LDA Topic Modelling

January 3, 2022

Topic Modelling using Latent Dirichlet Allocation

This notebook contains the approach for topic modelling for the BigIntel project by using LDA i.e. Latent Dirichlet Allocation technique. LDA's approach to topic modelling is it considers each document as a collection of topics in a certain proportion. And each topic as a collection of keywords, again, in a certain proportion.

We can describe the generative process of LDA as, given the M number of documents, N number of words, and prior K number of topics, the model trains to output:

- psi, the distribution of words for each topic K
- phi, the distribution of topics for each document i

Parameters of LDA - Alpha 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. - Beta parameter 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.

This project/notebook consists of several Tasks.

- **Task 1**: Installing all dependencies for our environment.
- **Task 2**: Importing the required libraries in the environment.
- **Task 3**: Exploratory Data Analysis
- **Task 4**: Data Analysis and Pre-processing of keywords by one-hot-encoding.
- **Task 5**: Data Preprocessing
- **Task 6**: Creating the Dictionary and Corpus needed for Topic Modelling
- **Task 7**: Building the Topic Model
- **Task 8**: Analysis top n keywords in each topic

0.0.1 Task 1: Installing all dependencies for our environment.

[3]:

```
import sys
!{sys.executable} -m pip install spacy
!{sys.executable} -m spacy download en
```

```
Requirement already satisfied (use --upgrade to upgrade): spacy in
/opt/conda/lib/python3.7/site-packages
Requirement already satisfied (use --upgrade to upgrade): thinc<8.1.0,>=8.0.9 in
/opt/conda/lib/python3.7/site-packages (from spacy)
Requirement already satisfied (use --upgrade to upgrade): typing-
```

```
extensions<4.0.0.0,>=3.7.4 in /opt/conda/lib/python3.7/site-packages (from
spacy)
Requirement already satisfied (use --upgrade to upgrade): jinja2 in
/opt/conda/lib/python3.7/site-packages (from spacy)
Requirement already satisfied (use --upgrade to upgrade):
pydantic!=1.8,!=1.8.1,<1.9.0,>=1.7.4 in /opt/conda/lib/python3.7/site-packages
(from spacy)
Requirement already satisfied (use --upgrade to upgrade):
catalogue<2.1.0,>=2.0.6 in /opt/conda/lib/python3.7/site-packages (from spacy)
Requirement already satisfied (use --upgrade to upgrade): wasabi<1.1.0,>=0.8.1
in /opt/conda/lib/python3.7/site-packages (from spacy)
Requirement already satisfied (use --upgrade to upgrade):
requests<3.0.0,>=2.13.0 in /opt/conda/lib/python3.7/site-packages (from spacy)
Requirement already satisfied (use --upgrade to upgrade): blis<0.8.0,>=0.4.0 in
/opt/conda/lib/python3.7/site-packages (from spacy)
Requirement already satisfied (use --upgrade to upgrade): cymem<2.1.0,>=2.0.2 in
/opt/conda/lib/python3.7/site-packages (from spacy)
Requirement already satisfied (use --upgrade to upgrade): spacy-
legacy<3.1.0,>=3.0.8 in /opt/conda/lib/python3.7/site-packages (from spacy)
Requirement already satisfied (use --upgrade to upgrade): preshed<3.1.0,>=3.0.2
in /opt/conda/lib/python3.7/site-packages (from spacy)
Requirement already satisfied (use --upgrade to upgrade): tqdm<5.0.0,>=4.38.0 in
/opt/conda/lib/python3.7/site-packages (from spacy)
```

```
Exception:
Traceback (most recent call last):
  File "/opt/conda/lib/python3.7/site-
packages/pip/_vendor/pkg_resources/__init__.py", line 2851, in _dep_map
    return self.__dep_map
  File "/opt/conda/lib/python3.7/site-
packages/pip/_vendor/pkg_resources/__init__.py", line 2685, in __getattr__
    raise AttributeError(attr)
AttributeError: _DistInfoDistribution__dep_map
During handling of the above exception, another exception occurred:
Traceback (most recent call last):
  File "/opt/conda/lib/python3.7/site-packages/pip/basecommand.py", line 209, in
main
    status = self.run(options, args)
  File "/opt/conda/lib/python3.7/site-packages/pip/commands/install.py", line
310, in run
    wb.build(autobuilding=True)
  File "/opt/conda/lib/python3.7/site-packages/pip/wheel.py", line 748, in build
    self.requirement_set.prepare_files(self.finder)
  File "/opt/conda/lib/python3.7/site-packages/pip/req/req_set.py", line 360, in
prepare_files
    ignore_dependencies=self.ignore_dependencies))
  File "/opt/conda/lib/python3.7/site-packages/pip/req/req_set.py", line 647, in
_prepare_file
    set(req_to_install.extras) - set(dist.extras)
  File "/opt/conda/lib/python3.7/site-
packages/pip/_vendor/pkg_resources/__init__.py", line 2810, in extras
    return [dep for dep in self._dep_map if dep]
  File "/opt/conda/lib/python3.7/site-
packages/pip/_vendor/pkg_resources/__init__.py", line 2853, in _dep_map
    self.__dep_map = self._compute_dependencies()
  File "/opt/conda/lib/python3.7/site-
packages/pip/_vendor/pkg_resources/__init__.py", line 2886, in
_compute_dependencies
    common = frozenset(reqs_for_extra(None))
  File "/opt/conda/lib/python3.7/site-
```

```
You are using pip version 8.1.1, however version 21.3.1 is available.

You should consider upgrading via the 'pip install --upgrade pip' command.
2021-12-13 10:57:28.378578: W
tensorflow/stream_executor/platform/default/dso_loader.cc:59] Could not load
dynamic library 'libcudart.so.10.1'; dlerror: libcudart.so.10.1: cannot open
shared object file: No such file or directory
2021-12-13 10:57:28.378638: I tensorflow/stream_executor/cuda/cudart_stub.cc:29]
Ignore above cudart dlerror if you do not have a GPU set up on your machine.
 As of spaCy v3.0, shortcuts like 'en' are deprecated. Please use the

full pipeline package name 'en_core_web_sm' instead.
Requirement already satisfied (use --upgrade to upgrade): en-core-web-sm==3.1.0
from https://github.com/explosion/spacy-
models/releases/download/en_core_web_sm-3.1.0/en_core_web_sm-3.1.0-py3-none-
any.whl#egg=en_core_web_sm==3.1.0 in /opt/conda/lib/python3.7/site-packages
Requirement already satisfied (use --upgrade to upgrade): spacy<3.2.0,>=3.1.0 in
/opt/conda/lib/python3.7/site-packages (from en-core-web-sm==3.1.0)
Requirement already satisfied (use --upgrade to upgrade): typing-
extensions<4.0.0.0,>=3.7.4 in /opt/conda/lib/python3.7/site-packages (from
spacy<3.2.0,>=3.1.0->en-core-web-sm==3.1.0)
Requirement already satisfied (use --upgrade to upgrade): typer<0.5.0,>=0.3.0 in
/opt/conda/lib/python3.7/site-packages (from spacy<3.2.0,>=3.1.0->en-core-web-
sm==3.1.0)
Requirement already satisfied (use --upgrade to upgrade): pathy>=0.3.5 in
/opt/conda/lib/python3.7/site-packages (from spacy<3.2.0,>=3.1.0->en-core-web-
sm==3.1.0)
Requirement already satisfied (use --upgrade to upgrade): tqdm<5.0.0,>=4.38.0 in
/opt/conda/lib/python3.7/site-packages (from spacy<3.2.0,>=3.1.0->en-core-web-
sm==3.1.0)
```

```
Exception:
Traceback (most recent call last):
  File "/opt/conda/lib/python3.7/site-
packages/pip/_vendor/pkg_resources/__init__.py", line 2851, in _dep_map
    return self.__dep_map
  File "/opt/conda/lib/python3.7/site-
packages/pip/_vendor/pkg_resources/__init__.py", line 2685, in __getattr__
    raise AttributeError(attr)
AttributeError: _DistInfoDistribution__dep_map
During handling of the above exception, another exception occurred:
Traceback (most recent call last):
  File "/opt/conda/lib/python3.7/site-packages/pip/basecommand.py", line 209, in
main
    status = self.run(options, args)
  File "/opt/conda/lib/python3.7/site-packages/pip/commands/install.py", line
310, in run
    wb.build(autobuilding=True)
  File "/opt/conda/lib/python3.7/site-packages/pip/wheel.py", line 748, in build
    self.requirement_set.prepare_files(self.finder)
  File "/opt/conda/lib/python3.7/site-packages/pip/req/req_set.py", line 360, in
prepare_files
    ignore_dependencies=self.ignore_dependencies))
  File "/opt/conda/lib/python3.7/site-packages/pip/req/req_set.py", line 647, in
_prepare_file
    set(req_to_install.extras) - set(dist.extras)
  File "/opt/conda/lib/python3.7/site-
packages/pip/_vendor/pkg_resources/__init__.py", line 2810, in extras
    return [dep for dep in self._dep_map if dep]
  File "/opt/conda/lib/python3.7/site-
packages/pip/_vendor/pkg_resources/__init__.py", line 2853, in _dep_map
    self.__dep_map = self._compute_dependencies()
  File "/opt/conda/lib/python3.7/site-
packages/pip/_vendor/pkg_resources/__init__.py", line 2886, in
_compute_dependencies
    common = frozenset(reqs_for_extra(None))
  File "/opt/conda/lib/python3.7/site-
```

```
You are using pip version 8.1.1, however version 21.3.1 is available.

You should consider upgrading via the 'pip install --upgrade pip' command.
```

[4]:

```
pip install gensim
```

```
Requirement already satisfied (use --upgrade to upgrade): gensim in
/opt/conda/lib/python3.7/site-packages
Requirement already satisfied (use --upgrade to upgrade): scipy>=0.18.1 in
/opt/conda/lib/python3.7/site-packages (from gensim)
Requirement already satisfied (use --upgrade to upgrade): six>=1.5.0 in
/opt/conda/lib/python3.7/site-packages (from gensim)
Requirement already satisfied (use --upgrade to upgrade): smart-open>=1.8.1 in
/opt/conda/lib/python3.7/site-packages (from gensim)
Requirement already satisfied (use --upgrade to upgrade): numpy>=1.11.3 in
/opt/conda/lib/python3.7/site-packages (from gensim)
You are using pip version 8.1.1, however version 21.3.1 is available.

You should consider upgrading via the 'pip install --upgrade pip' command.
Note: you may need to restart the kernel to use updated packages.
```

[5]:

```
pip install bokeh
```

```
Requirement already satisfied (use --upgrade to upgrade): bokeh in
/opt/conda/lib/python3.7/site-packages
Requirement already satisfied (use --upgrade to upgrade): packaging>=16.8 in
/opt/conda/lib/python3.7/site-packages (from bokeh)
Requirement already satisfied (use --upgrade to upgrade): typing-
extensions>=3.10.0 in /opt/conda/lib/python3.7/site-packages (from bokeh)
Requirement already satisfied (use --upgrade to upgrade): numpy>=1.11.3 in
/opt/conda/lib/python3.7/site-packages (from bokeh)
Requirement already satisfied (use --upgrade to upgrade): PyYAML>=3.10 in
/opt/conda/lib/python3.7/site-packages (from bokeh)
Requirement already satisfied (use --upgrade to upgrade): tornado>=5.1 in
/opt/conda/lib/python3.7/site-packages (from bokeh)
Requirement already satisfied (use --upgrade to upgrade): pillow>=7.1.0 in
/opt/conda/lib/python3.7/site-packages (from bokeh)
Requirement already satisfied (use --upgrade to upgrade): Jinja2>=2.9 in
/opt/conda/lib/python3.7/site-packages (from bokeh)
Requirement already satisfied (use --upgrade to upgrade): pyparsing>=2.0.2 in
/opt/conda/lib/python3.7/site-packages (from packaging>=16.8->bokeh)
Requirement already satisfied (use --upgrade to upgrade): MarkupSafe>=2.0 in
/opt/conda/lib/python3.7/site-packages (from Jinja2>=2.9->bokeh)
You are using pip version 8.1.1, however version 21.3.1 is available.

You should consider upgrading via the 'pip install --upgrade pip' command.
Note: you may need to restart the kernel to use updated packages.
```

[6]:

```
pip install pyLDAvis
```

```
Requirement already satisfied (use --upgrade to upgrade): pyLDAvis in
/opt/conda/lib/python3.7/site-packages
Requirement already satisfied (use --upgrade to upgrade): scikit-learn in
/opt/conda/lib/python3.7/site-packages (from pyLDAvis)
Requirement already satisfied (use --upgrade to upgrade): funcy in
/opt/conda/lib/python3.7/site-packages (from pyLDAvis)
Requirement already satisfied (use --upgrade to upgrade): scipy in
/opt/conda/lib/python3.7/site-packages (from pyLDAvis)
Requirement already satisfied (use --upgrade to upgrade): gensim in
/opt/conda/lib/python3.7/site-packages (from pyLDAvis)
Requirement already satisfied (use --upgrade to upgrade): numexpr in
/opt/conda/lib/python3.7/site-packages (from pyLDAvis)
Requirement already satisfied (use --upgrade to upgrade): sklearn in
/opt/conda/lib/python3.7/site-packages (from pyLDAvis)
Requirement already satisfied (use --upgrade to upgrade): joblib in
/opt/conda/lib/python3.7/site-packages (from pyLDAvis)
Requirement already satisfied (use --upgrade to upgrade): pandas>=1.2.0 in
/opt/conda/lib/python3.7/site-packages (from pyLDAvis)
Requirement already satisfied (use --upgrade to upgrade): future in
/opt/conda/lib/python3.7/site-packages (from pyLDAvis)
Requirement already satisfied (use --upgrade to upgrade): numpy>=1.20.0 in
/opt/conda/lib/python3.7/site-packages (from pyLDAvis)
Requirement already satisfied (use --upgrade to upgrade): setuptools in
/opt/conda/lib/python3.7/site-packages (from pyLDAvis)
Requirement already satisfied (use --upgrade to upgrade): jinja2 in
/opt/conda/lib/python3.7/site-packages (from pyLDAvis)
Requirement already satisfied (use --upgrade to upgrade): threadpoolctl>=2.0.0
in /opt/conda/lib/python3.7/site-packages (from scikit-learn->pyLDAvis)
Requirement already satisfied (use --upgrade to upgrade): six>=1.5.0 in
/opt/conda/lib/python3.7/site-packages (from gensim->pyLDAvis)
Requirement already satisfied (use --upgrade to upgrade): smart-open>=1.8.1 in
/opt/conda/lib/python3.7/site-packages (from gensim->pyLDAvis)
Requirement already satisfied (use --upgrade to upgrade): python-dateutil>=2.7.3
in /opt/conda/lib/python3.7/site-packages (from pandas>=1.2.0->pyLDAvis)
Requirement already satisfied (use --upgrade to upgrade): pytz>=2017.3 in
/opt/conda/lib/python3.7/site-packages (from pandas>=1.2.0->pyLDAvis)
Requirement already satisfied (use --upgrade to upgrade): MarkupSafe>=2.0 in
/opt/conda/lib/python3.7/site-packages (from jinja2->pyLDAvis)
You are using pip version 8.1.1, however version 21.3.1 is available.

You should consider upgrading via the 'pip install --upgrade pip' command.
Note: you may need to restart the kernel to use updated packages.
```

0.0.2 Task 2: Importing the required libraries in the environment.

[7]:

```
#Importing the necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
#tqdm module helps to create progress bars to track how long your code is␣
 ↪taking to process
from tqdm import tqdm
#pprint is to make our topics formatted a little nicer when we take a look
from pprint import pprint
from collections import Counter

import nltk
from nltk.corpus import stopwords

#gensim
import gensim
import gensim.corpora as corpora
from gensim.utils import simple_preprocess
from gensim.models import CoherenceModel, LdaModel

#spacy for lemmatization
import spacy

from sklearn.feature_extraction.text import CountVectorizer
import scipy.stats as stats

#For plotting clustering graph
from bokeh.plotting import figure, output_file, show
from bokeh.models import Label

#LDA and LSA\
from sklearn.decomposition import TruncatedSVD
from sklearn.decomposition import LatentDirichletAllocation
from sklearn.manifold import TSNE

#plotting tools
#import pyLDAvis
import pyLDAvis.gensim_models
pyLDAvis.enable_notebook()
#import pyLDAvis.gensim


# Enable logging for gensim - optional
import logging
```

```
logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s',␣
 ↪level=logging.ERROR)

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

```
2021-12-13 10:57:40.985104: W
tensorflow/stream_executor/platform/default/dso_loader.cc:59] Could not load
dynamic library 'libcudart.so.10.1'; dlerror: libcudart.so.10.1: cannot open
shared object file: No such file or directory
2021-12-13 10:57:40.985204: I tensorflow/stream_executor/cuda/cudart_stub.cc:29]
Ignore above cudart dlerror if you do not have a GPU set up on your machine.
```

- (t-SNE) t-Distributed Stochastic Neighbor Embedding is a non-linear dimensionality reduction algorithm used for exploring high-dimensional data. It maps multi-dimensional data to two or more dimensions suitable for human observation.

0.0.3 Task 3: Exploratory Data Analysis

[ ]:

```
unidf = pd.read_csv(r'df.csv',delimiter = '\t')
unidf.head()
```

[ ]:

```
unidf.info()
```

[ ]:

```
# Define helper functions
def get_top_n_words(n_top_words, count_vectorizer, text_data):
    '''
    returns a tuple of the top n words in a sample and their
    accompanying counts, given a CountVectorizer object and text sample
    '''
    vectorized_headlines = count_vectorizer.fit_transform(text_data.values)
    vectorized_total = np.sum(vectorized_headlines, axis=0)
    word_indices = np.flip(np.argsort(vectorized_total)[0,:], 1)
    word_values = np.flip(np.sort(vectorized_total)[0,:],1)

    word_vectors = np.zeros((n_top_words, vectorized_headlines.shape[1]))
    for i in range(n_top_words):
        word_vectors[i,word_indices[0,i]] = 1

    words = [word[0].encode('ascii').decode('utf-8') for
             word in count_vectorizer.inverse_transform(word_vectors)]

    return (words, word_values[0,:n_top_words].tolist()[0])
```

[ ]:

```
#Initializing the count vector
from sklearn.feature_extraction.text import CountVectorizer
count_vectorizer = CountVectorizer(stop_words = 'english')
```

```
keyword, keyword_val = get_top_n_words(n_top_words = 15,
                                      count_vectorizer = count_vectorizer,
                                      text_data = df.Keyword)

fig,ax = plt.subplots(figsize=(10,5))
ax.bar(range(len(keyword)), keyword_val)
ax.set_xticks(range(len(keyword)))
ax.set_xticklabels(keyword, rotation = 'vertical')
ax.set_title("Top Keywords in the df (excluding the stop words)")
ax.set_xlabel("Keywords")
ax.set_ylabel("Number of Occurences");
```

Next, we generate the hist of keyword word length, and use part-of-speech tagging to understand the types of keywords used.

[ ]:

```
#Initializing the count vector
count_vectorizer = CountVectorizer(stop_words = 'english')
keyword, keyword_val = get_top_n_words(n_top_words = 14,
                                      count_vectorizer = count_vectorizer,
                                      text_data = df.pos)

fig,ax = plt.subplots(figsize=(10,5))
ax.bar(range(len(keyword)), keyword_val)
ax.set_xticks(range(len(keyword)))
ax.set_xticklabels(keyword, rotation = 'vertical')
ax.set_title("Part-of-Speech Tagging for Keywords")
ax.set_xlabel("Type of Keywords")
ax.set_ylabel("Number of Occurences");
```

[ ]:

```
keyword_len = []

for index, row in unidf.iterrows():
    #print(len(row['keyword'].split()))
    keyword_len.append(len(row['keyword'].split()))

print(f'Average number of words in the keyword are: {np.mean(keyword_len)}' )
```

[ ]:

```
import scipy.stats as stats
y = stats.norm.pdf(np.linspace(1,10,50), np.mean(keyword_len), np.
 ↪std(keyword_len))

plt.hist(keyword_len, bins= range(1,10), density = True)
plt.plot(np.linspace(0,14,50), y, linewidth = 1)
plt.title("Keyword length")
plt.xlabel("Number of words")
plt.ylabel("Probability");
```

0.0.4 Task 5: Data Preprocessing

We use NLTK's wordnet to find the - meanings of words, synonymns, antonyms and more. - We use WordNetLemmatizer to get the root word. Filter out the stop words.

[ ]:
```
nltk.download('stopwords')
```

[ ]:
```
stop_words = stopwords.words('english')
#increasing the stopword list
stop_words.extend(['from','subject','re','edu','use'])
print(len(stop_words))
```

0.0.5 Data Preparation

1. `Lemmatization`: Lemmatization is nothing but converting a word to its root word. For example: the lemma of the word 'machines' is 'machine'. Likewise, 'walking' –> 'walk', 'mice' –> 'mouse' and so on.
2. `Tokenize and Clean-up`: Tokenizing each sentence into list of words, removing punctuations and unnecessary characters altogether. By using Gensim's simple_preprocess()
3. `Bigram and Trigram Models`: Bigrams are two words frequently occuring together in a document (eg, social media, where these two words are more likely to occur together then separately). We want to identify these so we can concatenate them and consider them as one word. We use **Pointwise Mutual Information** score to identify significant bigrams and trigrams to concatenate. We also filter them with the filter (noun/adj) (pos), because these are common structures pointing out noun-type n-grams. This helps LDA model better cluster topics.

[ ]:
```
#transforming all of the keywords into list
data = df.Keyword.values.tolist()

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

data_words = list(sent_to_words(data))

print(data_words[:1])
```

`models.phrases` automatically detects common phrases - aka the multi-word expressions, word n-gram collocations - from a stream of sentences. - An N-gram means a sequence of N words. So for example, "Medium blog" is a 2-gram (a bigram), "A Medium blog post" is a 4-gram, and "Write on Medium" is a 3-gram (trigram). 1. **min_count**: ignores all words and bigrams with total collected count lower than this. bydefault=5. 2. **threshold**: represents a threshold for forming the phrases (higher means fewer phrases)

[ ]:
```
# Build the bigram and trigram models
```

```
bigram = gensim.models.Phrases(data_words, min_count=1, threshold=2) # higher␣
 ↪threshold fewer phrases.
trigram = gensim.models.Phrases(bigram[data_words], threshold=100)

# Faster way to get a sentence clubbed as a trigram/bigram
bigram_mod = gensim.models.phrases.Phraser(bigram)
trigram_mod = gensim.models.phrases.Phraser(trigram)

# See trigram example
print(trigram_mod[bigram_mod[data_words[0]]])
```

[ ]:

```
def remove_stopwords(keywords):
    #This will remove stopwords and punctuation.
    return [[word for word in simple_preprocess(str(doc)) if word not in␣
 ↪stop_words] for doc in keywords]

def make_bigram(keywords):
    return [bigram_mod[doc] for doc in keywords]

def make_trigrams(keywords):
    return [trigram_mod[trigram_mod[doc]] for doc in keywords]

#reduces the different forms of the word to it's initial form (runs, ran to run)
def lemmatization(keywords, allowed_postags=['NOUN','ADJ','VERB','ADV']):
    texts_out = []
    for sent in keywords:
        doc = nlp(" ".join(sent))
        texts_out.append([token.lemma_ for token in doc if token.pos_ in␣
 ↪allowed_postags])
    return texts_out
```

The default spaCy pipeline is laid out like this: * `Tokenizer`: Breaks the full text into individual tokens. * `Tagger`: Tags each token with the part of speech. * `Parser`: Parses into noun chunks, amongst other things. * `Named Entity Recognizer (NER)`: Labels named entities, like U.S.A.

[ ]:

```
#Remove stop words
data_words_nostops = remove_stopwords(data_words)

#Make Bigrams
data_words_bigram = make_bigram(data_words_nostops)

#Initializing spacy -en- model, keepng only tagger component
nlp = spacy.load('en_core_web_sm', disable =['parser','ner'])

#Do lemmatization keeping only noun, adj, vb, adv
data_lem = lemmatization(data_words_bigram)
```

```
print(data_lem[:1])
```

0.0.6 Task 6: Creating the Dictionary and Corpus needed for Topic Modelling

Gensim creates a unique_id for each word in the document. The produced corpus shown above is the mapping of (word_id, word_frequency). For example: (0,1)= word_id 0 occurs once in the first document. These are the input labels for the LDA model. Corpus is a simple set of documents. These are the training labels.

[ ]:
```
#create dictionary
id2word = corpora.Dictionary(data_lem)

#Create corpus
keywords = data_lem

# Term Document Frequency
corpus = [id2word.doc2bow(keyword) for keyword in keywords]

#View
print(corpus[:1])
```

[ ]:
```
#seeing what word a given id corresponds to, passing the id as a key to the␣
 ↪dictionary
id2word[0]
```

[ ]:
```
#printing corpus (term-frequency)
[[(id2word[id], freq) for id,freq in cp] for cp in corpus[:2]]
```

0.0.7 Task 7: Building the Topic Model

Now that we have our corpus and dictionary, all we need is to provide the number of topics as well.

Finding the optimal number of topics for LDA The approach would be to build many LDA models with different values of number of topics (k) and pick the one that gives the highest coherence value. The function `compute_coherence_values()` trains multiple LDA models and provides the models and their corresponding coherence score.

[ ]:
```
print(gensim.__version__)
```

https://stackoverflow.com/questions/32313062/what-is-the-best-way-to-obtain-the-optimal-number-of-topics-for-a-lda-model-usin

[ ]:
```
# Considering 1-15 topics, as the last is cut off
num_topics = list(range(14)[1:])
num_keywords = 13

LDA_models = {}
LDA_topics = {}
```

```
for i in num_topics:
    LDA_models[i] = LdaModel(corpus=corpus,
                             id2word=id2word,
                             num_topics=i,
                             update_every=1,
                             chunksize=len(corpus),
                             passes=20,
                             alpha='auto',
                             random_state=42)

    shown_topics = LDA_models[i].show_topics(num_topics=i,
                                             num_words=num_keywords,
                                             formatted=False)
    LDA_topics[i] = [[word[0] for word in topic[1]] for topic in shown_topics]
```

```
[ ]: def jaccard_similarity(topic_1, topic_2):
         """
         Derives the Jaccard similarity of two topics

         Jaccard similarity:
         - A statistic used for comparing the similarity and diversity of sample sets
         - J(A,B) = (A   B)/(A   B)
         - Goal is low Jaccard scores for coverage of the diverse elements
         """
         intersection = set(topic_1).intersection(set(topic_2))
         union = set(topic_1).union(set(topic_2))

         return float(len(intersection))/float(len(union))
```

```
[ ]: LDA_stability = {}
     for i in range(0, len(num_topics)-1):
         jaccard_sims = []
         for t1, topic1 in enumerate(LDA_topics[num_topics[i]]): # pylint:␣
      ↪disable=unused-variable
             sims = []
             for t2, topic2 in enumerate(LDA_topics[num_topics[i+1]]): # pylint:␣
      ↪disable=unused-variable
                 sims.append(jaccard_similarity(topic1, topic2))

             jaccard_sims.append(sims)

         LDA_stability[num_topics[i]] = jaccard_sims

     mean_stabilities = [np.array(LDA_stability[i]).mean() for i in num_topics[:-1]]
```

[ ]:

```
coherences = [CoherenceModel(model=LDA_models[i], texts=data_lem,␣
 ↪dictionary=id2word, coherence='c_v').get_coherence()\
              for i in num_topics[:-1]]
```

[ ]:

```
coh_sta_diffs = [coherences[i] - mean_stabilities[i] for i in␣
 ↪range(num_keywords)[:-1]] # limit topic numbers to the number of keywords
coh_sta_max = max(coh_sta_diffs)
coh_sta_max_idxs = [i for i, j in enumerate(coh_sta_diffs) if j == coh_sta_max]
ideal_topic_num_index = coh_sta_max_idxs[0] # choose less topics in case␣
 ↪there's more than one max
ideal_topic_num = num_topics[ideal_topic_num_index]
```

[ ]:

```
coherences
```

[ ]:

```
plt.figure(figsize=(10,5))
ax = sns.lineplot(x=num_topics[:-1], y=mean_stabilities, label='Average Topic␣
 ↪Overlap')
ax = sns.lineplot(x=num_topics[:-1], y=coherences, label='Topic Coherence')

ax.axvline(x=ideal_topic_num, label='Ideal Number of Topics', color='black')
ax.axvspan(xmin=ideal_topic_num - 1, xmax=ideal_topic_num + 1, alpha=0.5,␣
 ↪facecolor='grey')

y_max = max(max(mean_stabilities), max(coherences)) + (0.10 *␣
 ↪max(max(mean_stabilities), max(coherences)))
ax.set_ylim([0, y_max])
ax.set_xlim([1, num_topics[-1]-1])

ax.axes.set_title('Model Metrics per Number of Topics')
ax.set_ylabel('Metric Level')
ax.set_xlabel('Number of Topics')
plt.show()
```

So from the above diagram, the ideal number of topics will be 11 as it will maximize coherence and minimize the topic overlap based on the Jaccard Similarity.

[ ]:

```
lda_model = gensim.models.ldamodel.LdaModel(corpus = corpus,
                                            id2word = id2word,
                                            num_topics = 11,
                                            random_state = 42,
                                            update_every=1,
                                            chunksize = 100,
                                            passes = 10,
                                            alpha = 'auto',
                                            per_word_topics = True)
```

```
[ ]: pprint(lda_model.print_topics())
     doc_lda = lda_model[corpus]
```

```
[ ]: ### Compute Model Perplexity and Coherence Score
```

```
[ ]: #A measure of how good the model is. The lower the better
     print('\nPerplexity: ', lda_model.log_perplexity(corpus))

     #Compute Coherence Score
     coherence_model_lda = CoherenceModel(model=lda_model, texts=data_lem,␣
      ↪dictionary=id2word, coherence='c_v')
     coherence_lda = coherence_model_lda.get_coherence()
     print('\nCoherence Score: ', coherence_lda)
```

```
[ ]: #Visualize the topics
     pyLDAvis.enable_notebook()
     vis = pyLDAvis.gensim_models.prepare(lda_model, corpus, id2word)
     vis
```

So, each bubble on the left-hand side are the topics. The larger the bubble, the more prevalent is that topic.

0.0.8 Task 8: Analysis top n keywords in each topic

```
[ ]: all_topics = {}
     num_terms = 10
     lambd = 0.6
     for i in range(1,11):
         topic = vis.topic_info[vis.topic_info.Category == 'Topic'+str(i)].copy()
         topic['relevance'] = topic['loglift']*(1-lambd)+topic['logprob']*lambd
         all_topics['Topic '+str(i)] = topic.sort_values(by='relevance',␣
      ↪ascending=False).Term[:num_terms].values
```

```
[ ]: pd.DataFrame(all_topics).T
```