STUDENT REPORT

NATURAL LANGUAGE PROCESSING-TOPIC MODELING

PRIYA PADHIYAR

Table of Contents

Introduction to NLP

Natural Language Processing gives computers the ability to read, understand and interpret human languages. For computers this is a difficult task due to large amount of unstructured data. Some of the tools we are using on daily basis are built on the concept of NLP. To name a few:

- *Spell checker*
- *Auto Complete*
- *Spam filter etc.*

NLP interprets the human language using many techniques ranging from statistical and machine learning methods to rules based and algorithm approaches. Some basic NLP tasks are:

- *Part of Speech tagging*
- *Lemmatization*
- *Tokenization*
- *Disambiguation*
- *Semantic etc.*

My project in Fall 2020 focuses on **Topic Modeling** - a machine learning technique that automatically analyzes text data to determine clusters of words for a set of documents. This is known as 'unsupervised' machine learning because it doesn't require a predefined list of tags or training data that's been previously classified by humans.

What is Topic Modeling?

When a document is provided, the topic modelling algorithms will form topics from the words present in the document. Each of the algorithms does this in a different way, but the basics are that the algorithms look at the co-occurrence of words in the document and if words often appear in the same document together, then these words are likely to form a topic together. The algorithm will form topics which group commonly co-occurring words. A topic in this sense, is just list of words that often appear together and also have scores associated with each of these words in the topic. The higher the score of a word in a topic, the higher that word's importance in the topic. Each topic will have a score for every word found in the document, in order to make sense of these topics we usually only look at the top words - the words with low scores are irrelevant.

The following are key factors to obtaining good segregation topics:

1. The quality of text processing.
2. The variety of topics the text talks about.
3. The choice of topic modeling algorithm.
4. The number of topics fed to the algorithm.
5. The algorithms tuning parameters.

How to do Topic Modeling

Before we jump into the coding part of Topic Modeling, I would like to give a theoretical background on this and thus have divided the report into two parts; In the first part, I will try to explain what topic modeling is and in the second part, I will provide a python tutorial of how to do topic modeling on the real-world dataset.

Topic Modeling refers to the process of dividing a corpus of documents in two:

⇒ A list of the topics covered by the documents in the corpus.

⇒ Several sets of documents from the corpus grouped by the topics they cover.

There are several algorithms for doing topic modeling. The most popular ones are:

- LDA – Latent Dirichlet Allocation – The one I'll be focusing on in this report. Its foundations are Probabilistic Graphical Models.
- LSA or LSI – Latent Semantic Analysis or Latent Semantic Indexing – Uses Singular Value Decomposition (SVD) on the Document-Term Matrix. Based on Linear Algebra.
- NMF – Non-Negative Matrix Factorization – Based on Linear Algebra.

Latent Dirichlet Allocation (LDA)

LDA is based on the following assumptions: Distributional Hypothesis (i.e. similar topics make use of similar words) and the Statistical Mixture Hypothesis (i.e. Documents talk about several topic).

The purpose of LDA is to map each document in our corpus to a set of topics which covers a good deal of the words in the document.

What LDA does in order to map the documents to a list of topics is assign topics to arrangements of words, e.g. n-grams such as best player for a topic related to sports. This stems from the assumption that documents are written with arrangements of words and that those arrangements determine topics. LDA also ignores syntactic information and treats documents as bags of words. It also assumes that all words in the document can be assigned a probability of belonging to a topic. That said, the goal of LDA is to determine the mixture of topics that a document contains. The output of this algorithm will be a vector that reports every topic for the documents being modeled.

LDA Hyperparameters

There are 3 hyperparameters to tweak in an LDA model to get the most accurate results.

1. Alpha: It controls document level similarity. A low value of alpha will assign fewer topics to each document whereas a high value of alpha will have the opposite effect.
2. Beta: It controls topic level similarity. A low value of beta will use fewer words to model a topic whereas a high value will use more words, thus making topics more similar between them.
3. Number of Topics: Since LDA cannot decide on the number of topics by itself it is an important hyperparameter that needs to be set and decided on the accuracy of the model which is calculated using topic Coherence and Topic Perplexity.

Plate Diagram of LDA Model:

Now, that we understand what the LDA hyperparameters are and the assumptions of the model let's understand the process in detail with the help of the below diagram:

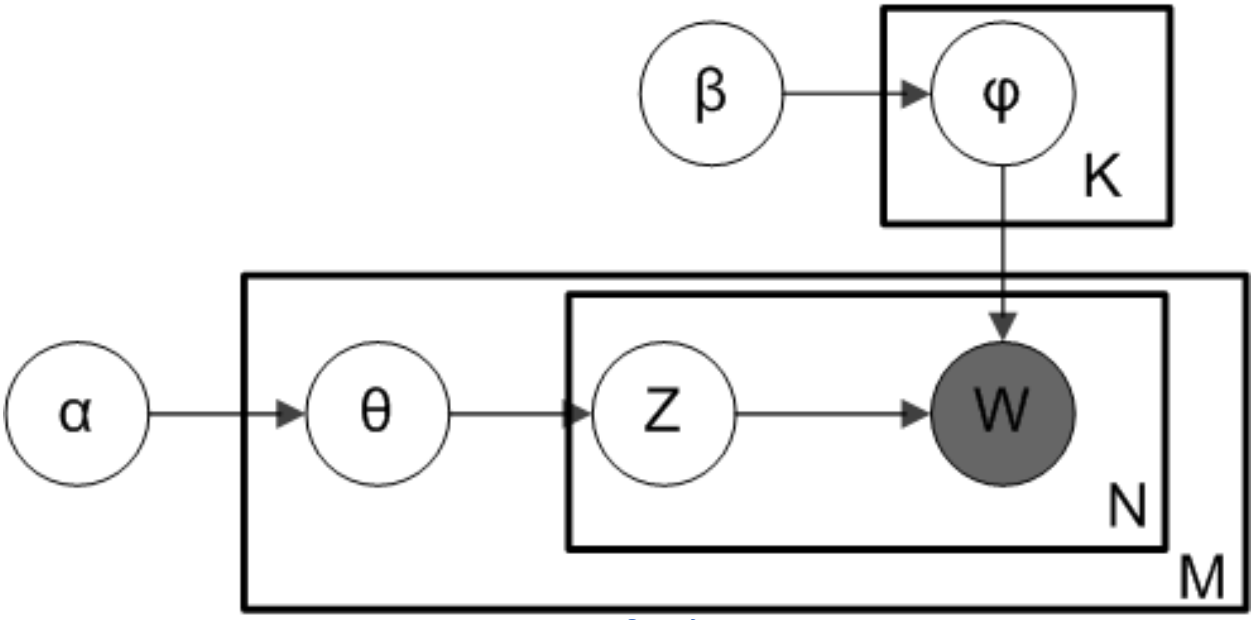


Plate Diagram of LDA Model

The description of the diagram is as below:

Description 1)

- Rectangles – Also known as plates they represent the loops of different sizes. The given loop iterates through each word in the document. The plate sizes are represented by

  K → Total number of Topics.

  N → Total number of words in a single Document.

  M → Total number of Documents.
- Circles - Represent the variables.
- Arrows – Represent the flow of the variables.

Description 2)

- Alpha and Beta are initial hyperparameters that are input by the end user which is visible in the diagram as there is no flow before them. These greatly control the results of the topic model.
- Alpha – It is a vector of size M. It contributes towards calculating theta.
- Theta – It is a distribution between documents and topics. (Total documents x Total topics)
- Beta – It is a vector of size K. It contributes towards calculating phi.
- Phi – It is a distribution between total topics and Vocabulary size i.e. the set of unique words (K x V).

In a nutshell, LDA is:

Latent → Everything that we don't know from prior.

Dirichlet → It is a distribution of distribution (i.e. distribution of topics in documents and distribution of words in the topic).

Allocation → This means that once we have Dirichlet, we will allocate topics to the documents and words of the document to topics.

How to Evaluate the Topic Model?

It is always beneficial to know if a trained model is good or bad and also be able to compare different models. In our case, Perplexity Measure and Coherence Score are the two measures we will consider in evaluating the model.

I. *Perplexity measure* captures how surprised a model is of new data it has not seen before and is measured as the normalized log-likelihood of a held-out test set. That is to say, how well does the model represent or reproduce the statistics of the held-out data.

II. *Topic Coherence* measures score a single topic by measuring the degree of semantic similarity between high scoring words in the topic. These measurements help distinguish between topics that are semantically interpretable topics and topics that are artifacts of statistical inference.

Model Implementation – Topic Modeling with Gensim

The complete code and json file is available as a Jupyter Notebook on GitHub.

https://github.com/priyapadhiyar/Topic-Modeling

Introduction

For the Independent study, I will take an example of the '20 Newsgroups' dataset and use LDA to extract the naturally discussed topics. I will be using the Latent Dirichlet Allocation (LDA) from Gensim package as it has an excellent implementation of LDA.

I will also extract the volume and percentage contribution of each topic to get an idea of how important a topic is.

Also, highlight the topic of each word and using that determine the dominant topic of each sentence which will further be helpful in sentiment analysis.

Package requirements

You will need to pip install the following packages in order to get started:

1. pip install -U gensim
2. pip install spacy
3. pip install pyLDAvis

Additionally, the following packages need to be imported:

1. import nltk; nltk.download('stopwords')
2. import spacy (Run this in command prompt or terminal)
3. import re
4. import numpy as np
5. import pandas as pd
6. from pprint import pprint
7. import gensim
8. import gensim.corpora as corpora
9. from gensim.utils import simple_preprocess
10. from gensim.models import CoherenceModel
11. import spacy
12. import pyLDAvis
13. import pyLDAvis.gensim
14. import matplotlib.pyplot as plt
15. import logging
16. import warnings
17. from nltk.corpus import stopwords

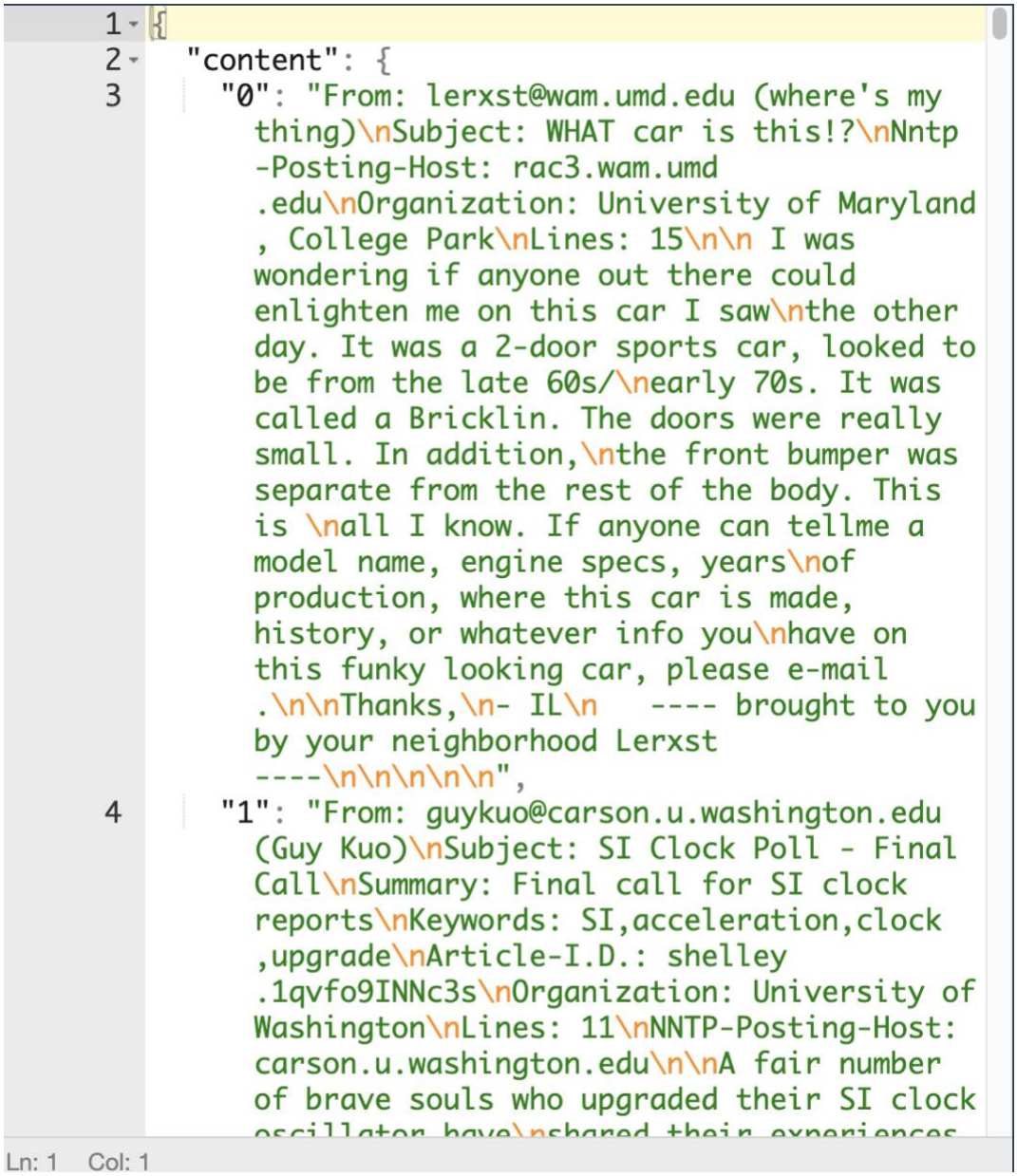
Import the dataset

Now that we have all the required packages installed and ready to use let's import the dataset and get started with Topic Modeling.

```
# Import Dataset
df = pd.read_json('https://raw.githubusercontent.com/selva86/datasets/master/newsgroups.json')
#print(df.target_names.unique())
df.head()
```

The newsgroup dataset comprises of around 11,000 newsgroups posts that we will model on and get the top Topics these posts are mostly talking upon.

	content	target	target_names
0	From: lerxst@wam.umd.edu (where's my thing)\nS...	7	rec.autos
1	From: guykuo@carson.u.washington.edu (Guy Kuo)...	4	comp.sys.mac.hardware
2	From: twillis@ec.ecn.purdue.edu (Thomas E Will...	4	comp.sys.mac.hardware
3	From: jgreen@amber (Joe Green)\nSubject: Re: W...	1	comp.graphics
4	From: jcm@head-cfa.harvard.edu (Jonathan McDow...	14	sci.space


{
"content": {
"0": "From: lerxst@wam.umd.edu (where's my thing)\nSubject: WHAT car is this!?\nNntp-Posting-Host: rac3.wam.umd.edu\nOrganization: University of Maryland, College Park\nLines: 15\n\n I was wondering if anyone out there could enlighten me on this car I saw\nthe other day. It was a 2-door sports car, looked to be from the late 60s/\nearly 70s. It was called a Bricklin. The doors were really small. In addition,\nthe front bumper was separate from the rest of the body. This is \nall I know. If anyone can tellme a model name, engine specs, years\nof production, where this car is made, history, or whatever info you\nhave on this funky looking car, please e-mail.\n\nThanks,\n- IL\n ---- brought to you by your neighborhood Lerxst ----\n\n\n\n\n",
"1": "From: guykuo@carson.u.washington.edu (Guy Kuo)\nSubject: SI Clock Poll - Final Call\nSummary: Final call for SI clock reports\nKeywords: SI,acceleration,clock,upgrade\nArticle-I.D.: shelley.1qvfo9INNc3s\nOrganization: University of Washington\nLines: 11\nNNTP-Posting-Host: carson.u.washington.edu\n\nA fair number of brave souls who upgraded their SI clock oscillator have\nshared their experiences
Ln: 1 Col: 1

Data Preprocessing

One of the key factors for obtaining a good segregation of topics is the quality of text preprocessing and hence we will start with that:

Remove Emails and single quotes using Regular Expression

As we can see from the above image about the dataset it contains Emails, new lines and single quotes which needs to be removed using regular expressions.

```
# Convert to list
data = df.content.values.tolist()

# Remove Emails
data = [re.sub('\S*@\S*\s?', '', sent) for sent in data]

# Remove new line characters
data = [re.sub('\s+', ' ', sent) for sent in data]

# Remove distracting single quotes
data = [re.sub("\'", "", sent) for sent in data]
```

After running the above code, we can notice that now the text is much cleaner and the Email Id's as well as the single quotes are removed. If you will notice after from: no more the Email Id is being displayed and I'm is → im.

```
1 print(df.content[2])
```

```
From: twillis@ec.ecn.purdue.edu (Thomas E Willis)
Subject: PB questions...
Organization: Purdue University Engineering Computer Network
Distribution: usa
Lines: 36

well folks, my mac plus finally gave up the ghost this weekend after
starting life as a 512k way back in 1985.  sooo, i'm in the market for a
new machine a bit sooner than i intended to be...

i'm looking into picking up a powerbook 160 or maybe 180 and have a bunch
of questions that (hopefully) somebody can answer:

* does anybody know any dirt on when the next round of powerbook
introductions are expected?  i'd heard the 185c was supposed to make an
appearence "this summer" but haven't heard anymore on it - and since i
don't have access to macleak, i was wondering if anybody out there had
more info...

* has anybody heard rumors about price drops to the powerbook line like the
ones the duo's just went through recently?

* what's the impression of the display on the 180?  i could probably swing
a 180 if i got the 80Mb disk rather than the 120, but i don't really have
a feel for how much "better" the display is (yea, it looks great in the
store, but is that all "wow" or is it really that good?).  could i solicit
some opinions of people who use the 160 and 180 day-to-day on if its worth
taking the disk size and money hit to get the active display?  (i realize
this is a real subjective question, but i've only played around with the
machines in a computer store breifly and figured the opinions of somebody
who actually uses the machine daily might prove helpful).

* how well does hellcats perform?  ;)

thanks a bunch in advance for any info - if you could email, i'll post a
summary (news reading time is at a premium with finals just around the
corner    :/ )
```

Figure 1: Text before the regex preprocessing

Highlighted portions in the above picture are the parts that gets cleaned up

```
1 pprint(data[2])
```

```
('From: (Thomas E Willis) Subject: PB questions... Organization: Purdue '
 'University Engineering Computer Network Distribution: usa Lines: 36 well '
 'folks, my mac plus finally gave up the ghost this weekend after starting '
 'life as a 512k way back in 1985. sooo, im in the market for a new machine a '
 'bit sooner than i intended to be... im looking into picking up a powerbook '
 '160 or maybe 180 and have a bunch of questions that (hopefully) somebody can '
 'answer: * does anybody know any dirt on when the next round of powerbook '
 'introductions are expected? id heard the 185c was supposed to make an '
 'appearence "this summer" but havent heard anymore on it - and since i dont '
 'have access to macleak, i was wondering if anybody out there had more '
 'info... * has anybody heard rumors about price drops to the powerbook line '
 'like the ones the duos just went through recently? * whats the impression of '
 'the display on the 180? i could probably swing a 180 if i got the 80Mb disk '
 'rather than the 120, but i dont really have a feel for how much "better" the '
 'display is (yea, it looks great in the store, but is that all "wow" or is it '
 'really that good?). could i solicit some opinions of people who use the 160 '
 'and 180 day-to-day on if its worth taking the disk size and money hit to get '
 'the active display? (i realize this is a real subjective question, but ive '
 'only played around with the machines in a computer store breifly and figured '
 'the opinions of somebody who actually uses the machine daily might prove '
 'helpful). * how well does hellcats perform? ;) thanks a bunch in advance for '
 'any info - if you could email, ill post a summary (news reading time is at a '
 'premium with finals just around the corner... :( ) -- Tom Willis \\ \\ '
 'Purdue Electrical Engineering '
 '----------------------------------------------------------------------- '
 '"Convictions are more dangerous enemies of truth than lies." - F. W. '
 'Nietzsche ')
```

Figure 2: Text after the regex preprocessing

The text is still messy and needs to be cleared up before we pass it through the model. Ultimate goal is to have words and no other special characters.

Tokenize Words

Tokenization is the task of chopping a document into pieces called tokens and at the same time throwing away certain characters like punctuations, parenthesis etc. Here is an example of tokenization for our dataset.

Before Tokenization → 'From: (Thomas E Willis)

After Tokenization → 'from', 'thomas', 'willis',

```
:
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[2])
```

```
['from', 'thomas', 'willis', 'subject', 'pb', 'questions', 'organization', 'purdue', 'university', 'engineering', 'co
mputer', 'network', 'distribution', 'usa', 'lines', 'well', 'folks', 'my', 'mac', 'plus', 'finally', 'gave', 'up', 't
he', 'ghost', 'this', 'weekend', 'after', 'starting', 'life', 'as', 'way', 'back', 'in', 'sooo', 'im', 'in', 'the',
'market', 'for', 'new', 'machine', 'bit', 'sooner', 'than', 'intended', 'to', 'be', 'im', 'looking', 'into', 'pickin
g', 'up', 'powerbook', 'or', 'maybe', 'and', 'have', 'bunch', 'of', 'questions', 'that', 'hopefully', 'somebody', 'ca
n', 'answer', 'does', 'anybody', 'know', 'any', 'dirt', 'on', 'when', 'the', 'next', 'round', 'of', 'powerbook', 'int
roductions', 'are', 'expected', 'id', 'heard', 'the', 'was', 'supposed', 'to', 'make', 'an', 'appearence', 'this', 's
ummer', 'but', 'havent', 'heard', 'anymore', 'on', 'it', 'and', 'since', 'dont', 'have', 'access', 'to', 'macleak',
'was', 'wondering', 'if', 'anybody', 'out', 'there', 'had', 'more', 'info', 'has', 'anybody', 'heard', 'rumors', 'abo
ut', 'price', 'drops', 'to', 'the', 'powerbook', 'line', 'like', 'the', 'ones', 'the', 'duos', 'just', 'went', 'throu
gh', 'recently', 'whats', 'the', 'impression', 'of', 'the', 'display', 'on', 'the', 'could', 'probably', 'swing', 'i
f', 'got', 'the', 'mb', 'disk', 'rather', 'than', 'the', 'but', 'dont', 'really', 'have', 'feel', 'for', 'how', 'muc
h', 'better', 'the', 'display', 'is', 'yea', 'it', 'looks', 'great', 'in', 'the', 'store', 'but', 'is', 'that', 'al
l', 'wow', 'or', 'is', 'it', 'really', 'that', 'good', 'could', 'solicit', 'some', 'opinions', 'of', 'people', 'who',
'use', 'the', 'and', 'day', 'to', 'day', 'on', 'if', 'its', 'worth', 'taking', 'the', 'disk', 'size', 'and', 'money',
'hit', 'to', 'get', 'the', 'active', 'display', 'realize', 'this', 'is', 'real', 'subjective', 'question', 'but', 'iv
e', 'only', 'played', 'around', 'with', 'the', 'machines', 'in', 'computer', 'store', 'breifly', 'and', 'figured', 't
he', 'opinions', 'of', 'somebody', 'who', 'actually', 'uses', 'the', 'machine', 'daily', 'might', 'prove', 'helpful',
'how', 'well', 'does', 'hellcats', 'perform', 'thanks', 'bunch', 'in', 'advance', 'for', 'any', 'info', 'if', 'you',
'could', 'email', 'ill', 'post', 'summary', 'news', 'reading', 'time', 'is', 'at', 'premium', 'with', 'finals', 'jus
t', 'around', 'the', 'corner', 'tom', 'willis', 'purdue', 'electrical', 'engineering', 'convictions', 'are', 'more',
'dangerous', 'enemies', 'of', 'truth', 'than', 'lies', 'nietzsche']
```

Create Bigram and Trigrams

So far, we have removed unnecessary characters and split the document into tokens. Now, in the next step we will construct bigrams from our news posts. This part of the function will group every pair of words and put them at the end. So, the sentence

'i scream for ice cream' → 'i scream for ice cream' 'i_scream' 'scream_for' 'for_ice' 'ice_cream'.

A bigram is a word pair like i_scream or ice_cream. The reason for doing this is that when we go from sentence to vector form of the document, we will lose the information about word ordering. Therefore, we could lose 'ice cream' amongst documents about putting ice and antiseptic cream on a wound (for example).

```
# Build the bigram and trigram models
bigram = gensim.models.Phrases(data_words, min_count=5, threshold=100) # 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[2]]])


```

```
['from', 'thomas', 'willis', 'subject', 'pb', 'questions', 'organization', 'purdue_university_engineering', 'compute
r', 'network', 'distribution_usa', 'lines', 'well', 'folks', 'my', 'mac', 'plus', 'finally', 'gave', 'up', 'the', 'gh
ost', 'this', 'weekend', 'after', 'starting', 'life', 'as', 'way', 'back', 'in', 'sooo', 'im', 'in', 'the', 'market',
'for', 'new', 'machine', 'bit', 'sooner', 'than', 'intended', 'to', 'be', 'im', 'looking', 'into', 'picking', 'up',
'powerbook', 'or', 'maybe', 'and', 'have', 'bunch', 'of', 'questions', 'that', 'hopefully', 'somebody', 'can', 'answe
r', 'does', 'anybody', 'know', 'any', 'dirt', 'on', 'when', 'the', 'next', 'round', 'of', 'powerbook', 'introduction
s', 'are', 'expected', 'id', 'heard', 'the', 'was', 'supposed', 'to', 'make', 'an', 'appearence', 'this', 'summer',
'but', 'havent', 'heard', 'anymore', 'on', 'it', 'and', 'since', 'dont', 'have', 'access', 'to', 'macleak', 'was', 'w
ondering', 'if', 'anybody', 'out', 'there', 'had', 'more', 'info', 'has', 'anybody', 'heard', 'rumors', 'about', 'pri
ce', 'drops', 'to', 'the', 'powerbook', 'line', 'like', 'the', 'ones', 'the', 'duos', 'just', 'went', 'through', 'rec
ently', 'whats', 'the', 'impression', 'of', 'the', 'display', 'on', 'the', 'could', 'probably', 'swing', 'if', 'got',
'the', 'mb', 'disk', 'rather', 'than', 'the', 'but', 'dont', 'really', 'have', 'feel', 'for', 'how', 'much', 'bette
r', 'the', 'display', 'is', 'yea', 'it', 'looks', 'great', 'in', 'the', 'store', 'but', 'is', 'that', 'all', 'wow',
'or', 'is', 'it', 'really', 'that', 'good', 'could', 'solicit', 'some', 'opinions', 'of', 'people', 'who', 'use', 'th
e', 'and', 'day', 'to', 'day', 'on', 'if', 'its', 'worth', 'taking', 'the', 'disk', 'size', 'and', 'money', 'hit', 't
o', 'get', 'the', 'active', 'display', 'realize', 'this', 'is', 'real', 'subjective', 'question', 'but', 'ive', 'onl
y', 'played', 'around', 'with', 'the', 'machines', 'in', 'computer', 'store', 'breifly', 'and', 'figured', 'the', 'op
inions', 'of', 'somebody', 'who', 'actually', 'uses', 'the', 'machine', 'daily', 'might', 'prove', 'helpful', 'how',
'well', 'does', 'hellcats', 'perform', 'thanks', 'bunch', 'in', 'advance', 'for', 'any', 'info', 'if', 'you', 'coul
d', 'email', 'ill', 'post', 'summary', 'news', 'reading', 'time', 'is', 'at', 'premium', 'with', 'finals', 'just', 'a
round', 'the', 'corner', 'tom', 'willis', 'purdue', 'electrical_engineering', 'convictions', 'are', 'more', 'dangerou
s_enemies', 'of', 'truth', 'than', 'lies', 'nietzsche']
```

In the end, we will filter by appearance frequency and so unnatural bigrams like ‘for_ice’ will be thrown out as they won’t appear enough to make it into the most popular tokens.

Remove stopwords and lemmatize

Some extremely common words which would appear to be of little value in helping select topics are excluded from the vocabulary, often known as stop words. A common list of stop words is:

⇒ A	And	As	At	An	Are
⇒ Be	By	For	From	It	Of
⇒ Was	Were	He	Is	The	With

Whereas, Lemmatization converts the word to its base form. For example: Running → Run, Eating → Eat. In our case of newgroup dataset the below highlighted portion is one of the examples of how document gets changed after removing stop words and lemmatization.

Before Stop words removal and Lemmatization:

'convictions', 'are', 'more', 'dangerous_enemies'

After Stop words removal and Lemmatization:

'conviction', 'dangerous_enemie'

```
# Do lemmatization keeping only noun, adj, vb, adv
data_lemmatized = lemmatization(data_words_bigrams, allowed_
print(data_lemmatized[2])
```

```
['question', 'engineering', 'computer', 'network', 'distribution_
eekend', 'start', 'life', 'way', 'back', 'sooo', 'be', 'market',
ok', 'pick', 'maybe', 'bunch', 'question', 'hopefully', 'answer',
'd', 'hear', 'suppose', 'make', 'appearence', 'summer', 'hear',
ar', 'rumor', 'price', 'drop', 'line', 'one', 'go', 'recently',
wing', 'get', 'disk', 'rather', 'really', 'feel', 'much', 'well',
d', 'could', 'solicit', 'opinion', 'people', 'day', 'worth', 'tal
ay', 'realize', 'real', 'subjective', 'question', 'have', 'play',
e', 'opinion', 'actually', 'use', 'machine', 'daily', 'may', 'pro
h', 'advance', 'info', 'could', 'email', 'ill', 'post', 'summary
rner', 'electrical_engineere', 'conviction', 'dangerous_enemie',
```

Creating the dictionary and Corpus needed for Topic Modeling

Now that we have our text all cleaned up, we will create a dictionary that will give every word a unique number and the frequency of its occurrence in a document. In a better language this is known as vectorization.

Run the LDA Model and get the Topics

We have finally reached the part where we model the data as now, we have everything we need to train an LDA Model.

⇒ In addition to the corpus and dictionary, we need to provide the number of topics as well.

⇒ Apart from that, alpha and eta are hyperparameters that affect sparsity of the topics. According to the Gensim docs, both defaults to 1.0/ num_topics prior.

⇒ Chunk size is the number of documents to be used in each training chunk. update_every determines how often the model parameters should be updated and passes is the total number of training passes.

⇒ I will be passing the number of topics = 2 after determining the optimal k value. This will be discussed in detail in next chapter.

⇒ We can see that Topic 0 comprises of words that relate to 'writing of articles' and Topic 1 relates to 'Sports'.

⇒ Also, we can see that the Coherence Score for this is 0.63 and a model having a Coherence Score above 0.50 is considered acceptable. Hence, the model is yielding good results we can say.

```
# Build LDA model
lda_model = gensim.models.ldamodel.LdaModel(corpus=corpus,
                                            id2word=id2word,
                                            num_topics=2,
                                            random_state=100,
                                            update_every=1,
                                            chunksize=200,
                                            passes=20,
                                            alpha='auto',
                                            per_word_topics=True)
```

```
# Print the Keyword in the 10 topics
pprint(lda_model.print_topics(num_topics=2, num_words=10))
doc_lda = lda_model[corpus]

```

```
[(0,
  '0.011*"line" + 0.010*"write" + 0.009*"would" + 0.006*"say" + 0.006*"know" + '
  '0.006*"article" + 0.006*"people" + 0.006*"make" + 0.005*"be" + '
  '0.005*"think"'),
 (1,
  '0.371*"ax" + 0.022*"team" + 0.017*"game" + 0.013*"player" + 0.009*"play" + '
  '0.008*"hockey" + 0.008*"season" + 0.007*"score" + 0.005*"win" + '
  '0.005*"playoff"')]
```

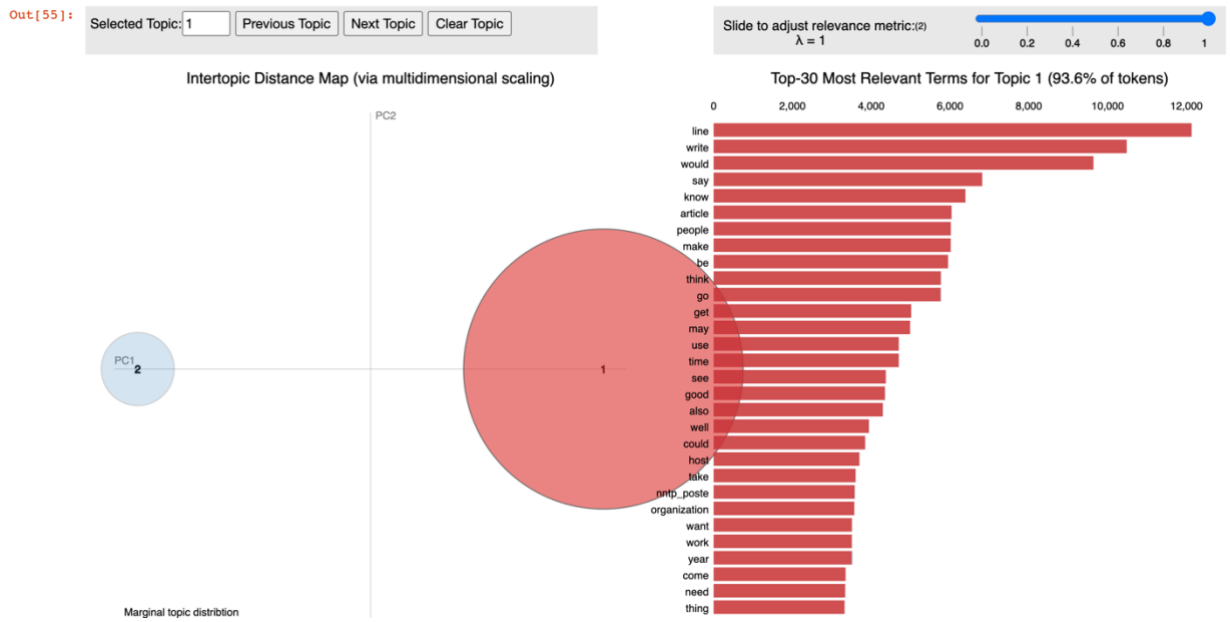
```
# Compute Perplexity
print('\nPerplexity: ', lda_model.log_perplexity(corpus))  # a measure of how good the model is. lower the better.

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

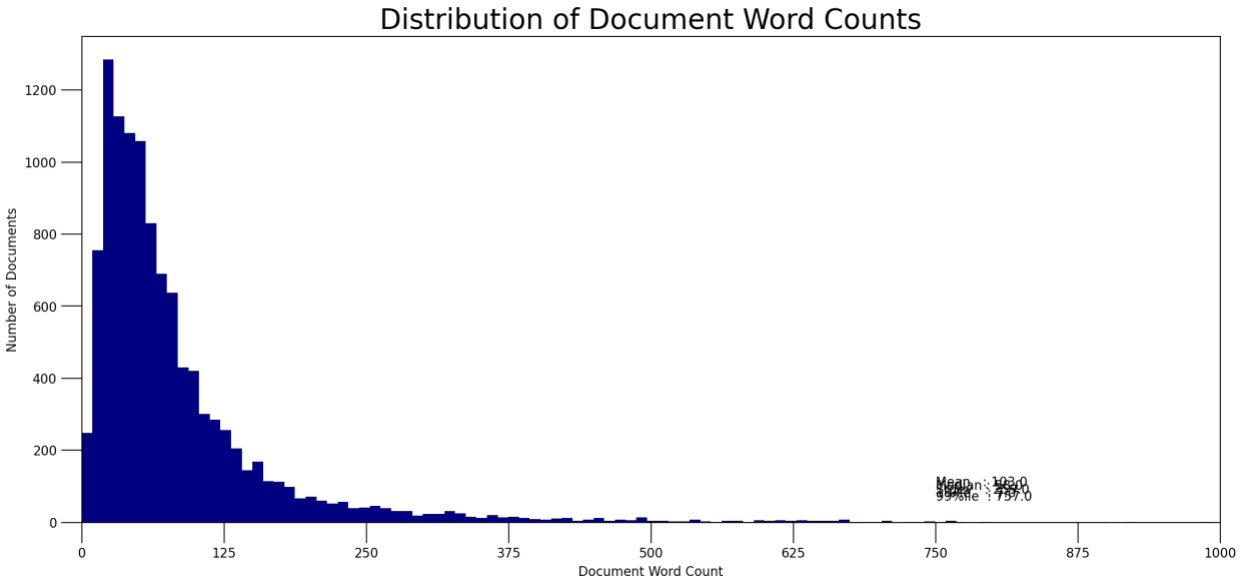
```
Perplexity:  -7.858967564217417

Coherence Score:  0.6384698542264106
```

Visualizations of the LDA Model

PyLDAvis is one of the most well-known visualization of Topic Modeling. We have two circles representing the 2 topics here. Since the size of the circle 1 is bigger it suggests that it is more dominant topic as compared to topic 2.

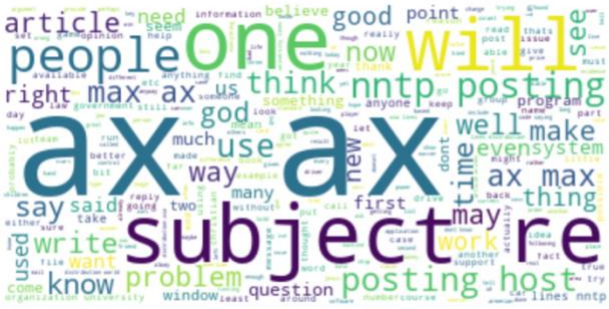
Also, the distance between the two circles represent the similarity between the topics. Since there is no overlapping and they are quite distant we can say that both topics are different than one another. 93% of the token belong to topic 1.



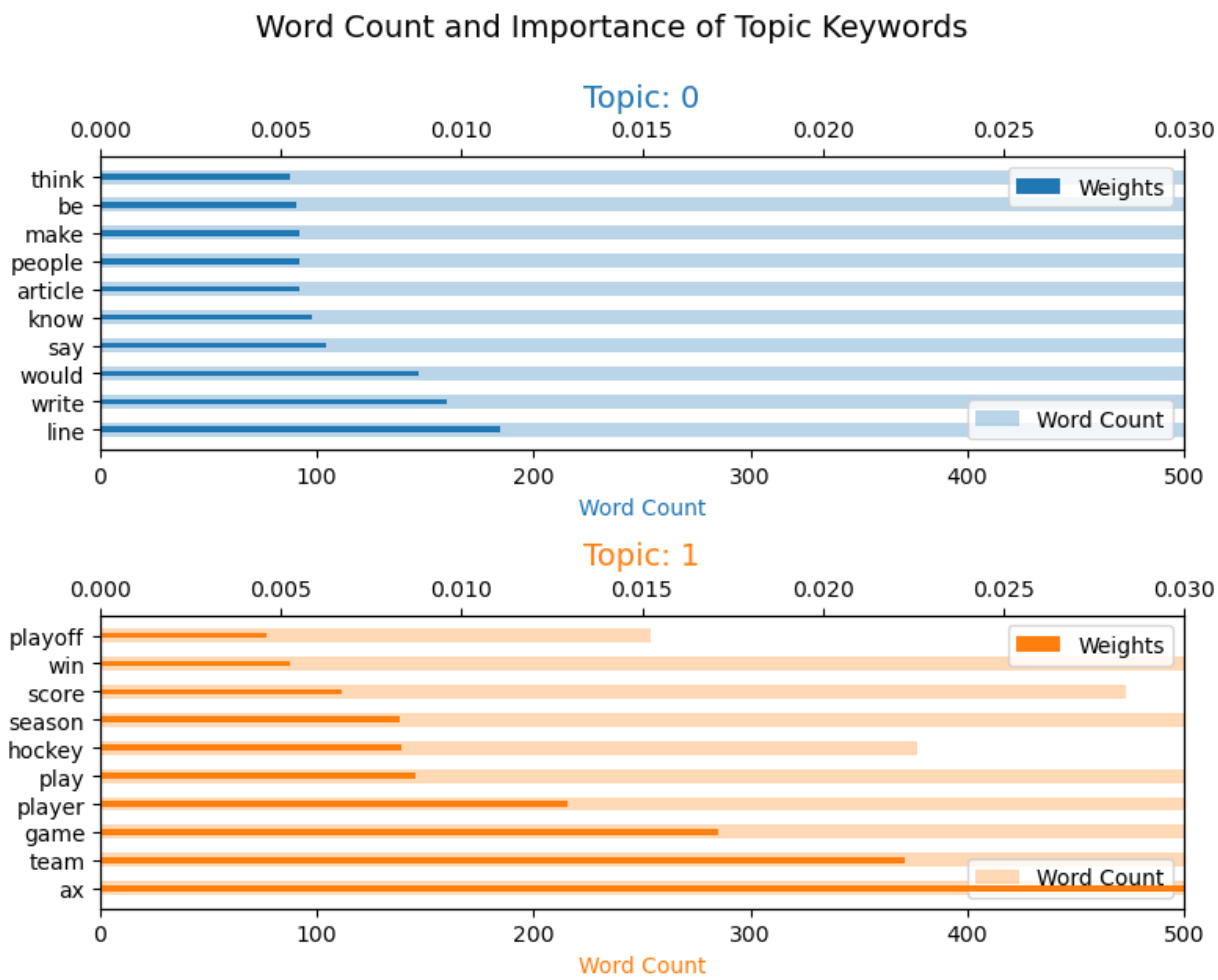


The dataset has more than 1200 documents and the highest word count is 750 words for a document.

```
from wordcloud import WordCloud
sample_list = []
for sublist in data_words:
    for item in sublist:
        sample_list.append(item)

list_string = ','.join((sample_list))
wordcloud = WordCloud(background_color="white", max_words=5000, contour_width=5, contour_color='steelblue')
wordcloud.generate(list_string)
wordcloud.to_image()
```

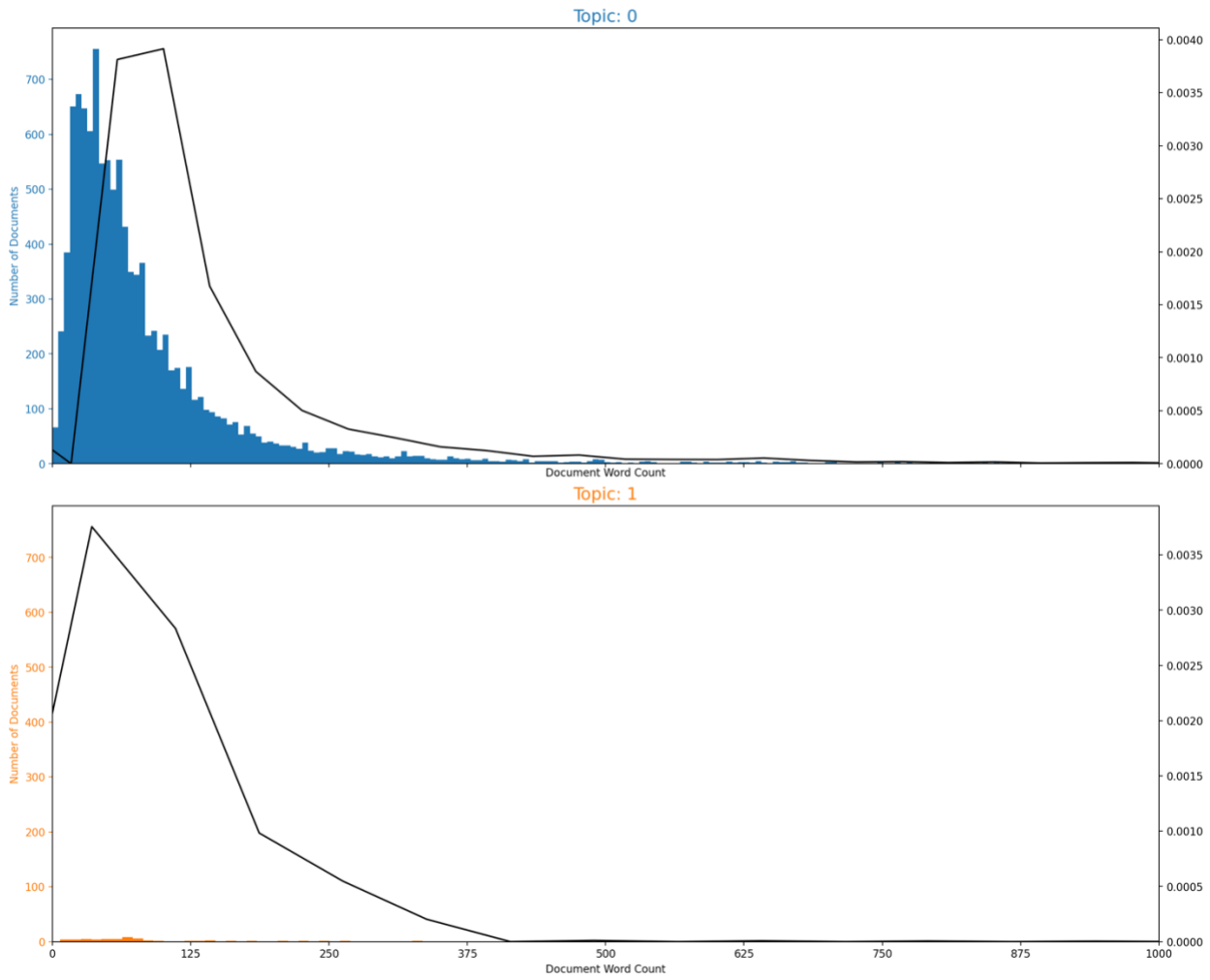


The most dominant word from the corpus. Words like ax, subject, will, one, re appeared quite frequently in the newspaper post.

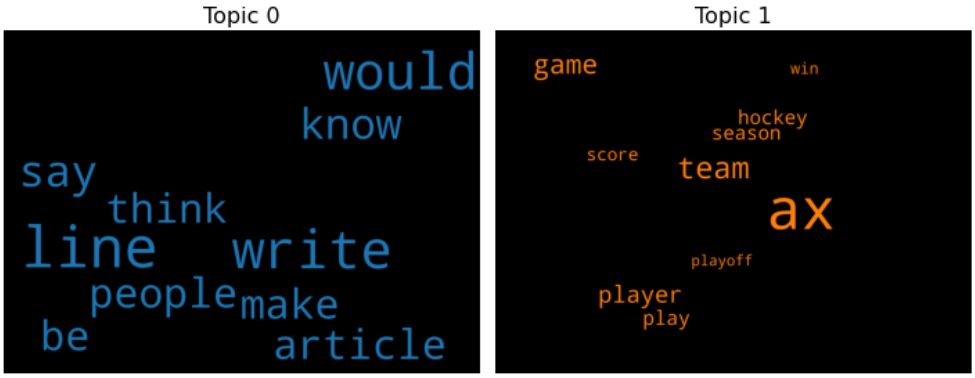


The above figure here shows the word count and importance of Topic Keywords. Importance is the weight of the word in the particular topic. The darker shade bars show the weights and the lighter shade bars show the word frequency.

The same diagram can also be represented in another way as attached below.



The top words of the topics can also be displayed as a word cloud.



Optimal k value for best Coherence Score

It is an important part that we have to decide what the topics mean by interpreting the top words. The model will find us as many topics as we tell it to so it a wise choice to make. Too large and we will likely only find very general topics which don't tell us anything new, too few and the algorithm will pick up on noise in the data and not return meaningful topics. So, this is an important parameter to think about.

There is no single best way and I am not even sure if there are any standard practices for this. Few ways to find the optimal k are:

1. Try out different values of k, select the one that has the largest likelihood.
2. Instead of LDA, see if you can use HDP-LDA
3. Train multiple LDA models.

I chose option 3 and after running the code for optimal k the result looks like this:

jupyter lda_tuning_results.csv ✔ 09/18/2020

File Edit View Language

```
Topics,Coherence
2,0.5526029760242888
2,0.5365325669851206
2,0.51719105391671O2
2,0.5073280817815647
3,0.5751143673500483
3,0.53757103593240 43
3,0.515117281209477
3,0.5086044742723261
4,0.5457981060991868
4,0.537359158253268
4,0.5098762957114495
4,0.5063347049648665
2,0.5443394437713119
2,0.5030002141661389
2,0.5069631379733213
2,0.5093402083028249
3,0.5479982431728958
3,0.5045515857633469
3,0.5049832834413392
3,0.503261207304552
4,0.54785209897157 86
4,0.4985661500343358
4,0.4795847222580221
4,0.5112460441051452
```

Advanced coding as per Business Requirements

We saw Gensim helps fetch the Topics from a corpus of documents so conveniently. But there is more than getting the Topics.

Every business has its own requirements which can be achieved by tweaking the results obtained by running the LDA Model.

In case of Workership, the requirement is to get sentence level dominant Topic.

Business Goal - Workership:

The end goal is to append all the sentences from the documents based on their dominant topics and further use these for Sentiment Analysis.

Click on this file to have access to the codes.

Step 1: Get the topic number of each word

```
counter = 0
for i in lda_model.show_topics(formatted = False, num_topics = 10, num_words=len(lda_model.id2word)):
    if counter == 0:
        topic_num = i[0]
        topic_words = i[1]
        temp1 = pd.DataFrame(topic_words)
        temp1.columns = ['word', topic_num]
        counter = counter + 1
        #     temp1['topic'] = topic_num
    else:
        topic_num = i[0]
        topic_words = i[1]
        temp2 = pd.DataFrame(topic_words)
        temp2.columns = ['word', topic_num]
        temp1 = pd.merge(temp1, temp2, on = "word", how = 'left')
temp1
```

	word	0	1	2	3	4
0	room	4.667239e-02	1.654942e-04	6.031700e-02	0.000012	0.000002
1	check	1.769649e-02	4.482792e-07	1.657244e-03	0.000012	0.000002
2	tell	1.340097e-02	2.998812e-07	3.494452e-07	0.000012	0.000002
3	get	1.295711e-02	7.963565e-03	1.458630e-05	0.000012	0.002330
4	ask	1.291850e-02	1.015269e-03	6.407576e-05	0.000012	0.000002
...	...	...	...	...	...	...
31284	moderize	6.483783e-07	2.924752e-07	3.427724e-07	0.000012	0.000002
31285	materialize	6.483783e-07	2.924670e-07	3.427837e-07	0.000012	0.000002
31286	onstreet	6.483783e-07	2.924670e-07	3.427837e-07	0.000012	0.000002
31287	compermise	6.483783e-07	2.924752e-07	3.427724e-07	0.000012	0.000002
31288	athenian	6.483778e-07	2.924791e-07	3.427749e-07	0.000012	0.000002

31289 rows × 6 columns

Challenge: How to get Dominant Topics for sentences if a word can have more than one Topic number assigned to it

Like we can notice in the image a word may be assigned to more than one topic i.e. a word can have multiple topic numbers but with varying weightages.

Step 2: Assign just one topic number to each word by choosing the topic number with highest weightage (Term Score) for that word.

```
# This is the ccode that extarcts the topic that has the highest weight
words=temp1
main_topic = list()
for i in range(len(words)):
    top1 = 0
    for j in range(1,6):
        if words.iloc[i,j] > top1:
            top1 = words.iloc[i,j]
            top2 = j-1
    main_topic.append(top2)
#print(i, top2)
#print(words.iloc[i, j])
words['main_topic'] = main_topic

    # If all you want is the word and its main topic save it into this dataframe
word_main_topic = words[['word', 'main_topic']]
word_main_topic

```

	word	main_topic
0	room	2
1	check	0
2	tell	0
3	get	0
4	ask	0
...	...	...
31284	moderize	3
31285	materialize	3
31286	onstreet	3
31287	compermise	3
31288	athenian	3

Step 3: Get dominant Topic of each sentence

```
from scipy import stats
import numpy as np

dict2={}
#split each sentence
for review_no in range(len(df)):
#for review_no in range(100):
    review= df.loc[review_no,'Review']
    review=review.replace(',','.')
    sentences=review.split('.')

#create an empty dictionay
    dict1={}
#split each word
    for sentence in sentences:
        words=sentence.split(' ')
        #dominant topic
        temp=word_main_topic[word_main_topic['word'].isin(words)]
        #use mode to find the max frequency of a topic
        array1=np.array(temp['main_topic'].tolist())
        m=stats.mode(array1)
        if len(m[0])==0:
            continue
        m=m[0][0]
        dict1[sentence]=str(m)

    dict2[review_no]=dict1

import json
with open('dominantTopic_sentences1.json','w') as file:
    json.dump(dict2,file)

```

- Here we split the documents into sentences and the sentences into words
- After splitting and having a dictionary of words and its respective dominant topic next I tried to count that which topic number is repeated maximum number of times in a sentence. For example: for “This hotel is clean” if the topic number of words are:

  This: 0 hotel: 0 is: 1 clean: 0

  Then the sentence will belong to Topic: ‘0’

Another scenario is that there is equal distribution of words belonging to same topic i.e. This: 0 hotel: 0 is: 1 clean: 1

In this case, we will select the first topic hence the sentence will belong to Topic: '0'.

- After running the code with above logic, we get the following output which I saved as a json file: -


```
{"0": {"nice hotel expensive parking got good deal stay hotel anniversary": "1", " arrived late evening took advice previous reviews did valet parking": "1", " check quick easy": "1", " little disappointed non-existent view room room clean nice size": "1", " bed comfortable woke stiff neck high pillows": "1", " not soundproof like heard music room night morning loud bangs doors opening closing hear people talking hallway": "0", " maybe just noisy neighbors": "0", " aveda bath products nice": "1", " did not goldfish stay nice touch taken advantage staying longer": "1", " location great walking distance shopping": "1", " overall nice experience having pay 40 parking night": "1"}, "1": {"ok nothing special charge diamond member hilton decided chain shot 20th anniversary seattle": "0", " start booked suite paid extra website description not": "0", " suite bedroom bathroom standard hotel room": "1", " took printed reservation desk showed said things like tv couch ect desk clerk told oh mixed suites description kimpton website sorry free breakfast": "0", " got kidding": "0", " embassy suits sitting room bathroom bedroom unlike kimpton calls suite": "1", " 5 day stay offer correct false advertising": "0", " send kimpton preferred guest website email asking failure provide suite advertised website reservation description furnished hard copy reservation printout website desk manager duty did not reply solution": "0", " send email trip guest survey did not follow email mail": "0", " guess tell concerned guest": "0", "the staff ranged indifferent not helpful": "1", " asked desk good breakfast spots neighborhood hood told no hotels": "1", " gee best breakfast spots seattle 1/2 block away convenient hotel does not know exist": "1", " arrived late night 11 pm inside run bellman busy chating cell phone help bags": "0", "prior arrival emailed hotel inform 20th anniversary half really picky wanted make sure good": "0", " got nice email saying like deliver bottle champagne chocolate covered strawberries room arrival celebrate": "1", " told needed foam pillows": "0", " arrival no champagne strawberries no foam pillows great room view alley high rise building good not better housekeeping staff cleaner room property": "1", " impressed left morning shopping room got short trips 2 hours": "1", " beds comfortable": "1", "not good ac-heat control 4 x 4 inch screen bring green shine directly eyes night": "1", " light sensitive tape controls": "1", "this not 4 start hotel clean business hotel super high rates": "1", " better chain hotels seattle": "1"}, "2": {"nice rooms not 4* experience hotel monaco seattle good hotel n't 4* level": "1", "positives large bathroom mediterranean suite comfortable bed pillowsattentive housekeeping staffnegatives ac unit malfunctioned stay desk disorganized": "1", " missed 3 separate wakeup calls": "1", " concierge busy hard touch": "1", " did n't provide guidance special requests": "1", "tv hard use ipod sound dock suite non functioning": "1", " decided book mediterranean suite 3 night weekend stay 1st choice rest party filled": "1", " comparison w spent 45 night larger square footage room great soaking tub whirlpool jets nice shower": "1", "before stay hotel arrange car service price 53 tip reasonable driver waiting arrival": "1", "checkin easy downside room picked 2 person jacuzi tub no bath accessories salts bubble bath did n't stay": "1", " night got 12/1a checked voucher bottle champagne nice gesture fish waiting room": "0", " impression room huge open space felt room big": "1", " tv far away bed chore change channel": "1", " ipod dock broken disappointing": "0", "in morning way asked desk check thermostat said 65f 74 2 degrees warm try cover face night bright blue light kept": "0", " got room night no": "1", " 1st drop desk": "0", " called maintainence came look thermostat told play settings happy digital box wo n't work": "0", " asked wakeup 10am morning did n't happen": "1", " called later 6pm nap wakeup forgot": "0", " 10am wakeup morning yep forgotten": "1", "the bathroom facilities great room surprised room sold whirlpool bath tub n't bath amenities": "1", " great relax water jets going": "0"}, "3": {"unique": "1", " great stay": "1", " wonderful time hotel monaco": "1", " location excellent short stroll main downtown shopping area": "1", " pet friendly room showed no signs animal hair smells": "0", " monaco suite sleeping area big striped curtains pulled closed nice touch felt cosy": "1", " goldfish named brandi enjoyed": "0", " did n't partake free wine coffee/tea service lobby thought great feature": "0", " great staff friendly": "1", " free wireless internet hotel worked suite 2 laptops": "1", " decor lovely eclectic mix pattens color palatte": "0", " animal print bathrobes feel like rock stars": "0", " nice did n't look like sterile chain hotel hotel personality excellent stay": "1"}, "4": {"great
```


- Now, these segregated sentences based on their Dominant Topics can be further studied for sentiment analysis.

Further Directions to go towards

In the coming semester I look forward to building further on the concept of NLP and some of the way forwards are:

- *Intelligent techniques to not only get the number of topics with best coherence score but also get suitable values of alpha and beta when I run the optimal k code.*
- *Get the Dominant topics on Sentence level without removing the stop words so that we have more meaningful sentences for sentimental analysis.*
- Explore about more methods like distributed environment, parallel processing, big data processing which I am currently unaware of.

References:

https://nlpforhackers.io/topic-modeling/

https://monkeylearn.com/blog/introduction-to-topic-modeling/#:~:text=Topic%20modeling%20is%20a%20machine,been%20previously%20classified%20by%20humans.

https://medium.com/analytics-vidhya/topic-modeling-using-lda-and-gibbs-sampling-explained-49d49b3d1045

https://towardsdatascience.com/evaluate-topic-model-in-python-latent-dirichlet-allocation-lda-7d57484bb5d0