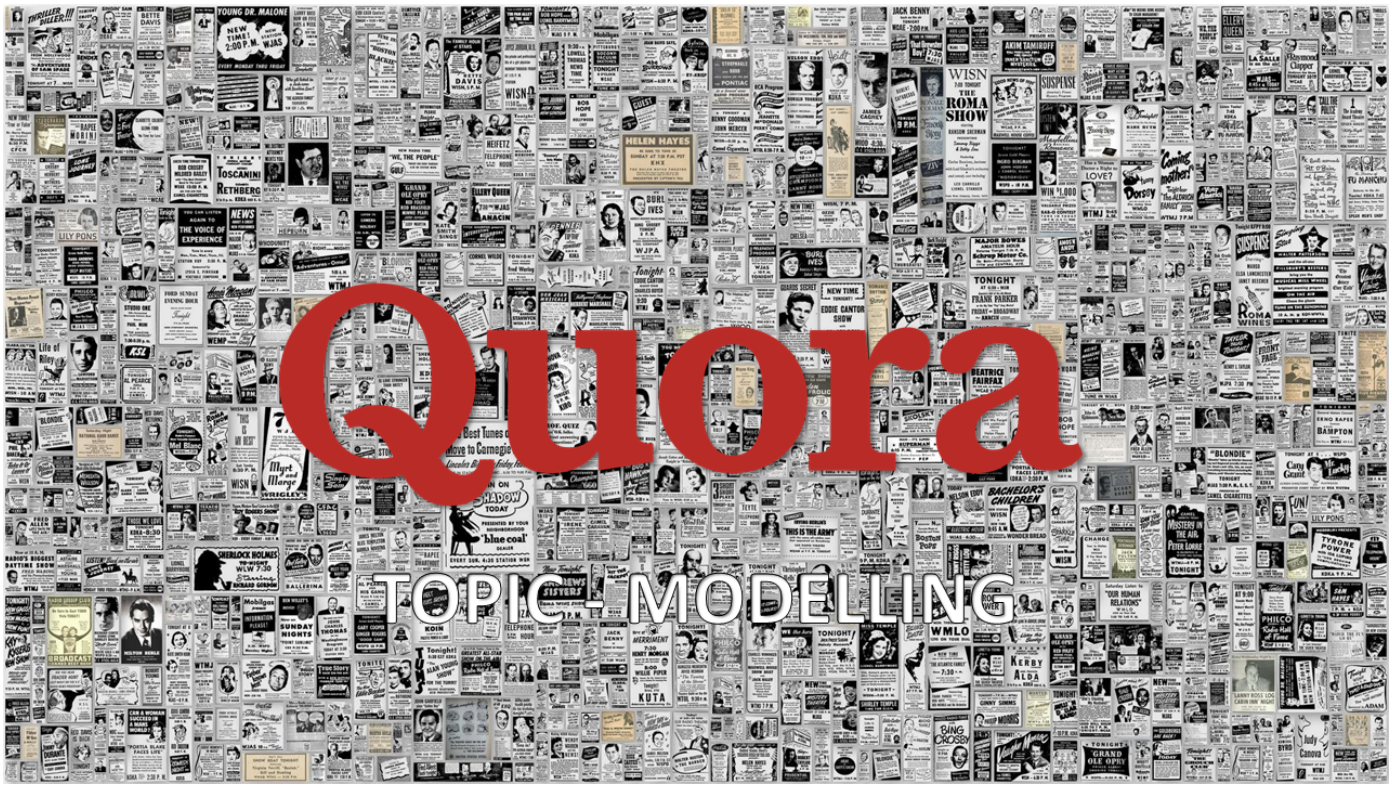


In this Topic Modelling project we will be working with a dataset of over 400,000 quora questions that have no labeled cateogry, and attempting to find 20 cateogries to assign these questions to. This model will be a kind of `Unsupervised Machine Learing Model` as we are working with the unlabelled data.

The dataset can be downloaded from https://www.kaggle.com/c/quora-insincere-questions-classification (https://www.kaggle.com/c/quora-insincere-questions-classification)

In this Project i will be using two methods for topic modelling which are widely known as LDA (Latent Dirichlet Allocation) and NMF (Non-negative Matrix Factorization) using Scikit Learn ML Package and will be visualizing it using pyLDAvis library

Importing the Necessary Libraries

In [1]:

```
import numpy as np
import pandas as pd
import nltk
from sklearn.feature_extraction.text import CountVectorizer,TfidfVectorizer
from sklearn.pipeline import Pipeline
from sklearn.metrics import confusion_matrix,classification_report
import matplotlib.pyplot as plt
import cufflinks as cf
from plotly.offline import init_notebook_mode,plot,iplot
import plotly.graph_objs as go
from plotly import tools
import plotly.figure_factory as ff
import spacy
init_notebook_mode(connected=True)
cf.go_offline()
```

In [2]:

```
nlp=spacy.load('en_core_web_lg')
```

In [4]:

```
df= pd.read_csv(r'C:\Univ\udemy\UPDATED_NLP_COURSE\05-Topic-Modeling\quora_questions.csv')
```

In [5]:

```
df.head()
```

Out[5]:

	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 [6]:

```
df.shape
```

Out[6]:

```
(404289, 1)
```

In [7]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 404289 entries, 0 to 404288
Data columns (total 1 columns):
Question    404289 non-null object
dtypes: object(1)
memory usage: 3.1+ MB
```

In [8]:

```
df.describe()
```

Out[8]:

	Question
count	404289
unique	290456
top	How do I improve my English speaking?
freq	50

In [10]:

```
df['Question'][0] #first sample question
```

Out[10]:

```
'What is the step by step guide to invest in share market in india?'
```

In [11]:

```
df= df.sample(frac=0.5,random_state=101)# as we cannot work with 4million rows with this cp
```

In [12]:

```
df.head()
```

Out[12]:

	Question
121917	How is the training in TCS?
280235	How do you know if someone has blocked you on ...
41961	What are the most amusing differences between ...
40962	What is happening when my eyes get heavy when ...
63548	Which is the best phone under 15000 currently?

In [13]:

```
df.shape
```

Out[13]:

```
(202144, 1)
```

In [ ]:

```

```

Text Pre-processing

At first we will clean the text data by removing the punctuations and stop words then we have to do Lemmatization to stem the words. We will be using spacy for this Pre-processing

In [14]:

```
def spacy_tokenizer(word):
    a = nlp(word) #tokenization
    b=[]
    for token in a:
        if (token.is_stop != True) and (token.is_punct != True): #Stop words and punctuatio
            b.append(token.lemma_) #Lemmatization

    return ' '.join(b)
```

In [15]:

```
df['processed_txt']= df['Question'].apply(spacy_tokenizer)
```

In [16]:

```
df
```

Out[16]:

	Question	processed_txt
121917	How is the training in TCS?	training TCS
280235	How do you know if someone has blocked you on ...	know block messenger
41961	What are the most amusing differences between ...	amusing difference british English american En...
40962	What is happening when my eyes get heavy when ...	happen eye heavy sleep
63548	Which is the best phone under 15000 currently?	good phone 15000 currently
...	...	...
382103	What are the most famous sports in the world? ...	famous sport world play
320755	How many protons and neutrons are in each isot...	proton neutron isotope
214577	What does Thrive Content Builder have that The...	thrive Content Builder Themify Builder
96659	What's the quickest way to get rid of belly fat?	quick way rid belly fat
351298	How can flu like symptoms be a sign of pregnancy?	flu like symptom sign pregnancy

202144 rows × 2 columns

In [96]:

```
df.to_csv("quora_ques_processed.csv",index=False)
```

In [2]:

```
df= pd.read_csv("quora_ques_processed.csv")
```

In [3]:

```
df
```

Out[3]:

	Question	processed_txt
0	How is the training in TCS?	training TCS
1	How do you know if someone has blocked you on ...	know block messenger
2	What are the most amusing differences between ...	amusing difference british English american En...
3	What is happening when my eyes get heavy when ...	happen eye heavy sleep
4	Which is the best phone under 15000 currently?	good phone 15000 currently
...	...	...
202139	What are the most famous sports in the world? ...	famous sport world play
202140	How many protons and neutrons are in each isot...	proton neutron isotope
202141	What does Thrive Content Builder have that The...	thrive Content Builder Themify Builder
202142	What's the quickest way to get rid of belly fat?	quick way rid belly fat
202143	How can flu like symptoms be a sign of pregnancy?	flu like symptom sign pregnancy

202144 rows × 2 columns

In [4]:

```
df=df.dropna() #after preprocessing some values might be null in the 'processed_txt' column
```

In [5]:

```
df.shape
```

Out[5]:

```
(202064, 2)
```

Document Term Matrix

In [6]:

```
dtm_cv= CountVectorizer(stop_words='english',max_df=0.95,min_df=10)
```

In [7]:

```
dtm_cv_mat = dtm_cv.fit_transform(df['processed_txt'])
```

In [8]:

```
dtm_cv_mat.shape
```

Out[8]:

```
(202064, 8122)
```

In [9]:

```
# to get all the voab words in the dataset use dtm_cv.get_feature_names()
# to get all the vocabulary with its index number use dtm_cv.vocabulary_
```

In [10]:

```
dtm_cv_mat
```

Out[10]:

```
<202064x8122 sparse matrix of type '<class 'numpy.int64'>'
        with 886574 stored elements in Compressed Sparse Row format>
```

so the resulting matrix of the datset is 202064x8122 where 202064 represents the rows and 8122 represents the vocab in the entire dataset

In [ ]:

```

```

Latent Dirchilet Allocation

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.

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.

In [11]:

```
from sklearn.decomposition import LatentDirichletAllocation
```

In [12]:

```
LDA_Model = LatentDirichletAllocation(n_components=20,n_jobs=-1,random_state=101)
```

In [13]:

```
LDA_Model.fit(dtm_cv_mat)
```

Out[13]:

```
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=20, n_jobs=-1,
                          perp_tol=0.1, random_state=101, topic_word_prior=N
one,
                          total_samples=1000000.0, verbose=0)
```

In [18]:

```
LDA_Model.components_ #gives the list of all 20 topics and its array of probability values
```

Out[18]:

```
array([[ 1.82221696, 13.99554316,  0.05      , ...,  0.05      ,
         0.05      ,  0.05      ],
       [ 1.65914269,  0.05      ,  0.05      , ...,  0.05      ,
         2.06972087,  0.05      ],
       [ 0.05      ,  0.05      ,  0.05      , ...,  0.05      ,
         0.05      ,  0.05      ],
       ...,
       [ 0.05      ,  0.05      ,  4.21450896, ...,  0.05      ,
         8.03027913,  0.05      ],
       [ 0.05      ,  0.05      ,  0.05      , ...,  0.05      ,
         0.05      ,  0.05      ],
       [ 0.05      ,  0.05      ,  0.05      , ...,  0.05      ,
         0.05      ,  0.05      ]])
```

In [49]:

```
LDA_Model.components_.shape
```

Out[49]:

```
(20, 8122)
```

Where the 20 indicates the clustering groupp(n_components) and 8122 indicates the vocabulary

To know what are the top 15 words in each topic, we can create a dataframe by using the below functions

In [50]:

```
topic_dict = {}
for index, topicprob in enumerate (LDA_Model.components_):
    emp_lst = []
    for element in topicprob.argsort()[-15:] :
        emp_lst.append( dtm_cv.get_feature_names()[element] )

    topic_dict[index]= emp_lst
```

In [51]:

```
topic_df = pd.DataFrame(topic_dict)
topic_df.columns=['Topic1','Topic2','Topic3','Topic4','Topic5','Topic6','Topic7','Topic8','
```

Top 15 topic keywords

In [69]:

```
topic_df.head(15)
```

Out[69]:

	Topic1	Topic2	Topic3	Topic4	Topic5	Topic6	Topic7	Topic8	Topic9	Topic10
0	drug	force	try	help	python	reduce	event	startup	search	stop
1	temperature	health	come	black	chinese	week	safety	stock	interesting	remove
2	drive	family	real	government	java	gain	propose	rid	purpose	india
3	city	happen	dark	problem	speak	fat	amazon	exist	add	china
4	hard	apply	small	india	code	day	handle	market	follow	play
5	car	join	wear	account	skill	police	value	god	interview	great
6	term	hate	universe	rupee	programming	hotel	source	know	difference	test
7	pro	new	light	ban	way	body	rate	mind	post	easy
8	university	life	white	math	read	food	design	believe	people	earn

Now we will predict the topic for each rows in the dataset and assign its corresponding token

In [14]:

```
topic_results= LDA_Model.transform(dtm_cv_mat)
```

In [18]:

```
df['Topic_ID'] = topic_results.argmax(axis=1)
```

In [19]:

```
df.head(10)
```

Out[19]:

	Question	processed_txt	Topic_ID
0	How is the training in TCS?	training TCS	16
1	How do you know if someone has blocked you on ...	know block messenger	3
2	What are the most amusing differences between ...	amusing difference british English american En...	4
3	What is happening when my eyes get heavy when ...	happen eye heavy sleep	7
4	Which is the best phone under 15000 currently?	good phone 15000 currently	16
5	Is time is precious, or are relationships prec...	time precious relationship precious	10
6	What would the consequences of a Brexit be for...	consequence Brexit UK citizen	6
7	How India can respond to the Uri terror attack?	India respond Uri terror attack	7
8	What are some of the most interesting facts ab...	interesting fact lion tiger	5
9	I'm going to kill myself. How should I do it?	go kill	2

In [72]:

```
from sklearn.decomposition import PCA
```

In [57]:

```
pca = PCA(n_components=3, random_state=42)
tred_viz = pca.fit_transform(topic_results)
```

In [58]:

```
tred_viz_df= pd.DataFrame(tred_viz)
```

In [73]:

```
tred_viz_df[1]
```

Out[73]:

```
0         0.218224
1        -0.026120
2        -0.530440
3        -0.070131
4        -0.015220
              ...
202059   -0.046113
202060   -0.066119
202061   -0.022943
202062    0.017808
202063    0.045588
Name: 1, Length: 202064, dtype: float64
```

In [74]:

```
tred_viz_df['topic_id_lda']= df['Topic_ID']
```

In [79]:

```
tred_viz_df
```

Out[79]:

	0	1	2	topic_id_lda
0	0.299328	0.218224	0.112689	16.0
1	-0.092590	-0.026120	-0.089713	3.0
2	-0.037306	-0.530440	0.528105	4.0
3	-0.098005	-0.070131	-0.087768	7.0
4	0.749380	-0.015220	-0.042834	16.0
...	...	...	...	...
202059	-0.075347	-0.046113	-0.088442	19.0
202060	-0.101394	-0.066119	-0.116032	15.0
202061	0.543198	-0.022943	-0.054236	0.0
202062	-0.032250	0.017808	-0.142069	6.0
202063	-0.077510	0.045588	-0.007692	0.0

202064 rows × 4 columns

In [76]:

```
tred_viz_df.to_csv('lda_3d_scat.csv',index=False)
```

In [2]:

```
tred_viz_df = pd.read_csv('lda_3d_scat.csv')
```

The above figure needs to be loaded again by running the code

We'll use a visualization package, pyLDAvis which is designed to help interactively with:

1. Better understanding and interpreting individual topics, and
2. Better understanding the relationships between the topics.

For (1), you can manually select each topic to view its top most frequent and/or "relevant" terms, using different values of the λ parameter. This can help when you're trying to assign a human interpretable name or "meaning" to each topic.

For (2), exploring the Intertopic Distance Plot can help you learn about how topics relate to each other, including potential higher-level structure between groups of topics.

In [21]:

```
import warnings
warnings.filterwarnings('ignore')
import pyLDAvis
import pyLDAvis.sklearn
pyLDAvis.enable_notebook()
p=pyLDAvis.sklearn.prepare(LDA_Model, dtm_cv_mat, dtm_cv, mds='tsne')# WE CAN USE mds= 'mmd
p
```

Out[21]:

In [108]:

```
pyLDAvis.save_html(p, 'lda_QUORA.html')
```

In [ ]:

```

```

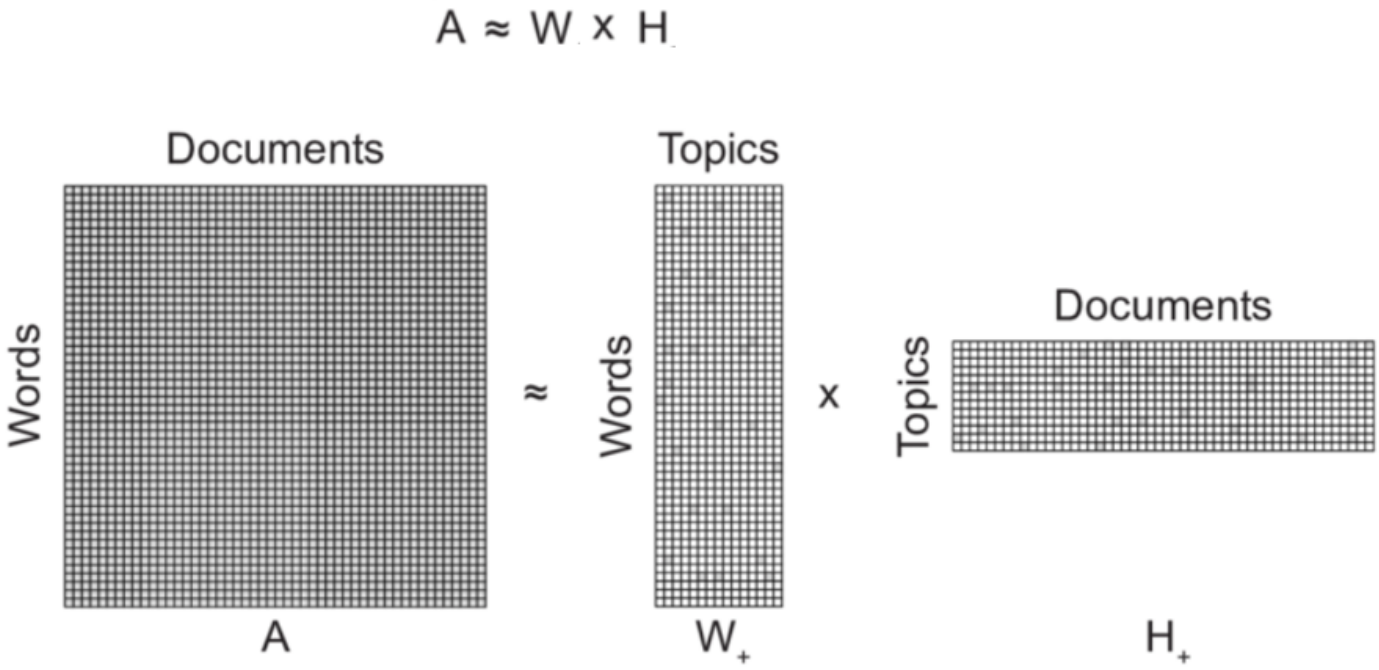
Non-Negative Matix Factorization

Family of linear algebra algorithms for identifying the latent structure in data represented as a non-negative matrix. NMF can be applied for topic modeling, where the input is term-document matrix, typically TF-IDF normalized.

Input: Term-Document matrix, number of topics.

Output: Two non-negative matrices of the original n words by k topics and those same k topics by the m original documents.

Basically, we are going to use linear algebra for topic modeling.



The Non - negative matrix factorization depends on the coefficient values of the H+ Matrix (in the above figure) we will be doing the TF-IDF Transformation instead of a simple CountVectorizer transformation

In [8]:

```
dtm_nmf = TfidfVectorizer(stop_words='english',max_df=0.95,  min_df=10)
```

In [9]:

```
dtm_nmf_mat =dtm_nmf.fit_transform(df['processed_txt'])
```

In [10]:

```
dtm_nmf_mat.shape
```

Out[10]:

```
(202064, 8122)
```

In [11]:

```
from sklearn.decomposition import NMF
```

In [12]:

```
NMF_Model = NMF(n_components=20,random_state=101,init='nndsvd')
```

In [13]:

```
NMF_Model.fit(dtm_nmf_mat)
```

Out[13]:

```
NMF(alpha=0.0, beta_loss='frobenius', init='nndsvd', l1_ratio=0.0, max_iter=
200,
    n_components=20, random_state=101, shuffle=False, solver='cd', tol=0.000
1,
    verbose=0)
```

In [14]:

```
NMF_Model.components_ #gives the list of all 20 topics and its array of probability values
```

Out[14]:

```
array([[5.84057359e-05, 3.69264010e-02, 5.44822857e-04, ...,
        0.00000000e+00, 4.98418265e-04, 1.15387128e-05],
       [0.00000000e+00, 0.00000000e+00, 3.56508644e-05, ...,
        0.00000000e+00, 0.00000000e+00, 6.10239233e-03],
       [8.28837821e-04, 2.98101924e-03, 0.00000000e+00, ...,
        0.00000000e+00, 0.00000000e+00, 4.58035299e-03],
       ...,
       [4.52509345e-04, 1.63137976e-02, 2.80841090e-05, ...,
        0.00000000e+00, 0.00000000e+00, 1.69436919e-03],
       [1.83673320e-04, 6.01638405e-03, 1.75272471e-03, ...,
        8.83832281e-04, 0.00000000e+00, 4.31798728e-03],
       [0.00000000e+00, 0.00000000e+00, 0.00000000e+00, ...,
        0.00000000e+00, 9.47992584e-04, 1.09340709e-03]])
```

In [15]:

```
NMF_Model.components_ .shape
```

Out[15]:

```
(20, 8122)
```

Where the 20 indicates the clustering groupp(n_components) and 8122 indicates the vocabulary

To know what are the top 15 words in each topic, we can create a dataframe by using the below functions

In [16]:

```
topic_dict2 = {}
for index, topicprob in enumerate (NMF_Model.components_):
    emp_lst = []
    for element in topicprob.argsort()[-15:] :
        emp_lst.append( dtm_nmf.get_feature_names()[element] )

    topic_dict2[index]= emp_lst
```

In [17]:

```
topic_df2 = pd.DataFrame(topic_dict2)
topic_df2.columns=['Topic1','Topic2','Topic3','Topic4','Topic5','Topic6','Topic7','Topic8',
```

Top 15 topic keywords

In [18]:

```
topic_df2.head(15)
```

Out[18]:

	Topic1	Topic2	Topic3	Topic4	Topic5	Topic6	Topic7	Topic8	Topic9	T
0	company	user	company	investment	website	balance	college	language	fact	
1	college	search	world	job	web	death	olympics	example	lesser	a
2	visit	add	tell	website	machine	big	china	computer	mind	pres
3	app	interview	friend	internet	business	decision	spotify	science	want	
4	phone	easily	culture	free	computer	earth	happen	java	2017	h
5	song	post	website	make	hack	want	place	software	resolution	
6	friend	mark	man	start	want	thing	business	major	important	pres
7	website	delete	woman	com	code	real	company	engineering	year	
8	site	improvement	live	home	beginner	moment	job	main	happen	e

Now we will predict the topic for each rows in the dataset and assign its corresponding token

In [19]:

```
topic_results_2= NMF_Model.transform(dtm_nmf_mat)
```

In [20]:

```
df['Topic_ID_NMF'] = topic_results_2.argmax(axis=1)
```

In [21]:

```
df.head(10)
```

Out[21]:

	Question	processed_txt	Topic_ID_NMF
0	How is the training in TCS?	training TCS	18
1	How do you know if someone has blocked you on ...	know block messenger	8
2	What are the most amusing differences between ...	amusing difference british English american En...	14
3	What is happening when my eyes get heavy when ...	happen eye heavy sleep	9
4	Which is the best phone under 15000 currently?	good phone 15000 currently	0
5	Is time is precious, or are relationships prec...	time precious relationship precious	15
6	What would the consequences of a Brexit be for...	consequence Brexit UK citizen	12
7	How India can respond to the Uri terror attack?	India respond Uri terror attack	6
8	What are some of the most interesting facts ab...	interesting fact lion tiger	8
9	I'm going to kill myself. How should I do it?	go kill	11

In [22]:

```
from sklearn.decomposition import PCA
```

In [23]:

```
pca = PCA(n_components=3, random_state=42)
tred_viz_2 = pca.fit_transform(topic_results_2)
```

In [24]:

```
tred_viz_df_2= pd.DataFrame(tred_viz_2)
```

In [5]:

```
tred_viz_df_2
```

Out[5]:

	0	1	2	topic_nmf
0	-0.000510	-0.000993	-0.002570	18.0
1	0.003056	0.003853	-0.007147	8.0
2	-0.011435	-0.008016	-0.025309	14.0
3	0.003557	0.003777	-0.001403	9.0
4	-0.010749	-0.001441	0.008127	0.0
...	...	...	...	...
202059	0.002431	-0.002350	-0.000267	18.0
202060	-0.000194	-0.001027	-0.002268	10.0
202061	-0.000236	-0.001095	-0.002128	6.0
202062	-0.008954	-0.000706	0.005233	0.0
202063	0.000070	-0.000740	-0.003018	0.0

202064 rows × 4 columns

In [25]:

```
tred_viz_df_2['topic_nmf']=df['Topic_ID_NMF']
```

In [28]:

```
tred_viz_df_2.to_csv('3d_sc.csv',index=False) #for safe purposes
```

In [4]:

```
tred_viz_df_2= pd.read_csv('3d_sc.csv')
```

In [23]:

```
tred_viz_df_2
```

Out[23]:

	0	1	2	topic_nmf
0	-0.000510	-0.000993	-0.002570	18.0
1	0.003056	0.003853	-0.007147	8.0
2	-0.011435	-0.008016	-0.025309	14.0
3	0.003557	0.003777	-0.001403	9.0
4	-0.010749	-0.001441	0.008127	0.0
...	...	...	...	...
202059	0.002431	-0.002350	-0.000267	18.0
202060	-0.000194	-0.001027	-0.002268	10.0
202061	-0.000236	-0.001095	-0.002128	6.0
202062	-0.008954	-0.000706	0.005233	0.0
202063	0.000070	-0.000740	-0.003018	0.0

202064 rows × 4 columns

From the abvoe figure we can see that the NMF model does well in topic modelling

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: