Unsupervised Topic Modelling using Latent Dirichlet Allocation

In this notebook, I have used Quora's question dataset to allocate a unique topic to each question using unsupervised method of
1) Latent Dirichlet Allocation(LDA)
2) Non-Negative Matrix Factorization(NMF)

In [4]:
```
#importing the relevant library
import pandas as pd
import spacy
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.decomposition import LatentDirichletAllocation
from spacy.lang.en.stop_words import STOP_WORDS
from sklearn.decomposition import NMF
import random
```

In [2]:
```
quora_df = pd.read_csv("quora_questions.csv")
```

In [3]:
```
quora_df.head()
```

Out[3]:

	Question
0	What is the step by step guide to invest in sh...
1	What is the story of Kohinoor (Koh-i-Noor) Dia...
2	How can I increase the speed of my internet co...
3	Why am I mentally very lonely? How can I solve...
4	Which one dissolve in water quikly sugar, salt...

In [ ]:
```

```

In [5]:
```
#Importing the largest english library as "en"
nlp = spacy.load('en')
```

In [6]:
```
#Lemmatization and Stop word removal on dataset
```

In [10]:
```
quora_df['Question'][0]
```

Out[10]:
```
'What is the step by step guide to invest in share market in india?'
```

In [11]:
```
doc = nlp(quora_df['Question'][0])
```

In [18]:
```
str_doc = ""
for token in doc:
    str_doc += " " + token.lemma_

str_doc
```

Out[18]:
```
' what be the step by step guide to invest in share market in india ?'
```

In [19]:
```
```

In [20]:
```
#Now, we lemmatize and then remove stop words
```

In [22]:
```
# Create list of word tokens after removing stopwords
filtered_sentence = ""

# Create list of word tokens
token_list = []
for token in doc:
    token_list.append(token.text)



for word in token_list:
    lexeme = nlp.vocab[word]
    if lexeme.is_stop == False:
        filtered_sentence += " " + word
print(token_list)
print(filtered_sentence)
```

```
['What', 'is', 'the', 'step', 'by', 'step', 'guide', 'to', 'invest', 'in', 'share', 'm
arket', 'in', 'india', '?']
 step step guide invest share market india ?
```

The first step we will execute is developing a tf-idf document-term matrix using TfidfVectorizer with the following conditions:

1) max_df = 0.95 ---> The word can occur in a maximum of 95% of the document
2) min_df = 2 ---> The word must occur in a minimum of 2 documents to be a unique word in the tf-idf
3) Remove all the stopwords in the spacy stopwords vocabulary

In [45]:
```
tfidf = TfidfVectorizer(max_df=0.95, min_df=2, stop_words='english')
```

In [47]:
```
#Fit and transform the input
dtm = tfidf.fit_transform(quora_df['Question'])
```

In [49]:
```
dtm # 404289 Quora questions and 38669 unique words
```

Out[49]:
```
<404289x38669 sparse matrix of type '<class 'numpy.float64'>'
        with 2002912 stored elements in Compressed Sparse Row format>
```

1) Latent Dirichlet Allocation (LDA) Modelling

In the following code cells, we will use LDA to assign the best possible topic out of k=10 different topics to each Quora question

In [64]:
```
LDA = LatentDirichletAllocation(n_components=10,random_state=42) # 10 topics, random_st
```

In [65]:
```
LDA.fit(dtm) # fitting the dtm in the LDA model
```

Out[65]:
```
LatentDirichletAllocation(batch_size=128, doc_topic_prior=None,
                          evaluate_every=-1, learning_decay=0.7,
                          learning_method='batch', learning_offset=10.0,
                          max_doc_update_iter=100, max_iter=10,
                          mean_change_tol=0.001, n_components=10, n_jobs=None,
                          perp_tol=0.1, random_state=42, topic_word_prior=None,
                          total_samples=1000000.0, verbose=0)
```

Viewing the number of unique words per LDA component(topic)

In [66]:
```
len(cv.get_feature_names()) # this is a list of stored words
```

Out[66]:
```
38669
```

In [68]:
```
for i in range(10): # selecting 10 random words from the list of unique words
    random_word_id = random.randint(0,38669)
    print(cv.get_feature_names()[random_word_id])
```

```
bhagvat
webceo
509
sgpa
omit
moodle
cliffs
disrupting
guardian
gaddar
```

In [69]:
```
len(LDA.components_) #type is a numpy array containing probabilities for each word (7 b
```

Out[69]:
```
10
```

In the following cell, each of the 38669 words has been listed as a probability that it belongs to a particular topic

In [70]:
```
LDA.components_
```

Out[70]:
```
array([[1.00004583e-01, 1.00018189e-01, 1.00000003e-01, ...,
        1.00000001e-01, 1.87820010e+00, 1.00000001e-01],
       [1.00012428e-01, 1.08752198e+01, 1.00000004e-01, ...,
        1.00000001e-01, 1.00000001e-01, 1.00000001e-01],
       [1.00000569e-01, 1.93794516e-01, 1.00000005e-01, ...,
        1.00000001e-01, 1.00000001e-01, 1.00000001e-01],
       ...,
       [6.03551160e+00, 2.29179532e+02, 1.00000004e-01, ...,
        1.00000001e-01, 1.00000001e-01, 1.00000001e-01],
       [1.00010139e-01, 8.79048227e+00, 1.00136069e-01, ...,
        1.00000001e-01, 1.00000001e-01, 1.00000001e-01],
       [1.26779647e-01, 1.70058015e+01, 1.00000004e-01, ...,
        1.22648290e+00, 1.00000001e-01, 1.22648290e+00]])
```

In [71]:
```
single_topic = LDA.components_[0] # first topic
```

In [72]:
```
# Returns the indices that would sort this array from lowest probability to highest for
single_topic.argsort()
```

Out[72]:
```
array([17176, 19303,  9274, ..., 15060, 12200,  4632], dtype=int64)
```

In [73]:
```
# Word least representative of this topic
single_topic[19303]
```

Out[73]:
```
0.1000000002567497
```

In [74]:
```
# Word most representative of this topic
single_topic[4632]
```

Out[74]:
```
1293.8034393594346
```

In [75]:
```
for index,topic in enumerate(LDA.components_):
    print(f'THE TOP 15 WORDS FOR TOPIC #{index}')
    print([cv.get_feature_names()[i] for i in topic.argsort()[-15:]])
    print('\n')
```

```
THE TOP 15 WORDS FOR TOPIC #0
['company', 'mechanical', 'engineer', 'marketing', 'india', 'science', 'work', 'job',
'career', 'software', 'computer', 'does', 'good', 'engineering', 'best']


THE TOP 15 WORDS FOR TOPIC #1
['indian', 'rupee', 'modi', 'rs', 'black', 'india', 'word', 'best', 'english', 'progra
mming', 'notes', '1000', '500', 'language', 'learn']


THE TOP 15 WORDS FOR TOPIC #2
['education', 'hotel', 'good', 'travel', 'universe', 'book', 'favorite', 'energy', 'do
es', 'read', 'purpose', 'best', 'time', 'books', 'life']


THE TOP 15 WORDS FOR TOPIC #3
['day', 'answers', 'mind', 'does', 'answer', 'things', 'best', 'new', 'movie', 'know',
'ask', 'people', 'question', 'questions', 'quora']


THE TOP 15 WORDS FOR TOPIC #4
['apply', 'jobs', 'canada', 'college', 'overcome', 'visa', 'difference', 'depression',
'job', 'card', 'car', 'best', 'rid', 'india', 'does']


THE TOP 15 WORDS FOR TOPIC #5
['china', 'people', 'women', 'relationship', 'mean', 'math', 'did', 'pakistan', 'indi
a', 'feel', 'sex', 'war', 'like', 'world', 'does']


THE TOP 15 WORDS FOR TOPIC #6
['meaning', 'good', 'way', 'school', 'fat', 'day', 'days', 'places', 'visit', 'study',
'exam', 'prepare', 'lose', 'best', 'weight']


THE TOP 15 WORDS FOR TOPIC #7
['india', 'business', 'english', 'start', 'hillary', 'clinton', 'president', 'earn',
'online', 'donald', 'make', 'improve', 'best', 'trump', 'money']


THE TOP 15 WORDS FOR TOPIC #8
['way', 'password', 'number', 'free', 'does', 'whatsapp', 'use', 'app', 'android', 'we
bsite', 'phone', 'best', 'account', 'facebook', 'instagram']


THE TOP 15 WORDS FOR TOPIC #9
['die', 'time', 'age', 'friend', 'best', 'thing', 'important', 'increase', 'like', 'li
fe', 'people', 'old', 'does', 'girl', 'love']
```

From the above output:
we cans see that each set of words represents a particular topic that we have to decide(as per our best knowledge)

#0 --> **Job** related questions

#1 --> **Finance** related questions

#2 --> **Lifestyle** related questions

#3 --> **QnA** related questions

#4 --> **Application/Jobs** related questions

#5 --> **People/Nationality** related questions

#6 --> **Competition** related questions

#7 --> **Politics** related questions

#8 --> **Social Media** related questions

#9 --> **Friends** related questions

In [76]:
```
topic_results = LDA.transform(dtm)
```

In [77]:
```
topic_results.shape # 404289 questions, and 10 topics
```

Out[77]:
```
(404289, 10)
```

In [78]:
```
topic_results[0] # gives ten different probabilities for the first document. Document b
```

Out[78]:
```
array([0.03085334, 0.41976533, 0.0308673 , 0.03083837, 0.03083856,
       0.03083856, 0.03084914, 0.33346796, 0.03084128, 0.03084015])
```

In [79]:
```
quora_df.head()
```

Out[79]:

	Question
0	What is the step by step guide to invest in sh...
1	What is the story of Kohinoor (Koh-i-Noor) Dia...
2	How can I increase the speed of my internet co...
3	Why am I mentally very lonely? How can I solve...
4	Which one dissolve in water quikly sugar, salt...

In [80]:
```
topic_results.argmax(axis=1)
```

Out[80]:
```
array([1, 4, 8, ..., 0, 6, 9], dtype=int64)
```

In [81]:
```
quora_df['Topic'] = topic_results.argmax(axis=1)
```

In [82]:
```
quora_df.head(10)
```

Out[82]:

	Question	Topic
0	What is the step by step guide to invest in sh...	1
1	What is the story of Kohinoor (Koh-i-Noor) Dia...	4
2	How can I increase the speed of my internet co...	8
3	Why am I mentally very lonely? How can I solve...	3
4	Which one dissolve in water quikly sugar, salt...	2
5	Astrology: I am a Capricorn Sun Cap moon and c...	2
6	Should I buy tiago?	1
7	How can I be a good geologist?	2
8	When do you use シ instead of し?	3
9	Motorola (company): Can I hack my Charter Moto...	8

2) Non-Negative Matrix Factorization (NMF)

In [85]:
```
nmf_model = NMF(n_components=10,random_state=42)
```

In [86]:
```
nmf_model.fit(dtm) # fitting our tf-idf matrix
```

Out[86]:
```
NMF(alpha=0.0, beta_loss='frobenius', init=None, l1_ratio=0.0, max_iter=200,
    n_components=10, random_state=42, shuffle=False, solver='cd', tol=0.0001,
    verbose=0)
```

In [87]:
```
len(tfidf.get_feature_names())
```

Out[87]:
```
38669
```

In [88]:
```
nmf_model.components_
```

Out[88]:
```
array([[5.49082867e-05, 5.19966837e-02, 4.61688642e-05, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       [1.56032411e-03, 4.31966058e-04, 2.88899887e-05, ...,
        0.00000000e+00, 3.17185353e-03, 0.00000000e+00],
       [0.00000000e+00, 0.00000000e+00, 0.00000000e+00, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       ...,
       [2.31001605e-05, 1.76207768e-03, 0.00000000e+00, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       [1.37967210e-03, 4.47447315e-02, 3.92158710e-06, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       [9.19079320e-04, 4.78220509e-03, 1.90051559e-05, ...,
        1.78457124e-05, 0.00000000e+00, 1.78457124e-05]])
```

In [90]:
```
len(nmf_model.components_)
```

Out[90]:
```
10
```

In [91]:
```
single_topic = nmf_model.components_[0]
```

In [92]:
```
single_topic.argsort()
```

Out[92]:
```
array([38668, 13392, 13390, ..., 22925, 37515,  4632], dtype=int64)
```

In [94]:
```
len(single_topic) # 38669 unique word positions with their probabilities for each topic
```

Out[94]:
```
38669
```

In [93]:
```
# Word least representative of this topic
single_topic[13392]
```

Out[93]:
```
0.0
```

In [95]:
```
# Word most representative of this topic
single_topic[4632]
```

Out[95]:
```
7.367351238463149
```

In [96]:
```
for index,topic in enumerate(nmf_model.components_):
    print(f'THE TOP 15 WORDS FOR TOPIC #{index}')
    print([tfidf.get_feature_names()[i] for i in topic.argsort()[-15:]])
    print('\n')
```

```
THE TOP 15 WORDS FOR TOPIC #0
['places', 'phone', 'buy', 'lose', 'laptop', 'time', 'movie', 'ways', 'weight', '201
6', 'books', 'book', 'movies', 'way', 'best']


THE TOP 15 WORDS FOR TOPIC #1
['looking', 'exist', 'girl', 'look', 'compare', 'really', 'cost', 'time', 'sex', 'lon
g', 'work', 'feel', 'like', 'mean', 'does']


THE TOP 15 WORDS FOR TOPIC #2
['add', 'answered', 'needing', 'post', 'easily', 'improvement', 'delete', 'asked', 'go
ogle', 'answers', 'answer', 'ask', 'question', 'questions', 'quora']


THE TOP 15 WORDS FOR TOPIC #3
['friends', 'facebook', 'black', 'internet', 'free', 'easiest', 'home', 'easy', 'youtu
be', 'ways', 'way', 'earn', 'online', 'make', 'money']


THE TOP 15 WORDS FOR TOPIC #4
['death', 'changed', 'want', 'change', 'live', 'moment', 'things', 'good', 'real', 'da
y', 'important', 'thing', 'meaning', 'purpose', 'life']


THE TOP 15 WORDS FOR TOPIC #5
['company', 'china', 'olympics', 'available', 'engineering', 'business', 'spotify', 'c
ountry', 'start', 'job', 'good', 'world', 'war', 'pakistan', 'india']


THE TOP 15 WORDS FOR TOPIC #6
['speaking', 'languages', 'writing', 'java', 'speak', 'learning', 'skills', 'start',
'way', 'good', 'improve', 'programming', 'language', 'english', 'learn']


THE TOP 15 WORDS FOR TOPIC #7
['presidency', 'happen', 'think', 'presidential', 'vote', '2016', 'better', 'electio
n', 'win', 'did', 'hillary', 'president', 'clinton', 'donald', 'trump']


THE TOP 15 WORDS FOR TOPIC #8
['new', 'modi', 'currency', 'economy', 'government', 'think', 'ban', 'banning', 'blac
k', 'indian', 'rupee', 'rs', '1000', 'notes', '500']


THE TOP 15 WORDS FOR TOPIC #9
['employees', 'girl', 'mind', 'world', 'time', 'going', 'day', 'new', 'things', 'don',
'like', 'think', 'love', 'know', 'people']
```

From the above output:
we cans see that each set of words represents a particular topic that we have to decide(as per our best knowledge)

#0 --> **Technical/Books/Movies** related questions

#1 --> **Looks** related questions

#2 --> **QnA** related questions

#3 --> **Social Media** related questions

#4 --> **Life** related questions

#5 --> **People/Nationality** related questions

#6 --> **Language/Programming** related questions

#7 --> **Politics** related questions

#8 --> **Finance** related questions

#9 --> **Daily time** related questions

In [97]:
```
topic_results = nmf_model.transform(dtm)
```

In [98]:
```
topic_results.shape # 404289 quora questions and 10 topics with probability as value in
```

Out[98]:
```
(404289, 10)
```

In [99]:
```
topic_results[0]
```

Out[99]:
```
array([0.00037633, 0.        , 0.        , 0.00053401, 0.        ,
       0.03007269, 0.00014986, 0.        , 0.00118431, 0.        ])
```

In [100]:
```
quora_df.head()
```

Out[100]:

	Question	Topic
0	What is the step by step guide to invest in sh...	1
1	What is the story of Kohinoor (Koh-i-Noor) Dia...	4
2	How can I increase the speed of my internet co...	8
3	Why am I mentally very lonely? How can I solve...	3
4	Which one dissolve in water quikly sugar, salt...	2

In [101]:
```
quora_df['Topic_NMF'] = topic_results.argmax(axis=1)
```

In [103]:
```
quora_df.head(10)
```

Out[103]:

	Question	Topic	Topic_NMF
0	What is the step by step guide to invest in sh...	1	5
1	What is the story of Kohinoor (Koh-i-Noor) Dia...	4	0
2	How can I increase the speed of my internet co...	8	3
3	Why am I mentally very lonely? How can I solve...	3	8
4	Which one dissolve in water quikly sugar, salt...	2	1
5	Astrology: I am a Capricorn Sun Cap moon and c...	2	1
6	Should I buy tiago?	1	0
7	How can I be a good geologist?	2	6
8	When do you use シ instead of し?	3	2
9	Motorola (company): Can I hack my Charter Moto...	8	5

Now, let us verify if our topic association is correct
Let us first see what the first Quora question in our dataset is and accordingly decide of the correct topic has been allocated to it

In [105]:
```
quora_df['Question'][0]
```

Out[105]:
```
'What is the step by step guide to invest in share market in india?'
```

```
It seems like the first question is related to Finance. <br><br>
Now, we can see that according to LDA, the topic assigned is 1, which is the second
topic, Finance. So, it matches with the Quora question asked.
<br>
For NMF, topic 5 has been allocated to this question, which is business/people
related. This topic may not be directly relatd but can be considered close.
```

In [ ]:
```

```