '
  '0.007*"think"'),
 (4,
  '0.010*"would" + 0.010*"use" + 0.009*"could" + 0.009*"product" + '
  '0.007*"like" + 0.006*"becaus" + 0.006*"risk" + 0.006*"also" + '
  '0.006*"market" + 0.006*"compani"'),
 (5,
  '0.011*"use" + 0.008*"becaus" + 0.007*"state" + 0.007*"would" + '
  '0.006*"differ" + 0.006*"think" + 0.005*"problem" + 0.005*"chang" + '
  '0.005*"peopl" + 0.005*"exampl"'),
 (6,
  '0.018*"use" + 0.011*"data" + 0.009*"variabl" + 0.008*"make" + '
  '0.007*"becaus" + 0.007*"line" + 0.007*"would" + 0.006*"one" + 0.006*"show" '
  '+ 0.006*"exampl"'),
 (7,
  '0.010*"would" + 0.010*"compani" + 0.008*"differ" + 0.007*"peopl" + '
  '0.007*"work" + 0.006*"time" + 0.006*"use" + 0.005*"one" + 0.005*"becaus" + '
  '0.005*"exampl"'),
 (8,
  '0.012*"peopl" + 0.009*"one" + 0.009*"theori" + 0.007*"becaus" + '
  '0.006*"level" + 0.006*"make" + 0.005*"differ" + 0.005*"exampl" + '
  '0.005*"polit" + 0.005*"social"'),
 (9,
  '0.012*"argument" + 0.012*"problem" + 0.011*"use" + 0.011*"evid" + '
  '0.009*"make" + 0.008*"becaus" + 0.007*"would" + 0.007*"think" + '
  '0.007*"bias" + 0.006*"exampl"')]
```

From the Topic 0 is a represented as 0.013*“would” + 0.011*”becaus” + 0.009*“one” + 0.007*”use” + 0.007*“make” + 0.006*”like” + 0.006*“chang” + 0.006*”differ” + 0.005*“data” + 0.005*”increas.” It means the top 10 keywords that contribute to this topic are: would, becaus, one.. and so on and the weight of would on topic 0 is 0.013.

[ ]:
```python
#print the top 20 words in each topics
topics_matrix = lda_model.show_topics(formatted=False, num_words=20)
topics_matrix = np.array(topics_matrix)
topic_words = topics_matrix[:,1]

for i in topic_words:
    print([str(word[0]) for word in i])
    print()
```

```
['would', 'becaus', 'one', 'use', 'make', 'like', 'chang', 'differ', 'data',
'increas', 'mean', 'exampl', 'group', 'c', 'also', 'need', 'think', 'v',
'activ', 'water']

['system', 'would', 'level', 'think', 'individu', 'understand', 'peopl', 'one',
'differ', 'exampl', 'effect', 'complex', 'need', 'interact', 'becaus', 'also',
'group', 'social', 'market', 'motiv']

['would', 'model', 'one', 'use', 'valu', 'n', 'p', 'time', 'becaus', 'differ',
'number', 'probabl', 'b', 'distribut', 'x', 'function', 'first', 'data', 'case',
'also']
```

```
['use', 'x', 'studi', 'would', 'one', 'becaus', 'research', 'understand',
'also', 'think', 'differ', 'effect', 'need', 'help', 'way', 'could', 'make',
'hypothesi', 'test', 'data']

['would', 'use', 'could', 'product', 'like', 'becaus', 'risk', 'also', 'market',
'compani', 'think', 'one', 'us', 'creat', 'make', 'invest', 'differ', 'countri',
'increas', 'idea']

['use', 'becaus', 'state', 'would', 'differ', 'think', 'problem', 'chang',
'peopl', 'exampl', 'one', 'way', 'mean', 'make', 'also', 'class', 'hypothesi',
'could', 'structur', 'network']

['use', 'data', 'variabl', 'make', 'becaus', 'line', 'would', 'one', 'show',
'exampl', 'activ', 'mean', 'bias', 'word', 'point', 'could', 'like', 'control',
'also', 'differ']

['would', 'compani', 'differ', 'peopl', 'work', 'time', 'use', 'one', 'becaus',
'exampl', 'like', 'also', 'make', 'could', 'valu', 'interest', 'way', 'market',
'help', 'chang']

['peopl', 'one', 'theori', 'becaus', 'level', 'make', 'differ', 'exampl',
'polit', 'social', 'would', 'cultur', 'societi', 'human', 'person', 'individu',
'way', 'base', 'govern', 'moral']

['argument', 'problem', 'use', 'evid', 'make', 'becaus', 'would', 'think',
'bias', 'exampl', 'state', 'also', 'thesi', 'one', 'differ', 'way', 'claim',
'solut', 'base', 'effect']


/var/folders/0h/xyv81g2n7sj6zr0c9cw30gkc0000gn/T/ipykernel_13803/4262890021.py:3
: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences
(which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths
or shapes) is deprecated. If you meant to do this, you must specify
'dtype=object' when creating the ndarray.
  topics_matrix = np.array(topics_matrix)
```

[ ]:
```
#visualize the result

import pyLDAvis
# import pyLDAvis.gensim
import pyLDAvis.gensim_models as gensimvis
import os
import pickle

# Visualize the topics
pyLDAvis.enable_notebook()
```

[ ]:
```
LDAvis_prepared = gensimvis.prepare(lda_model, corpus, id2word)
```

```
  if LooseVersion(np.__version__) < '1.13':
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/setuptools/_distutils/version.py:351: DeprecationWarning: distutils
Version classes are deprecated. Use packaging.version instead.
  other = LooseVersion(other)
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/joblib/backports.py:36: DeprecationWarning: distutils Version classes
are deprecated. Use packaging.version instead.
  if LooseVersion(np.__version__) < '1.13':
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/setuptools/_distutils/version.py:351: DeprecationWarning: distutils
Version classes are deprecated. Use packaging.version instead.
  other = LooseVersion(other)
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/joblib/backports.py:36: DeprecationWarning: distutils Version classes
are deprecated. Use packaging.version instead.
  if LooseVersion(np.__version__) < '1.13':
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/setuptools/_distutils/version.py:351: DeprecationWarning: distutils
Version classes are deprecated. Use packaging.version instead.
  other = LooseVersion(other)
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/joblib/backports.py:36: DeprecationWarning: distutils Version classes
are deprecated. Use packaging.version instead.
  if LooseVersion(np.__version__) < '1.13':
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/setuptools/_distutils/version.py:351: DeprecationWarning: distutils
Version classes are deprecated. Use packaging.version instead.
  other = LooseVersion(other)
```

```
[ ]: pyLDAvis.display(LDAvis_prepared)
```

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

The relevance metric: λ

- $\lambda = 1$: ranking soley by probability is decending order
- $\lambda = 0$: ranking soley by the lift. lift = the ratio of a term's probability within a topic to its marginal probability across the corpus. It generally decreases with globally frequent terms. But it can be noisy too if the rare terms only occur in one single topic.

$P(T|w)$: the likelihood that observed word w was generated by latent topic T. We use it to determine how informative a specific word w can inform the topics. If word tells little of the topic mixture it will receive a low distinctiveness score.

```
[ ]: pyLDAvis.save_html(LDAvis_prepared, 'lda.html')
```

2.2 Running LDA using tf-idf

```
[ ]: from gensim import corpora, models

     tfidf = models.TfidfModel(corpus)
     corpus_tfidf = tfidf[corpus]

     # Build LDA model on tf-idf
     lda_model_tf = gensim.models.LdaMulticore(corpus=corpus_tfidf,
                                             id2word=id2word,
                                             num_topics=num_topics)
     # Print the Keyword in the 10 topics
     pprint(lda_model_tf.print_topics())
```

```
[(0,
  '0.004*"system" + 0.004*"peopl" + 0.003*"differ" + 0.003*"level" + '
  '0.003*"one" + 0.003*"emot" + 0.003*"would" + 0.003*"think" + '
  '0.003*"understand" + 0.003*"interact"'),
 (1,
  '0.009*"problem" + 0.005*"solut" + 0.004*"solv" + 0.004*"use" + '
  '0.003*"think" + 0.003*"constraint" + 0.003*"would" + 0.003*"differ" + '
  '0.003*"goal" + 0.003*"help"'),
 (2,
  '0.009*"x" + 0.006*"data" + 0.006*"variabl" + 0.006*"distribut" + '
  '0.006*"sampl" + 0.005*"probabl" + 0.005*"valu" + 0.005*"model" + 0.004*"p" '
  '+ 0.004*"would"'),
 (3,
  '0.003*"would" + 0.003*"compani" + 0.003*"peopl" + 0.003*"market" + '
  '0.003*"differ" + 0.003*"think" + 0.002*"make" + 0.002*"one" + 0.002*"use" + '
  '0.002*"need"'),
 (4,
  '0.009*"poll" + 0.009*"student" + 0.008*"complet" + 0.008*"https" + '
  '0.007*"present" + 0.007*"com" + 0.007*"googl" + 0.007*"doc" + 0.006*"edit" '
  '+ 0.006*"document"'),
 (5,
  '0.007*"n" + 0.004*"would" + 0.004*"node" + 0.004*"algorithm" + 0.003*"tree" '
  '+ 0.003*"time" + 0.003*"number" + 0.003*"use" + 0.003*"one" + 0.003*"list"'),
 (6,
  '0.003*"would" + 0.002*"co" + 0.002*"x" + 0.002*"increas" + 0.002*"water" + '
  '0.002*"becaus" + 0.002*"chang" + 0.002*"system" + 0.002*"use" + '
  '0.002*"peopl"'),
 (7,
  '0.006*"compani" + 0.004*"market" + 0.003*"product" + 0.003*"would" + '
  '0.003*"use" + 0.003*"custom" + 0.003*"risk" + 0.003*"cost" + 0.003*"make" + '
  '0.002*"think"'),
 (8,
  '0.003*"would" + 0.003*"use" + 0.003*"energi" + 0.002*"effect" + '
  '0.002*"becaus" + 0.002*"differ" + 0.002*"water" + 0.002*"one" + '
  '0.002*"cell" + 0.002*"observ"'),
 (9,
  '0.007*"argument" + 0.006*"evid" + 0.005*"p" + 0.005*"hypothesi" + 0.004*"b" '
  '+ 0.004*"induct" + 0.004*"thesi" + 0.004*"c" + 0.004*"true" + 0.004*"use"')]
```

[ ]:
```
#print the top 20 words in each topics
topics_matrix = lda_model_tf.show_topics(formatted=False, num_words=20)
topics_matrix = np.array(topics_matrix)
topic_words = topics_matrix[:,1]

for i in topic_words:
    print([str(word[0]) for word in i])
    print()
```

```
['system', 'peopl', 'differ', 'level', 'one', 'emot', 'would', 'think',
'understand', 'interact', 'use', 'individu', 'cultur', 'becaus', 'agent',
'social', 'way', 'exampl', 'emerg', 'make']

['problem', 'solut', 'solv', 'use', 'think', 'constraint', 'would', 'differ',
'goal', 'help', 'one', 'process', 'understand', 'need', 'appli', 'level',
'becaus', 'could', 'identifi', 'activ']

['x', 'data', 'variabl', 'distribut', 'sampl', 'probabl', 'valu', 'model', 'p',
'would', 'mean', 'use', 'function', 'treatment', 'vector', 'calcul', 'differ',
'number', 'test', 'random']

['would', 'compani', 'peopl', 'market', 'differ', 'think', 'make', 'one', 'use',
'need', 'strategi', 'becaus', 'system', 'could', 'product', 'countri', 'also',
'chang', 'effect', 'group']

['poll', 'student', 'complet', 'https', 'present', 'com', 'googl', 'doc',
'edit', 'document', 'usp', 'share', 'would', 'use', 'argument', 'think',
'becaus', 'one', 'make', 'could']

['n', 'would', 'node', 'algorithm', 'tree', 'time', 'number', 'use', 'one',
'list', 'becaus', 'valu', 'sort', 'first', 'make', 'x', 'optim', 'case', 'row',
'think']

['would', 'co', 'x', 'increas', 'water', 'becaus', 'chang', 'system', 'use',
'peopl', 'one', 'differ', 'carbon', 'time', 'histogram', 'earth', 'state',
'temperatur', 'citi', 'could']

['compani', 'market', 'product', 'would', 'use', 'custom', 'risk', 'cost',
'make', 'think', 'peopl', 'invest', 'price', 'activ', 'rate', 'differ',
'financi', 'also', 'becaus', 'busi']
```

```
['would', 'use', 'energi', 'effect', 'becaus', 'differ', 'water', 'one', 'cell',
'observ', 'test', 'could', 'chang', 'hypothesi', 'increas', 'complianc', 'like',
'theori', 'time', 'level']

['argument', 'evid', 'p', 'hypothesi', 'b', 'induct', 'thesi', 'c', 'true',
'use', 'data', 'conclus', 'valid', 'premis', 'deduct', 'sentenc', 'logic',
'theori', 'test', 'q']
```

```
/var/folders/0h/xyv81g2n7sj6zr0c9cw30gkc0000gn/T/ipykernel_13803/697313039.py:3:
VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences
(which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths
or shapes) is deprecated. If you meant to do this, you must specify
'dtype=object' when creating the ndarray.
  topics_matrix = np.array(topics_matrix)
```

```
[ ]: LDAvis_prepared_tf = gensimvis.prepare(lda_model_tf, corpus_tfidf, id2word)
```

```
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/joblib/backports.py:36: DeprecationWarning: distutils Version classes
are deprecated. Use packaging.version instead.
  if LooseVersion(np.__version__) < '1.13':
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/setuptools/_distutils/version.py:351: DeprecationWarning: distutils
Version classes are deprecated. Use packaging.version instead.
  other = LooseVersion(other)
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/joblib/backports.py:36: DeprecationWarning: distutils Version classes
are deprecated. Use packaging.version instead.
  if LooseVersion(np.__version__) < '1.13':
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/setuptools/_distutils/version.py:351: DeprecationWarning: distutils
Version classes are deprecated. Use packaging.version instead.
  other = LooseVersion(other)
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/joblib/backports.py:36: DeprecationWarning: distutils Version classes
are deprecated. Use packaging.version instead.
  if LooseVersion(np.__version__) < '1.13':
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/setuptools/_distutils/version.py:351: DeprecationWarning: distutils
Version classes are deprecated. Use packaging.version instead.
  other = LooseVersion(other)
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/joblib/backports.py:36: DeprecationWarning: distutils Version classes
are deprecated. Use packaging.version instead.
  if LooseVersion(np.__version__) < '1.13':
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/joblib/backports.py:36: DeprecationWarning: distutils Version classes
are deprecated. Use packaging.version instead.
```

```
  other = LooseVersion(other)
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/joblib/backports.py:36: DeprecationWarning: distutils Version classes
are deprecated. Use packaging.version instead.
  if LooseVersion(np.__version__) < '1.13':
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/setuptools/_distutils/version.py:351: DeprecationWarning: distutils
Version classes are deprecated. Use packaging.version instead.
  other = LooseVersion(other)
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/joblib/backports.py:36: DeprecationWarning: distutils Version classes
are deprecated. Use packaging.version instead.
  if LooseVersion(np.__version__) < '1.13':
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/setuptools/_distutils/version.py:351: DeprecationWarning: distutils
Version classes are deprecated. Use packaging.version instead.
  other = LooseVersion(other)
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/joblib/backports.py:36: DeprecationWarning: distutils Version classes
are deprecated. Use packaging.version instead.
  if LooseVersion(np.__version__) < '1.13':
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/setuptools/_distutils/version.py:351: DeprecationWarning: distutils
Version classes are deprecated. Use packaging.version instead.
  other = LooseVersion(other)
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/joblib/backports.py:36: DeprecationWarning: distutils Version classes
are deprecated. Use packaging.version instead.
  if LooseVersion(np.__version__) < '1.13':
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/setuptools/_distutils/version.py:351: DeprecationWarning: distutils
Version classes are deprecated. Use packaging.version instead.
  other = LooseVersion(other)
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/joblib/backports.py:36: DeprecationWarning: distutils Version classes
are deprecated. Use packaging.version instead.
  if LooseVersion(np.__version__) < '1.13':
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/setuptools/_distutils/version.py:351: DeprecationWarning: distutils
Version classes are deprecated. Use packaging.version instead.
  other = LooseVersion(other)
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/joblib/backports.py:36: DeprecationWarning: distutils Version classes
are deprecated. Use packaging.version instead.
  if LooseVersion(np.__version__) < '1.13':
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/setuptools/_distutils/version.py:351: DeprecationWarning: distutils
Version classes are deprecated. Use packaging.version instead.
```

```
    other = LooseVersion(other)
```

[ ]:
```
pyLDAvis.display(LDAvis_prepared_tf)
```

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

[ ]:
```
pyLDAvis.save_html(LDAvis_prepared_tf, 'lda_tf.html')
```

3 Select the best number of clusters by coherence score

We can coherence score in topic modeling to measure how interpretable the topics are to humans. We select the best number of clusters based on highest coherence score.

[ ]:
```
from gensim.models import CoherenceModel

# Compute Coherence Score
coherence_model_lda = CoherenceModel(model=lda_model, texts=texts,␣
 ↪dictionary=id2word, coherence='c_v')
coherence_lda = coherence_model_lda.get_coherence()
print('Coherence Score bag of words: ', coherence_lda)

coherence_model_lda = CoherenceModel(model=lda_model_tf, texts=texts,␣
 ↪dictionary=id2word, coherence='c_v')
coherence_lda_tf = coherence_model_lda.get_coherence()
print('Coherence Score tf-idf: ', coherence_lda_tf)
```

```
Coherence Score bag of words:  0.33813657622075743
Coherence Score tf-idf:  0.3884673157891732
```

[ ]:
```
num_list = list(range(2,11))
coherence = {}
for num_topics in num_list:
    lda_model_tf = gensim.models.LdaMulticore(corpus=corpus_tfidf,
                                   id2word=id2word,
                                   num_topics=num_topics)
    coherence_model_lda = CoherenceModel(model=lda_model_tf, texts=texts,␣
 ↪dictionary=id2word, coherence='c_v')
    coherence_lda = coherence_model_lda.get_coherence()
    coherence[num_topics] = coherence_lda
```

[ ]:
```
coherence
```

[ ]:
```
{2: 0.29721896496192934,
 3: 0.31396921106978004,
 4: 0.34535485103863095,
 5: 0.2559763428291707,
 6: 0.3195102597091959,
 7: 0.3465651203928427,
 8: 0.3767523695290471,
 9: 0.3953096197373716,
 10: 0.38321805318943086}
```

The next can be tuning the hyperparameters for alpha and beta. - Evaluate Topic Models: Latent Dirichlet Allocation (LDA)

[ ]:
```
n = 9

# Build LDA model on tf-idf
lda_model_tf = gensim.models.LdaMulticore(corpus=corpus_tfidf,
                                       id2word=id2word,
                                       num_topics= n)
# Print the Keyword in the 10 topics
pprint(lda_model_tf.print_topics())
```

```
[(0,
  '0.004*"data" + 0.004*"use" + 0.004*"studi" + 0.003*"would" + 0.003*"think" '
  '+ 0.003*"make" + 0.003*"one" + 0.003*"differ" + 0.003*"bias" + '
  '0.003*"audienc"'),
 (1,
  '0.004*"art" + 0.003*"music" + 0.003*"moral" + 0.002*"use" + 0.002*"cultur" '
  '+ 0.002*"would" + 0.002*"peopl" + 0.002*"think" + 0.002*"differ" + '
  '0.002*"one"'),
 (2,
  '0.004*"variabl" + 0.004*"attent" + 0.004*"memori" + 0.004*"line" + '
  '0.004*"data" + 0.003*"slope" + 0.003*"regress" + 0.003*"would" + '
  '0.003*"process" + 0.003*"correl"'),
 (3,
  '0.006*"system" + 0.005*"problem" + 0.004*"model" + 0.003*"level" + '
  '0.003*"differ" + 0.003*"agent" + 0.003*"use" + 0.003*"interact" + '
  '0.003*"complex" + 0.003*"solut"'),
 (4,
  '0.013*"x" + 0.009*"n" + 0.007*"p" + 0.005*"b" + 0.005*"valu" + '
  '0.005*"function" + 0.005*"number" + 0.004*"tree" + 0.004*"f" + '
  '0.004*"probabl"'),
 (5,
  '0.006*"student" + 0.005*"poll" + 0.004*"complet" + 0.004*"present" + '
  '0.003*"peopl" + 0.003*"would" + 0.003*"countri" + 0.002*"differ" + '
  '0.002*"becaus" + 0.002*"use"'),
 (6,
  '0.004*"peopl" + 0.003*"leader" + 0.003*"would" + 0.003*"one" + '
  '0.003*"state" + 0.003*"power" + 0.002*"system" + 0.002*"individu" + '
  '0.002*"becaus" + 0.002*"think"'),
 (7,
  '0.005*"thesi" + 0.005*"argument" + 0.004*"evid" + 0.004*"use" + 0.003*"flu" '
  '+ 0.003*"clone" + 0.003*"would" + 0.002*"make" + 0.002*"effect" + '
  '0.002*"becaus"'),
 (8,
  '0.007*"compani" + 0.006*"market" + 0.005*"product" + 0.004*"custom" + '
  '0.004*"risk" + 0.004*"valu" + 0.004*"cost" + 0.004*"would" + 0.004*"price" '
  '+ 0.003*"rate"')]
```

[ ]:
```
#print the top 20 words in each topics
topics_matrix = lda_model_tf.show_topics(formatted=False, num_words=20)
topics_matrix = np.array(topics_matrix)
topic_words = topics_matrix[:,1]

for i in topic_words:
    print([str(word[0]) for word in i])
    print()
```

```
['data', 'use', 'studi', 'would', 'think', 'make', 'one', 'differ', 'bias',
'audienc', 'hypothesi', 'research', 'could', 'understand', 'googl', 'design',
'variabl', 'effect', 'becaus', 'test']

['art', 'music', 'moral', 'use', 'cultur', 'would', 'peopl', 'think', 'differ',
'one', 'becaus', 'also', 'way', 'like', 'work', 'make', 'artist', 'understand',
'context', 'could']

['variabl', 'attent', 'memori', 'line', 'data', 'slope', 'regress', 'would',
'process', 'correl', 'use', 'brain', 'r', 'model', 'temperatur', 'co',
'increas', 'one', 'becaus', 'differ']

['system', 'problem', 'model', 'level', 'differ', 'agent', 'use', 'interact',
'complex', 'solut', 'would', 'one', 'emerg', 'solv', 'properti', 'becaus',
'understand', 'think', 'could', 'state']

['x', 'n', 'p', 'b', 'valu', 'function', 'number', 'tree', 'f', 'probabl',
'distribut', 'algorithm', 'vector', 'c', 'node', 'would', 'matrix', 'z', 'use',
'time']

['student', 'poll', 'complet', 'present', 'peopl', 'would', 'countri', 'differ',
'becaus', 'use', 'think', 'one', 'social', 'chang', 'econom', 'like', 'could',
'class', 'effect', 'exampl']

['peopl', 'leader', 'would', 'one', 'state', 'power', 'system', 'individu',
'becaus', 'think', 'level', 'differ', 'polit', 'group', 'govern', 'social',
'chang', 'make', 'exampl', 'could']

['thesi', 'argument', 'evid', 'use', 'flu', 'clone', 'would', 'make', 'effect',
'becaus', 'word', 'one', 'exampl', 'organ', 'think', 'support', 'statement',
'claim', 'c', 'reader']

['compani', 'market', 'product', 'custom', 'risk', 'valu', 'cost', 'would',
'price', 'rate', 'busi', 'invest', 'increas', 'financi', 'strategi', 'growth',
'differ', 'becaus', 'need', 'brand']
```

```
/var/folders/0h/xyv81g2n7sj6zr0c9cw30gkc0000gn/T/ipykernel_13803/697313039.py:3:
VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences
(which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths
or shapes) is deprecated. If you meant to do this, you must specify
'dtype=object' when creating the ndarray.
  topics_matrix = np.array(topics_matrix)
```

[ ]:
```
LDAvis_prepared_tf = gensimvis.prepare(lda_model_tf, corpus_tfidf, id2word)
```

```
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/pyLDAvis/_prepare.py:246: FutureWarning: In a future version of pandas
all arguments of DataFrame.drop except for the argument 'labels' will be
keyword-only
  default_term_info = default_term_info.sort_values(
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/past/builtins/misc.py:45: DeprecationWarning: the imp module is
deprecated in favour of importlib; see the module's documentation for
alternative uses
  from imp import reload
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/past/builtins/misc.py:45: DeprecationWarning: the imp module is
deprecated in favour of importlib; see the module's documentation for
alternative uses
  from imp import reload
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/past/builtins/misc.py:45: DeprecationWarning: the imp module is
deprecated in favour of importlib; see the module's documentation for
alternative uses
  from imp import reload
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/past/builtins/misc.py:45: DeprecationWarning: the imp module is
deprecated in favour of importlib; see the module's documentation for
alternative uses
  from imp import reload
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/past/builtins/misc.py:45: DeprecationWarning: the imp module is
deprecated in favour of importlib; see the module's documentation for
alternative uses
  from imp import reload
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/past/builtins/misc.py:45: DeprecationWarning: the imp module is
deprecated in favour of importlib; see the module's documentation for
alternative uses
  from imp import reload
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/past/builtins/misc.py:45: DeprecationWarning: the imp module is
```

```
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/joblib/backports.py:36: DeprecationWarning: distutils Version classes
are deprecated. Use packaging.version instead.
  if LooseVersion(np.__version__) < '1.13':
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/setuptools/_distutils/version.py:351: DeprecationWarning: distutils
Version classes are deprecated. Use packaging.version instead.
  other = LooseVersion(other)
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/joblib/backports.py:36: DeprecationWarning: distutils Version classes
are deprecated. Use packaging.version instead.
  if LooseVersion(np.__version__) < '1.13':
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/setuptools/_distutils/version.py:351: DeprecationWarning: distutils
Version classes are deprecated. Use packaging.version instead.
  other = LooseVersion(other)
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/joblib/backports.py:36: DeprecationWarning: distutils Version classes
are deprecated. Use packaging.version instead.
  if LooseVersion(np.__version__) < '1.13':
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/setuptools/_distutils/version.py:351: DeprecationWarning: distutils
Version classes are deprecated. Use packaging.version instead.
  other = LooseVersion(other)
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/joblib/backports.py:36: DeprecationWarning: distutils Version classes
are deprecated. Use packaging.version instead.
  if LooseVersion(np.__version__) < '1.13':
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/setuptools/_distutils/version.py:351: DeprecationWarning: distutils
Version classes are deprecated. Use packaging.version instead.
  other = LooseVersion(other)
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/joblib/backports.py:36: DeprecationWarning: distutils Version classes
are deprecated. Use packaging.version instead.
  if LooseVersion(np.__version__) < '1.13':
/Users/swimmingcircle/Library/Python/3.9/lib/python/site-
packages/setuptools/_distutils/version.py:351: DeprecationWarning: distutils
Version classes are deprecated. Use packaging.version instead.
  other = LooseVersion(other)
```

[ ]:
```
pyLDAvis.display(LDAvis_prepared_tf)
pyLDAvis.save_html(LDAvis_prepared_tf, '9_lda_tf.html')
```