In [1]:
```
#Import all the needed libraries
import pandas as pd  # data processing, CSV file I/O (e.g. pd.read_csv)
import numpy as np # linear algebra
import matplotlib.pyplot as plt
%matplotlib inline
import json
import seaborn as sns
import re
import collections
from wordcloud import WordCloud
```

```
C:\Users\92312\Anaconda3\lib\site-packages\statsmodels\tools\_testing.py:19: FutureWarning: pandas.util.testing is deprecated. Use
testing instead.
  import pandas.util.testing as tm
```

In [2]:
```
# https://pypi.org/project/tweet-preprocessor/
# https://github.com/5hirish/tweet_scrapper
# https://github.com/pablobarbera/pytwools
import preprocessor as p
#from preprocessor import *
#from preprocessor.api import set_options as set_options1
from preprocessor.api import clean as clean
# pip install tweet-preprocessor
# https://pypi.org/project/tweet-preprocessor/
from sklearn.feature_extraction.text import CountVectorizer
import nltk
import string
from pprint import pprint

# Gensim
import gensim
import gensim.corpora as corpora
from gensim.utils import simple_preprocess
from gensim.models import CoherenceModel

# spacy for lemmatization
import spacy

# Plotting tools
import pyLDAvis
import pyLDAvis.gensim  # don't skip this


# 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)
```

In [3]:
```
#Reading the raw data collected from the Twitter
tweets_data = []
tweets_data_path = 'd:\\semester 2\\03 DL\\Project\\all.txt'
tweets_file = open(tweets_data_path, "r")
for line in tweets_file:
    try:
        tweet = json.loads(line)
        tweets_data.append(tweet)
    except:
        continue
```

In [4]:
```
print('tweet variable length: ',len(tweet))
print('tweet_data variable length: ',len(tweets_data))
print('type of tweet variable: ',type(tweet))
print('type of tweet_data variable',type(tweets_data))
```

```
tweet variable length:  26
tweet_data variable length:  25911
type of tweet variable:  <class 'dict'>
type of tweet_data variable <class 'list'>
```

In [5]:
```
# converting tweets_data variable (of list type) to data frame
fulldata = pd.DataFrame(tweets_data)
```

In [6]:
```
print("{} samples with {} features ".format(*fulldata.shape))
```

```
25911 samples with 32 features
```

In [7]:
```
fulldata.head()
```

Out[7]:

	created_at	id	id_str	text	truncated	display_text_range	entities	metadata	
0	Fri May 22 18:01:13 +0000 2020	1263892686479777793	1263892686479777793	Alhamdulilah, i'm so proud of my son he took b...	False	[0, 273]	{'hashtags': [{'text': 'kashmirbleeds', 'indic...	{'iso_language_code': 'en', 'result_type': 're...	href="http://t
1	Fri May 22 17:59:47 +0000 2020	1263892327891980289	1263892327891980289	RT @nazir_lord: Indian Occupied #kashmir Distr...	False	[0, 140]	{'hashtags': [{'text': 'kashmir', 'indices': [...	{'iso_language_code': 'en', 'result_type': 're...	href="http://t
2	Fri May 22 17:58:29 +0000 2020	1263892001826750465	1263892001826750465	RT @KhaledBeydoun: Muslims are only newsworthy...	False	[0, 124]	{'hashtags': [{'text': 'kashmirbleeds', 'indic...	{'iso_language_code': 'en', 'result_type': 're...	href="http://t
3	Fri May 22 17:55:57 +0000 2020	1263891360945508353	1263891360945508353	RT @moazamhussain77: Every year he celebrates ...	False	[0, 140]	{'hashtags': [], 'symbols': [], 'user_mentions...	{'iso_language_code': 'en', 'result_type': 're...	href="http://t
4	Fri May 22 17:49:50 +0000 2020	1263889822596431874	1263889822596431874	RT @nazir_lord: Indian Occupied #kashmir Distr...	False	[0, 140]	{'hashtags': [{'text': 'kashmir', 'indices': [...	{'iso_language_code': 'en', 'result_type': 're...	href="http://t

5 rows × 32 columns

In [8]:
```
#tweets_data = fulldata.copy()
#tweets_data.shape
```

In [9]:
```
#Create a function to see if the tweet is a retweet
def is_RT(tweet):
    if 'retweeted_status' not in tweet:
        return False
    else:
        return True
```

In [10]:
```
#Create a function to see if the tweet is a reply to a tweet of #another user, if so return said user.
def is_Reply_to(tweet):
    if 'in_reply_to_screen_name' not in tweet:
        return False
    else:
        return tweet['in_reply_to_screen_name']
```

In [11]:
```
#Create function for taking the most used Tweet sources off the #source column
def reckondevice(tweet):
    if 'iPhone' in tweet['source'] or ('iOS' in tweet['source']):
        return 'iPhone'
    elif 'Android' in tweet['source']:
        return 'Android'
    elif 'Mobile' in tweet['source'] or ('App' in tweet['source']):
        return 'Mobile device'
    elif 'Mac' in tweet['source']:
        return 'Mac'
    elif 'Windows' in tweet['source']:
        return 'Windows'
    elif 'Bot' in tweet['source']:
        return 'Bot'
    elif 'Web' in tweet['source']:
        return 'Web'
    elif 'Instagram' in tweet['source']:
        return 'Instagram'
    elif 'Blackberry' in tweet['source']:
        return 'Blackberry'
    elif 'iPad' in tweet['source']:
        return 'iPad'
    elif 'Foursquare' in tweet['source']:
        return 'Foursquare'
    else:
        return '-'
```

In [12]:
```
#Convert the Tweet JSON data to a pandas Dataframe, and take the #desired fields from the JSON. More could be added if needed.
tweets = pd.DataFrame()
print('tweets shape',tweets.shape)
print('tweets type',type(tweets))
```

```
tweets shape (0, 0)
tweets type <class 'pandas.core.frame.DataFrame'>
```

In [13]:
```
#Convert the Tweet JSON data to a pandas Dataframe, and take the #desired fields from the JSON. More could be added if needed.
tweets['text'] = list(map(lambda tweet: tweet['text'] if 'extended_tweet' not in tweet else tweet['extended_tweet']['full_text'],
tweets['Username'] = list(map(lambda tweet: tweet['user']['screen_name'], tweets_data))
tweets['Timestamp'] = list(map(lambda tweet: tweet['created_at'], tweets_data))
tweets['lenght'] = list(map(lambda tweet: len(tweet['text']) if'extended_tweet' not in tweet else len(tweet['extended_tweet']['fu
tweets['location'] = list(map(lambda tweet: tweet['user']['location'], tweets_data))
tweets['device'] = list(map(reckondevice, tweets_data))
tweets['RT'] = list(map(is_RT, tweets_data))
tweets['Reply'] = list(map(is_Reply_to, tweets_data))
```

In [14]:
```
print('tweets shape',tweets.shape)
```

```
tweets shape (25911, 8)
```

In [15]:
```
tweets.head()
```

Out[15]:

	text	Username	Timestamp	lenght	location	device	RT	Reply
0	Alhamdulilah, i'm so proud of my son he took b...	mujeebfarooq7	Fri May 22 18:01:13 +0000 2020	273	Indian Occupied Kashmir	Android	False	None
1	RT @nazir_lord: Indian Occupied #kashmir Distr...	ateeqawan	Fri May 22 17:59:47 +0000 2020	140	England, United Kingdom	iPhone	True	None
2	RT @KhaledBeydoun: Muslims are only newsworthy...	shahrukhkadri20	Fri May 22 17:58:29 +0000 2020	124		Android	True	None
3	RT @moazamhussain77: Every year he celebrates ...	Politicsinworld	Fri May 22 17:55:57 +0000 2020	140		Android	True	None
4	RT @nazir_lord: Indian Occupied #kashmir Distr...	AMARKHAN3380	Fri May 22 17:49:50 +0000 2020	140		Android	True	None

In [16]:
```
#See the percentage of tweets from the initial set that are #retweets:
RT_tweets = tweets[tweets['RT'] == True]
print(f"The percentage of retweets is {round(len(RT_tweets)/len(tweets)*100)}% of all the tweets")
```

```
The percentage of retweets is 78% of all the tweets
```

In [17]:
```
#See the percentage of tweets from the initial set that are replies #to tweets of another user:
Reply_tweets = tweets[tweets['Reply'].apply(type) == str]
print(f"The percentage of retweets is {round(len(Reply_tweets)/len(tweets)*100)}% of all the tweets")
```

```
The percentage of retweets is 5% of all the tweets
```

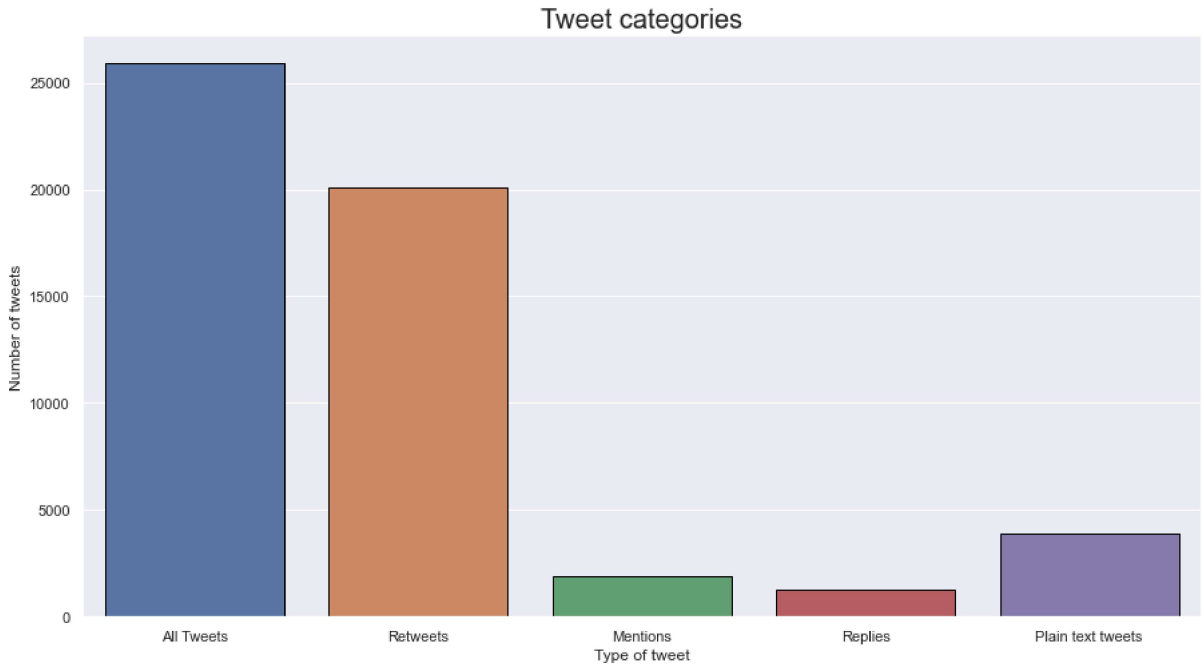
In [18]:
```
#See the percentage of tweets from the initial set that have #mentions and are not retweets:
mention_tweets = tweets[~tweets['text'].str.contains("RT")  & tweets['text'].str.contains("@")]
print(f"The percentage of retweets is {round(len(mention_tweets)/len(tweets)*100)}% of all the tweets")
```

```
The percentage of retweets is 7% of all the tweets
```

In [19]:
```
#See how many tweets inside are plain text tweets (No RT or mention)
plain_text_tweets = tweets[~tweets['text'].str.contains("@")  & ~tweets['text'].str.contains("RT")]
print(f"The percentage of retweets is {round(len(plain_text_tweets)/len(tweets)*100)}% of all the tweets")
```

```
The percentage of retweets is 15% of all the tweets
```

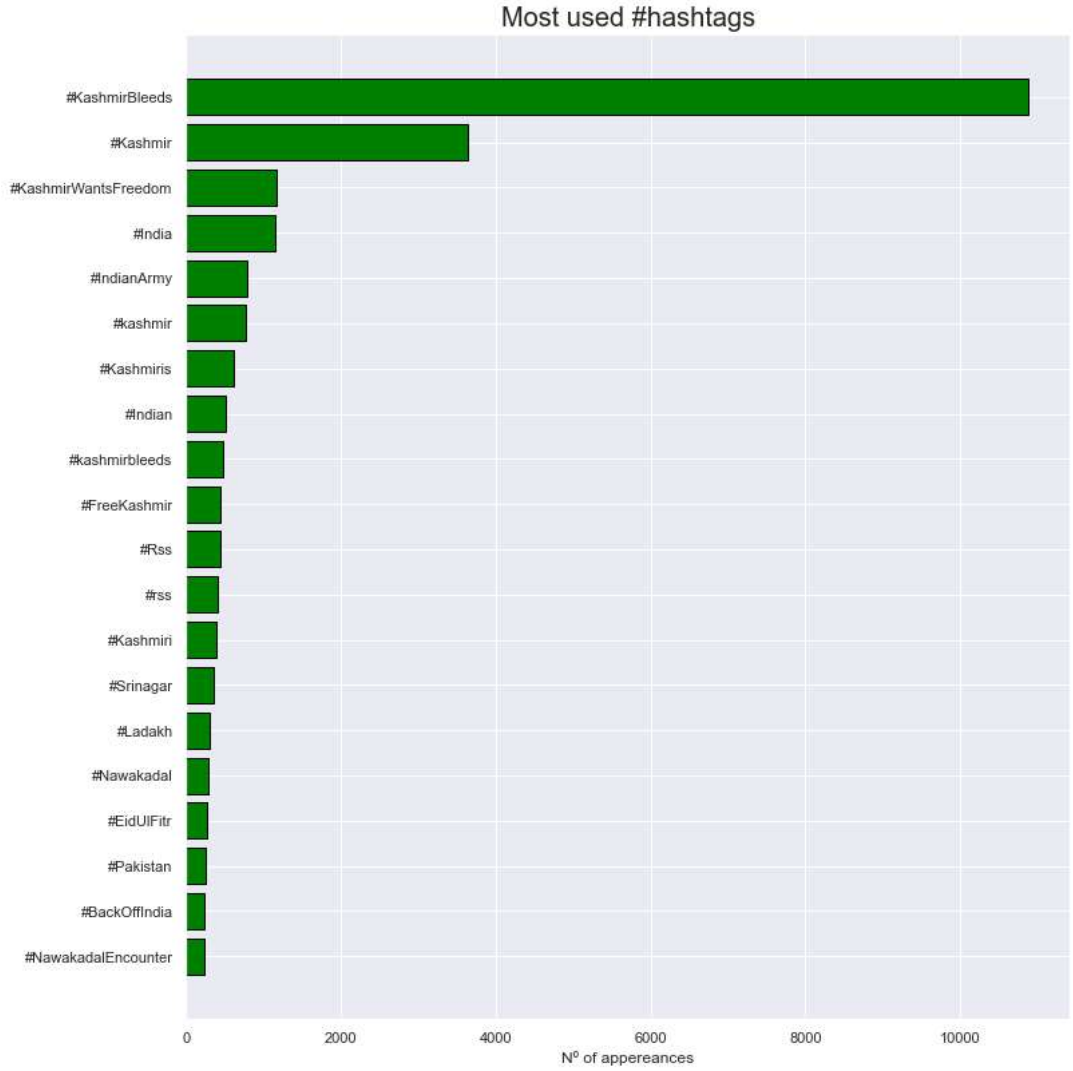
In [20]:
```
#Now we will plot all the different categories. Note that the reply #tweets are inside the mention tweets
len_list = [ len(tweets), len(RT_tweets),len(mention_tweets), len(Reply_tweets), len(plain_text_tweets)]
item_list = ['All Tweets','Retweets', 'Mentions', 'Replies', 'Plain text tweets']
plt.figure(figsize=(15,8))
sns.set(style="darkgrid")
plt.title('Tweet categories', fontsize = 20)
plt.xlabel('Type of tweet')
plt.ylabel('Number of tweets')
sns.barplot(x = item_list, y = len_list,  edgecolor = 'black', linewidth=1)

plt.show()
```



In [21]:
```
#To see the most used hashtags.
hashtags = []
hashtag_pattern = re.compile(r"#[a-zA-Z]+")
hashtag_matches = list(tweets['text'].apply(hashtag_pattern.findall))
hashtag_dict = {}
for match in hashtag_matches:
    for singlematch in match:
        if singlematch not in hashtag_dict.keys():
            hashtag_dict[singlematch] = 1
        else:
            hashtag_dict[singlematch] = hashtag_dict[singlematch]+1
```

In [22]:
```
#Making a list of the most used hashtags and their values
hashtag_ordered_list =sorted(hashtag_dict.items(), key=lambda x:x[1])
hashtag_ordered_list = hashtag_ordered_list[::-1]
#Separating the hashtags and their values into two different lists
hashtag_ordered_values = []
hashtag_ordered_keys = []
#Pick the 20 most used hashtags to plot
for item in hashtag_ordered_list[0:20]:
    hashtag_ordered_keys.append(item[0])
    hashtag_ordered_values.append(item[1])
```

In [23]:
```
#Plotting a graph with the most used hashtags
fig, ax = plt.subplots(figsize = (12,12))
y_pos = np.arange(len(hashtag_ordered_keys))
ax.barh(y_pos ,list(hashtag_ordered_values)[::-1], align='center', color = 'green', edgecolor = 'black', linewidth=1)
ax.set_yticks(y_pos)
ax.set_yticklabels(list(hashtag_ordered_keys)[::-1])
ax.set_xlabel("Nº of appereances")
ax.set_title("Most used #hashtags", fontsize = 20)
plt.tight_layout(pad=3)
plt.show()
```

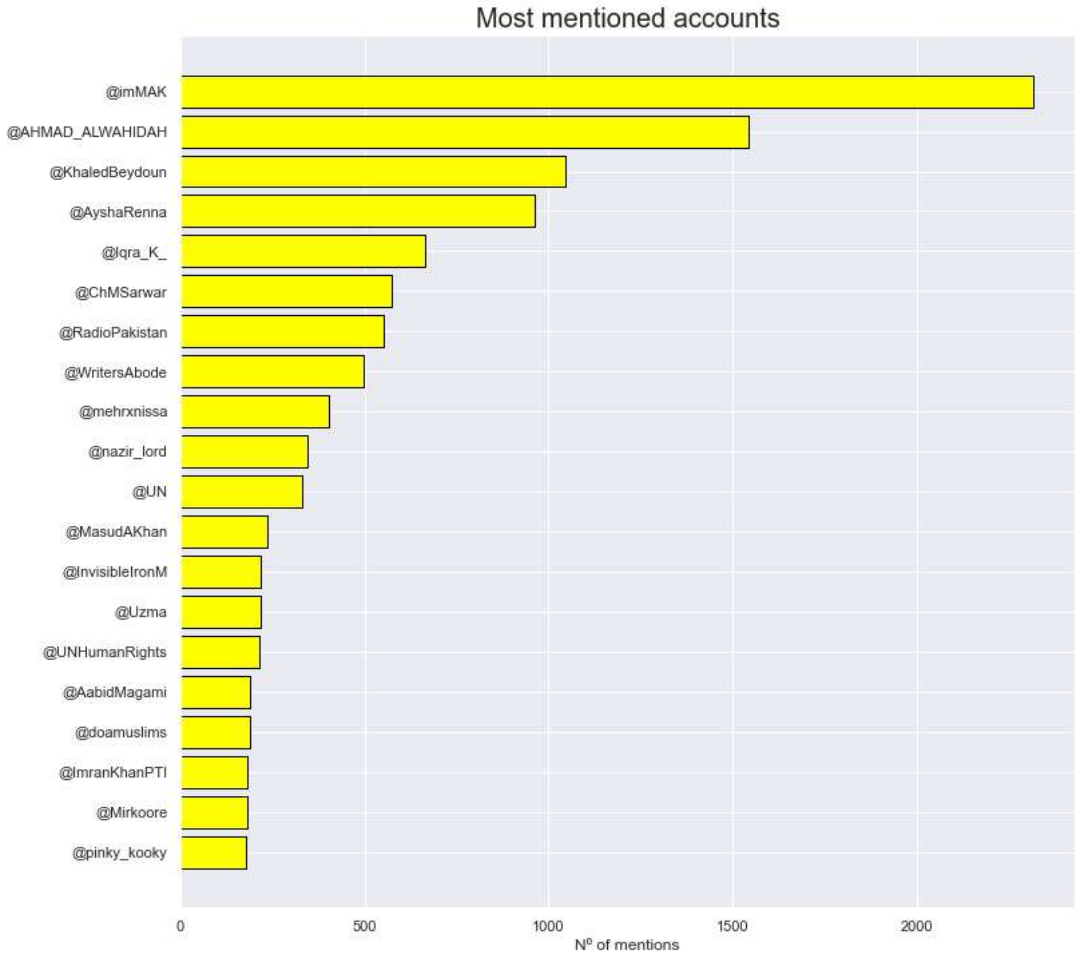


In [24]:
```
#Make a wordcloud plot of the most used hashtags, for this we need a #dictionary
#where the keys are the words and the values are the number of #appearances
hashtag_ordered_dict = {}
for item in hashtag_ordered_list[0:20]:
    hashtag_ordered_dict[item[0]] = item[1]
wordcloud = WordCloud(width=1000, height=1000, random_state=21, max_font_size=200, background_color = 'white').generate_from_freq
plt.figure(figsize=(15, 10))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off')

plt.show()
```

In [25]:
```
#Now we will do the same with the mentions:
mentions = []
mention_pattern = re.compile(r"@[a-zA-Z_]+")
mention_matches = list(tweets['text'].apply(mention_pattern.findall))
mentions_dict = {}
for match in mention_matches:
    for singlematch in match:
        if singlematch not in mentions_dict.keys():
            mentions_dict[singlematch] = 1
        else:
            mentions_dict[singlematch] = mentions_dict[singlematch]+1
```

In [26]:
```
#Create an ordered list of tuples with the most mentioned users and #the number of times they have been mentioned
mentions_ordered_list =sorted(mentions_dict.items(), key=lambda x:x[1])
mentions_ordered_list = mentions_ordered_list[::-1]
#Pick the 20 top mentioned users to plot and separate the previous #list into two list: one with the users and one with the value
mentions_ordered_values = []
mentions_ordered_keys = []
for item in mentions_ordered_list[0:20]:
    mentions_ordered_keys.append(item[0])
    mentions_ordered_values.append(item[1])
```

In [27]:
```
fig, ax = plt.subplots(figsize = (12,12))
y_pos = np.arange(len(mentions_ordered_values))
ax.barh(y_pos ,list(mentions_ordered_values)[::-1], align='center', color = 'yellow', edgecolor = 'black', linewidth=1)
ax.set_yticks(y_pos )
ax.set_yticklabels(list(mentions_ordered_keys)[::-1])
ax.set_xlabel("Nº of mentions")
ax.set_title("Most mentioned accounts", fontsize = 20)

plt.show()
```



In [28]:
```
#Make a wordcloud representation for the most mentioned accounts too
mentions_ordered_dict = {}
for item in mentions_ordered_list[0:20]:
    mentions_ordered_dict[item[0]] = item[1]
wordcloud = WordCloud(width=1000, height=1000, random_state=21, max_font_size=200, background_color = 'white').generate_from_freq
plt.figure(figsize=(15, 10))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off')

plt.show()
```

In [149]:
```
# Filtering
Retweets_16 = fulldata.loc[(fulldata['retweet_count'] > 1) & (fulldata['favorite_count'] > 1)]
Retweets_16.shape
```

Out[149]: (873, 32)

In [150]:
```
# Filtering
Retweets_15 = fulldata.loc[(fulldata['retweet_count'] >= 2)]
Retweets_15.shape
```

Out[150]: (19991, 32)

In [151]:
```
Retweets_15.head(5)
```

Out[151]:

	created_at	id	id_str	text	truncated	display_text_range	entities	metadata	
1	Fri May 22 17:59:47 +0000 2020	1263892327891980289	1263892327891980289	RT @nazir_lord: Indian Occupied #kashmir Distr...	False	[0, 140]	{'hashtags': [{'text': 'kashmir', 'indices': [...	{'iso_language_code': 'en', 'result_type': 're...	href="http://t
2	Fri May 22 17:58:29 +0000 2020	1263892001826750465	1263892001826750465	RT @KhaledBeydoun: Muslims are only newsworthy...	False	[0, 124]	{'hashtags': [{'text': 'kashmirbleeds', 'indic...	{'iso_language_code': 'en', 'result_type': 're...	href="http://t
3	Fri May 22 17:55:57 +0000 2020	1263891360945508353	1263891360945508353	RT @moazamhussain77: Every year he celebrates ...	False	[0, 140]	{'hashtags': [], 'symbols': [], 'user_mentions...	{'iso_language_code': 'en', 'result_type': 're...	href="http://t
4	Fri May 22 17:49:50 +0000 2020	1263889822596431874	1263889822596431874	RT @nazir_lord: Indian Occupied #kashmir Distr...	False	[0, 140]	{'hashtags': [{'text': 'kashmir', 'indices': [...	{'iso_language_code': 'en', 'result_type': 're...	href="http://t
6	Fri May 22 17:44:36 +0000 2020	1263888504905437191	1263888504905437191	RT @nazir_lord: Indian Occupied #kashmir Distr...	False	[0, 140]	{'hashtags': [{'text': 'kashmir', 'indices': [...	{'iso_language_code': 'en', 'result_type': 're...	href="http://t

5 rows × 32 columns

In [152]:
```
# extracting column name text
tweetsData = pd.DataFrame(Retweets_15, columns=['text'])
tweetsData.shape
```

Out[152]: (19991, 1)

In [153]:
```
tweetsData = tweetsData.dropna()
tweetsData = tweetsData.drop_duplicates(subset='text', keep="first")
tweetsData.shape
```

Out[153]: (2434, 1)

In [154]:
```
#Reserved words (RT, FAV)
p.set_options(p.OPT.URL, p.OPT.EMOJI, p.OPT.MENTION, p.OPT.HASHTAG, p.OPT.SMILEY, p.OPT.RESERVED)#, p.OPT.NUMBER)
```

In [155]:
```
tweetsData.head()
```

Out[155]:

	text
1	RT @nazir_lord: Indian Occupied #kashmir Distr...
2	RT @KhaledBeydoun: Muslims are only newsworthy...
3	RT @moazamhussain77: Every year he celebrates ...
12	RT @BaaghiTV: Foreign Minister phone call to U...
14	RT @thedowntowner_: #Kashmir #NoEIDinBleedingK...

In [156]:
```
# getting rid of @user
clean_text = pd.DataFrame(tweetsData['text'].apply(lambda x: p.clean(x)))
clean_text.shape
```

Out[156]: (2434, 1)

In [157]:
```
clean_text_copy = clean_text.copy()
clean_text_copy.shape # should be same as clean_text.shape
```

Out[157]: (2434, 1)

In [158]:
```
# dropping duplicates
clean_text_copy = clean_text_copy.drop_duplicates(subset='text', keep="first")
clean_text_copy.shape
```

Out[158]: (2315, 1)

In [159]:
```
clean_text_copy.head()
```

Out[159]:

	text
1	: Indian Occupied Distraction of civilian home...
2	: Muslims are only newsworthy when they're vil...
3	: Every year he celebrates "Quds-day" demonstr...
12	: Foreign Minister phone call to UN Secretary ...
14	: SHAHEED ZINDA HAI (May Allah (swt) rest the ...

In [160]:
```
clean_text_copy.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2315 entries, 1 to 25904
Data columns (total 1 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   text    2315 non-null   object
dtypes: object(1)
memory usage: 36.2+ KB
```

Pre-processing text data

Most of the text data are cleaned by following below steps.

Remove punctuations
Tokenization - Converting a sentence into list of words
Remove stopwords
Lammetization/stemming - Tranforming any form of a word to its root word

In [161]:
```
# get a word count per sentence column
def word_count(sentence):
    return len(sentence.split())

clean_text_copy['word_count'] = clean_text_copy['text'].apply(word_count)
clean_text_copy['word_count'].shape
```

Out[161]: (2315,)

In [162]:
```
clean_text_copy.head(5)
```

Out[162]:

	text	word_count
1	: Indian Occupied Distraction of civilian home...	20
2	: Muslims are only newsworthy when they're vil...	10
3	: Every year he celebrates "Quds-day" demonstr...	18
12	: Foreign Minister phone call to UN Secretary ...	15
14	: SHAHEED ZINDA HAI (May Allah (swt) rest the ...	15

In [163]:
```
# Get a bool series representing which row satisfies the condition i.e. True for
# row in which value of 'Age' column is more than 30
minthreewords = clean_text_copy.apply(lambda x: True if x['word_count'] > 3 else False , axis=1)

# Count number of True in series
numOfRows = len(minthreewords[minthreewords == True].index)

print('Number of Rows in dataframe in which Age > 3 : ', numOfRows)
```

```
Number of Rows in dataframe in which Age > 3 :  2278
```

In [164]:
```
minthreewords.shape
```

Out[164]: (2315,)

In [165]:
```
minthreewords.head(5)
```

Out[165]:
```
1     True
2     True
3     True
12    True
14    True
dtype: bool
```

In [166]:
```
selected_tweets_morethanthree = clean_text_copy.loc[clean_text_copy['word_count'] > 3]
selected_tweets_morethanthree.shape
```

Out[166]:
```
(2278, 2)
```

In [167]:
```
selected_tweets_morethanthree.head()
```

Out[167]:

	text	word_count
1	: Indian Occupied Distraction of civilian home...	20
2	: Muslims are only newsworthy when they're vil...	10
3	: Every year he celebrates "Quds-day" demonstr...	18
12	: Foreign Minister phone call to UN Secretary ...	15
14	: SHAHEED ZINDA HAI (May Allah (swt) rest the ...	15

In [168]:
```
#  Tokenize words and Clean-up text
# Let’s tokenize each sentence into a list of words, removing punctuations and unnecessary characters altogether.
# Gensim’s simple_preprocess() is great for this. Additionally I have set deacc=True to remove the punctuations.
data = list(selected_tweets_morethanthree.text)
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])
print(f"length of data_words: {len(data_words)}")
```

```
[['indian', 'occupied', 'distraction', 'of', 'civilian', 'homes', 'in', 'srinagar', 'is', 'plan', 'for', 'illegal', 'settlements'
length of data_words: 2278
```

In [169]:
```
# Creating Bigram and Trigram Models
# Bigrams are two words frequently occurring together in the document. Trigrams are 3 words frequently occurring.
# Gensim’s Phrases model can build and implement the bigrams, trigrams, quadgrams and more. The two important arguments to Phrase
# The higher the values of these param, the harder it is for words to be combined to bigrams.

# Build the bigram and trigram models
# 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[0]]])
```

```
['indian', 'occupied', 'distraction', 'of', 'civilian', 'homes', 'in', 'srinagar', 'is', 'plan', 'for', 'illegal', 'settlements',
```

In [170]:
```
#  Remove Stopwords, Make Bigrams and Lemmatize
# The bigrams model is ready.
# Let’s define the functions to remove the stopwords, make bigrams and lemmatization and call them sequentially.
# NLTK Stop words
nltk.download('stopwords')
stop_words = nltk.corpus.stopwords.words('english')
def remove_stopwords(texts):
    return [[word for word in simple_preprocess(str(doc)) if word not in stop_words] for doc in texts]

def make_bigrams(texts):
    return [bigram_mod[doc] for doc in texts]

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

def lemmatization(texts, allowed_postags=['NOUN', 'ADJ', 'VERB', 'ADV']):
    """https://spacy.io/api/annotation"""
    texts_out = []
    for sent in texts:
        doc = nlp(" ".join(sent))
        texts_out.append([token.lemma_ for token in doc if token.pos_ in allowed_postags])
    return texts_out
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\92312\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

In [171]:
```
# Let's call the functions in order.

# Remove Stop Words
data_words_nostops = remove_stopwords(data_words)
print(f"length of data_words_nostops: {len(data_words_nostops)}")
```

```
length of data_words_nostops: 2278
```

In [172]:
```
# Form Bigrams
data_words_bigrams = make_bigrams(data_words_nostops)
print(f"length of data_words_bigrams: {len(data_words_bigrams)}")
```

```
length of data_words_bigrams: 2278
```

In [173]:
```
# Initialize spacy 'en' model, keeping only tagger component (for efficiency)
# python3 -m spacy download en
import spacy

# python3 -m spacy download en_core_web_sm
# pip3 install https://github.com/explosion/spacy-models/releases/download/en_core_web_sm-2.2.0/en_core_web_sm-2.2.0.tar.gz
#pip install --user en_core_web_sm
#import en_core_web_sm
#nlp = en_core_web_sm.load()

# I write the full name rather than en
nlp = spacy.load('en_core_web_sm',disable=['parser','ner']) # original loads spacy library
#nlp = spacy.load('en',disable=['parser','ner'])
#nlp = spacy.load('en',parse=True,tag=True, entity=True)
# was getting this error "spacy Can't find model 'en_core_web_sm' on windows 10 and Python 3.5.3 :: Anaconda custom (64-bit)"
# solution:
# https://stackoverflow.com/questions/54334304/spacy-cant-find-model-en-core-web-sm-on-windows-10-and-python-3-5-3-anacon
#nlp = spacy.load(r'C:\Users\92312\Anaconda3\Lib\site-packages\en_core_web_sm\en_core_web_sm-2.2.0',disable=['parser','ner'])
```

```
C:\Users\92312\AppData\Roaming\Python\Python37\site-packages\spacy\util.py:271: UserWarning: [W031] Model 'en_core_web_sm' (2.2.0
th the current spaCy version (2.3.0). This may lead to unexpected results or runtime errors. To resolve this, download a newer cor
with the current spaCy version. For more details and available updates, run: python -m spacy validate
  warnings.warn(warn_msg)
```

In [174]:
```
# Do lemmatization keeping only noun, adj, vb, adv
data_lemmatized = lemmatization(data_words_bigrams, allowed_postags=['NOUN', 'ADJ', 'VERB', 'ADV'])
print(f"length of data_lemmatized: {len(data_lemmatized)}")
```

```
length of data_lemmatized: 2278
```

In [175]:
```
print(data_lemmatized[:1])
```

```
[['occupy', 'distraction', 'civilian', 'home', 'srinagar', 'illegal', 'settlement']]
```

In [176]:
```
# Create the Dictionary and Corpus needed for Topic Modeling
# The two main inputs to the LDA topic model are the dictionary(id2word) and the corpus. Let's create them.

# Create Dictionary
id2word = corpora.Dictionary(data_lemmatized)

# Create Corpus
texts = data_lemmatized

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

# View
```

In [177]:
```
print(f"length of id2word: {len(id2word)}")
print(f"length of texts: {len(texts)}")
print(f"length of corpus: {len(corpus)}")
print(corpus[:1])
```

```
length of id2word: 2655
length of texts: 2278
length of corpus: 2278
[[(0, 1), (1, 1), (2, 1), (3, 1), (4, 1), (5, 1), (6, 1)]]
```

In [178]:
```
id2word[1]
```

Out[178]:
```
'distraction'
```

In [179]:
```
# Human readable format of corpus (term-frequency)
[[(id2word[id], freq) for id, freq in cp] for cp in corpus[:100]]
```

Out[179]:
```
[[('civilian', 1),
  ('distraction', 1),
  ('home', 1),
  ('illegal', 1),
  ('occupy', 1),
  ('settlement', 1),
  ('srinagar', 1)],
 [('newsworthy', 1), ('victim', 1), ('villain', 1)],
 [('away', 1),
  ('celebrate', 1),
  ('demonstrate', 1),
  ('everyday', 1),
  ('live', 1),
  ('mile', 1),
  ('palestinian', 1),
  ('peo', 1),
  ('people', 1),
  ('support', 1),
  ('year', 1)],
 [('amp', 1), ('phone', 1), ('serious', 1), ('situation', 1)],
 [],
 [('bus', 1),
  ('choose', 1),
  ('dear', 1),
  ('indian', 1),
  ('long', 1),
  ('sp', 1),
  ('throw', 1)],
 [('clean', 1),
  ('conscience', 1),
  ('heart', 1),
  ('kind', 1),
  ('leader', 1),
  ('need', 1),
  ('solve', 1),
  ('unending', 1),
  ('warfare', 1)],
 [('amp', 1), ('phone', 1), ('serious', 1), ('situation', 1)],
 [('burn', 1),
  ('family', 2),
  ('house', 1),
  ('include', 1),
  ('life', 1),
  ('lose', 1),
  ('saving', 1)],
 [('people', 2),
  ('amp', 1),
  ('indian', 1),
  ('act', 1),
  ('cruel', 1),
  ('homeless', 1),
  ('make', 1),
  ('way', 1)],
 [('heart', 1),
  ('arm', 1),
  ('back', 1),
  ('bassim', 1),
  ('call', 1),
  ('dead', 1),
  ('emanate', 1),
  ('go', 1),
  ('hold', 1),
  ('mot', 1),
  ('silent', 1),
  ('son', 1)],
 [('civilian', 1), ('force', 1), ('rescue', 1), ('security', 1), ('tho', 1)],
 [('amp', 1),
  ('situation', 1),
  ('actual', 1),
  ('come', 1),
  ('concoct', 1),
  ('ground', 1),
  ('journalist', 1),
  ('reflect', 1),
  ('stop', 1),
  ('want', 1)],
 [('want', 1), ('death', 1), ('feel', 1), ('kill', 1), ('terror', 1)],
 [('picture', 1)],
 [('dear', 1),
  ('break', 1),
  ('humanity', 1),
  ('organisation', 1),
  ('save', 1),
  ('silence', 1),
  ('step', 1),
  ('world', 1)],
```

```
[('click', 1), ('full', 1), ('link', 1), ('video', 1), ('watch', 1)],
[('access', 1), ('black', 1), ('month', 1)],
[('people', 1),
 ('kill', 1),
 ('country', 1),
 ('innocent', 1),
 ('terrorist', 1)],
[('burn', 1),
 ('go', 1),
 ('basim', 1),
 ('die', 1),
 ('injury', 1),
 ('last', 1),
 ('remind', 1),
 ('succumbed', 1),
 ('word', 1)],
[('do', 2), ('friend', 2), ('painful', 1), ('pal', 1), ('state', 1)],
[('dear', 1),
 ('heart', 1),
 ('arm', 1),
 ('bassim', 2),
 ('call', 1),
 ('dead', 1),
 ('emanate', 1),
 ('go', 1),
 ('hold', 1),
 ('silent', 1),
 ('son', 1),
 ('intonate', 1),
 ('mother', 1),
 ('remember', 1),
 ('shafiq', 1)],
[('picture', 1), ('speak', 1), ('volume', 1)],
[('want', 1), ('death', 1), ('feel', 1), ('kill', 1), ('terror', 1)],
[('arrogance', 1), ('unbelievable', 1)],
[('amp', 1),
 ('situation', 1),
 ('actual', 1),
 ('come', 1),
 ('concoct', 1),
 ('ground', 1),
 ('journalist', 1),
 ('reflect', 1),
 ('stop', 1),
 ('want', 1),
 ('fake', 1),
 ('report', 1)],
[('word', 1),
 ('allow', 2),
 ('beat', 1),
 ('even', 1),
 ('meet', 1),
 ('publicly', 1),
 ('utter', 1)],
[('picture', 1), ('speak', 1), ('volume', 1)],
[('deep', 1),
 ('freedom', 1),
 ('inside', 1),
 ('line', 1),
 ('palm', 1),
 ('right', 1),
 ('run', 1),
 ('vein', 1),
 ('write', 1)],
[('burn', 1), ('family', 1), ('house', 1), ('relate', 1), ('trend', 1)],
[('srinagar', 1),
 ('indian', 1),
 ('force', 1),
 ('stop', 1),
 ('humanity', 1),
 ('die', 1),
 ('blood', 1),
 ('pool', 1),
 ('time', 1),
 ('turn', 1)],
[],
[('victim', 1),
 ('live', 1),
 ('innocent', 1),
 ('forever', 1),
 ('forget', 1),
 ('let', 1),
 ('memo', 1),
 ('never', 1),
 ('single', 1)],
[('admit', 1),
```

```
 ('also', 1),
 ('bad', 1),
 ('fateful', 1),
 ('gut', 1),
 ('happen', 1),
 ('much', 1),
 ('night', 1),
 ('say', 1)],
[('dear', 1),
 ('come', 1),
 ('picture', 1),
 ('time', 1),
 ('anticipate', 1),
 ('condition', 1),
 ('disturb', 1),
 ('peace', 1)],
[('go', 1),
 ('word', 1),
 ('allow', 2),
 ('beat', 1),
 ('even', 1),
 ('meet', 1),
 ('publicly', 1),
 ('utter', 1),
 ('happen', 1),
 ('know', 1),
 ('pray', 1)],
[('civilian', 1),
 ('force', 1),
 ('rescue', 1),
 ('security', 1),
 ('tweeting', 1)],
[('life', 1),
 ('make', 1),
 ('day', 1),
 ('hell', 1),
 ('operation', 1),
 ('search', 1)],
[('feel', 2),
 ('care', 1),
 ('empathize', 1),
 ('incorrigibility', 1),
 ('sorry', 1),
 ('still', 1),
 ('strange', 1)],
[('end', 1), ('lockdown', 1)],
[('indian', 1),
 ('lose', 1),
 ('force', 1),
 ('eye', 1),
 ('shoot', 1),
 ('story', 1),
 ('vision', 1)],
[('home', 1),
 ('burn', 1),
 ('force', 1),
 ('hofe', 1),
 ('red', 1),
 ('reduce', 1),
 ('rubble', 1),
 ('safe', 1),
 ('stay', 2)],
[('live', 1),
 ('death', 1),
 ('innocent', 1),
 ('day', 1),
 ('end', 1),
 ('begin', 1),
 ('dawn', 1),
 ('passi', 1),
 ('place', 1),
 ('rule', 1),
 ('suffer', 1),
 ('tyrant', 1)],
[('support', 1),
 ('day', 1),
 ('oppress', 1),
 ('oppressor', 1),
 ('universal', 1)],
[('year', 1),
 ('hold', 1),
 ('security', 1),
 ('last', 1),
 ('counci', 1),
 ('direct', 1),
 ('honest', 1),
```

```
 ('plebiscite', 1),
 ('refuse', 1)],
[('home', 1), ('oppressor', 1), ('bloody', 1), ('destroy', 1)],
[('burn', 1),
 ('family', 1),
 ('house', 1),
 ('lose', 1),
 ('relate', 1),
 ('trend', 1)],
[('kind', 1),
 ('breaker', 1),
 ('circuit', 1),
 ('diminish', 1),
 ('diplomatic', 1),
 ('hard', 1),
 ('see', 1),
 ('violence', 1)],
[('away', 1),
 ('mile', 1),
 ('still', 1),
 ('city', 1),
 ('erupt', 1),
 ('literally', 1),
 ('main', 1),
 ('quite', 1),
 ('sargodha', 1),
 ('street', 1)],
[('away', 1),
 ('celebrate', 1),
 ('demonstrate', 1),
 ('everyday', 1),
 ('live', 1),
 ('mile', 1),
 ('palestinian', 1),
 ('people', 2),
 ('support', 1),
 ('year', 1),
 ('kill', 1),
 ('never', 1),
 ('bat', 1),
 ('eyelid', 1),
 ('get', 1),
 ('maim', 1),
 ('neighbourhood', 1)],
[('dear', 1),
 ('house', 1),
 ('come', 1),
 ('kill', 1),
 ('picture', 1),
 ('allow', 1),
 ('time', 1),
 ('anticipate', 1),
 ('condition', 1),
 ('disturb', 1),
 ('peace', 1),
 ('atleast', 1),
 ('blind', 1),
 ('enough', 1),
 ('fire', 1),
 ('horrible', 1),
 ('judiciary', 1),
 ('law', 1),
 ('loot', 1),
 ('set', 1)],
[('amp', 1),
 ('eye', 1),
 ('abuse', 1),
 ('conflict', 1),
 ('ever', 1),
 ('relation', 1)],
[('burn', 1),
 ('come', 1),
 ('decoration', 1),
 ('happy', 1),
 ('holiday', 1),
 ('wish', 1)],
[('crime', 1), ('culture', 1), ('look', 1), ('muslim', 1), ('seem', 1)],
[('home', 1),
 ('indian', 1),
 ('state', 1),
 ('destroy', 1),
 ('anger', 1),
 ('gunfight', 1),
 ('leave', 1),
 ('rise', 1),
 ('trail', 1)],
```

```
[('support', 1),
 ('day', 1),
 ('oppress', 1),
 ('oppressor', 1),
 ('universal', 1)],
[('day', 2), ('happiness', 1), ('sorrowful', 1)],
[('home', 1),
 ('indian', 1),
 ('state', 1),
 ('destroy', 1),
 ('anger', 1),
 ('gunfight', 1),
 ('leave', 1),
 ('rise', 1),
 ('trail', 1)],
[('speak', 1)],
[('year', 1),
 ('hold', 1),
 ('last', 1),
 ('direct', 1),
 ('honest', 1),
 ('plebiscite', 1),
 ('refuse', 1),
 ('resolution', 1)],
[('freedom', 1), ('right', 1)],
[('freedom', 1), ('right', 1)],
[('day', 2), ('happiness', 1), ('sorrowful', 1)],
[('stay', 1), ('purpose', 1)],
[],
[('word', 1),
 ('know', 1),
 ('actually', 1),
 ('ofcourse', 1),
 ('sell', 1),
 ('shame', 1),
 ('soul', 1)],
[('end', 1), ('lockdown', 1)],
[('break', 1),
 ('day', 1),
 ('stay', 1),
 ('big', 1),
 ('connected', 1),
 ('restore', 1)],
[('stay', 1), ('purpose', 1)],
[('indian', 1),
 ('force', 1),
 ('kill', 2),
 ('innocent', 1),
 ('boy', 1),
 ('brutality', 1),
 ('crpf', 1),
 ('today', 1),
 ('young', 1)],
[('victim', 1),
 ('stop', 1),
 ('say', 1),
 ('car', 1),
 ('security_official', 1),
 ('signal', 1),
 ('soldier', 1),
 ('tell', 1)],
[('people', 1),
 ('kill', 1),
 ('world', 1),
 ('terrorist', 1),
 ('big', 1),
 ('extremist_gang', 1),
 ('hold_accountable', 1),
 ('movement', 1)],
[('people', 1),
 ('kill', 1),
 ('world', 1),
 ('terrorist', 1),
 ('big', 1),
 ('extremist_gang', 1),
 ('hold_accountable', 1),
 ('movement', 1)],
[('legislation', 1),
 ('new', 1),
 ('piece', 1),
 ('protest', 1),
 ('roll', 1),
 ('vociferously', 1)],
[('victim', 1), ('basim_aijaz', 1), ('late', 1)],
[('indian', 1),
 ('force', 1),
```

```
 ('kill', 2),
 ('innocent', 1),
 ('boy', 1),
 ('brutality', 1),
 ('crpf', 1),
 ('young', 1)],
[('blood', 1),
 ('day', 1),
 ('muslim', 1),
 ('attack', 1),
 ('bcz', 1),
 ('priceless', 1),
 ('shed', 1),
 ('week', 1)],
[('body', 1), ('wound', 1)],
[('do', 2), ('friend', 2), ('painful', 1), ('state', 1)],
[('year', 1),
 ('force', 1),
 ('fire', 1),
 ('funeral', 1),
 ('open', 1),
 ('procession', 1)],
[('need', 3), ('help', 1), ('king', 1), ('warrior', 1)],
[('roofless', 1), ('talk', 1)],
[('say', 1),
 ('muslim', 1),
 ('legislation', 1),
 ('new', 1),
 ('piece', 1),
 ('protest', 1),
 ('roll', 1),
 ('vociferously', 1),
 ('alter', 1),
 ('israeli', 1),
 ('page', 1),
 ('playbook', 1),
 ('region', 1),
 ('straight', 1)],
[('also', 1), ('eliminate', 1), ('fall', 1), ('regime', 1), ('utterly', 1)],
[],
[('break', 1),
 ('day', 1),
 ('stay', 1),
 ('big', 1),
 ('connected', 1),
 ('restore', 1)],
[('injury', 1),
 ('succumbed', 1),
 ('basim_aijaz', 1),
 ('funeral', 1),
 ('collapse', 1),
 ('prayer', 1),
 ('receive', 1),
 ('site', 1)],
[('life', 1),
 ('right', 1),
 ('free', 1),
 ('guide', 1),
 ('hate', 1),
 ('mind', 1),
 ('path', 1)],
[('hold', 1),
 ('kill', 1),
 ('innocent', 1),
 ('brutally', 1),
 ('date', 1),
 ('hundred', 1)],
[('crime', 1), ('culture', 1), ('look', 1), ('muslim', 1), ('seem', 1)],
[('mercy', 1), ('war', 1)],
[('burn', 1),
 ('house', 1),
 ('loot', 1),
 ('allege', 1),
 ('arson', 1),
 ('human', 1),
 ('local', 1),
 ('ransack', 1),
 ('resident', 1),
 ('shield', 1),
 ('use', 1)],
[('srinagar', 1),
 ('people', 1),
 ('lose', 1),
 ('kill', 1),
 ('damage', 1),
 ('least', 1),
```

```
 ('militant', 1),
 ('residential_house', 1)],
[('newsworthy', 1), ('victim', 1), ('villain', 1)],
[('heart', 1),
 ('get', 1),
 ('broken', 1),
 ('doda', 1),
 ('http', 1),
 ('matyrdom', 1)],
[('allow', 1),
 ('admit', 1),
 ('also', 1),
 ('bad', 2),
 ('fateful', 1),
 ('gut', 1),
 ('happen', 2),
 ('much', 1),
 ('night', 1),
 ('say', 1),
 ('reason', 1)],
[('dear', 1),
 ('picture', 1),
 ('claim', 1),
 ('clearly', 1),
 ('evident', 1),
 ('government', 1),
 ('normal', 1)],
[('cleanse', 1),
 ('colonize', 1),
 ('ethnically', 1),
 ('introduce', 1),
 ('ple', 1)],
[('victim', 1),
 ('die', 1),
 ('conflict', 1),
 ('basim_aijaz', 1),
 ('late', 1),
 ('bloom', 1),
 ('due', 1),
 ('flower', 1),
 ('injure', 1),
 ('martyr', 1),
 ('spring', 1)],
[('humanity', 1), ('follower', 1), ('request', 1), ('sake', 1), ('stand', 1)]]
```

In [180]:
```
# Build LDA model
lda_model = gensim.models.ldamodel.LdaModel(corpus=corpus,
                                            id2word=id2word,
                                            num_topics=3,
                                            random_state=100,
                                            update_every=1,
                                            chunksize=100,
                                            passes=10,
                                            alpha='auto',
                                            per_word_topics=True)
```

In [181]:
```
# Print the Keyword in the 10 topics
pprint(lda_model.print_topics())
doc_lda = lda_model[corpus]
print(f"length of doc_lda: {len(doc_lda)}")
```

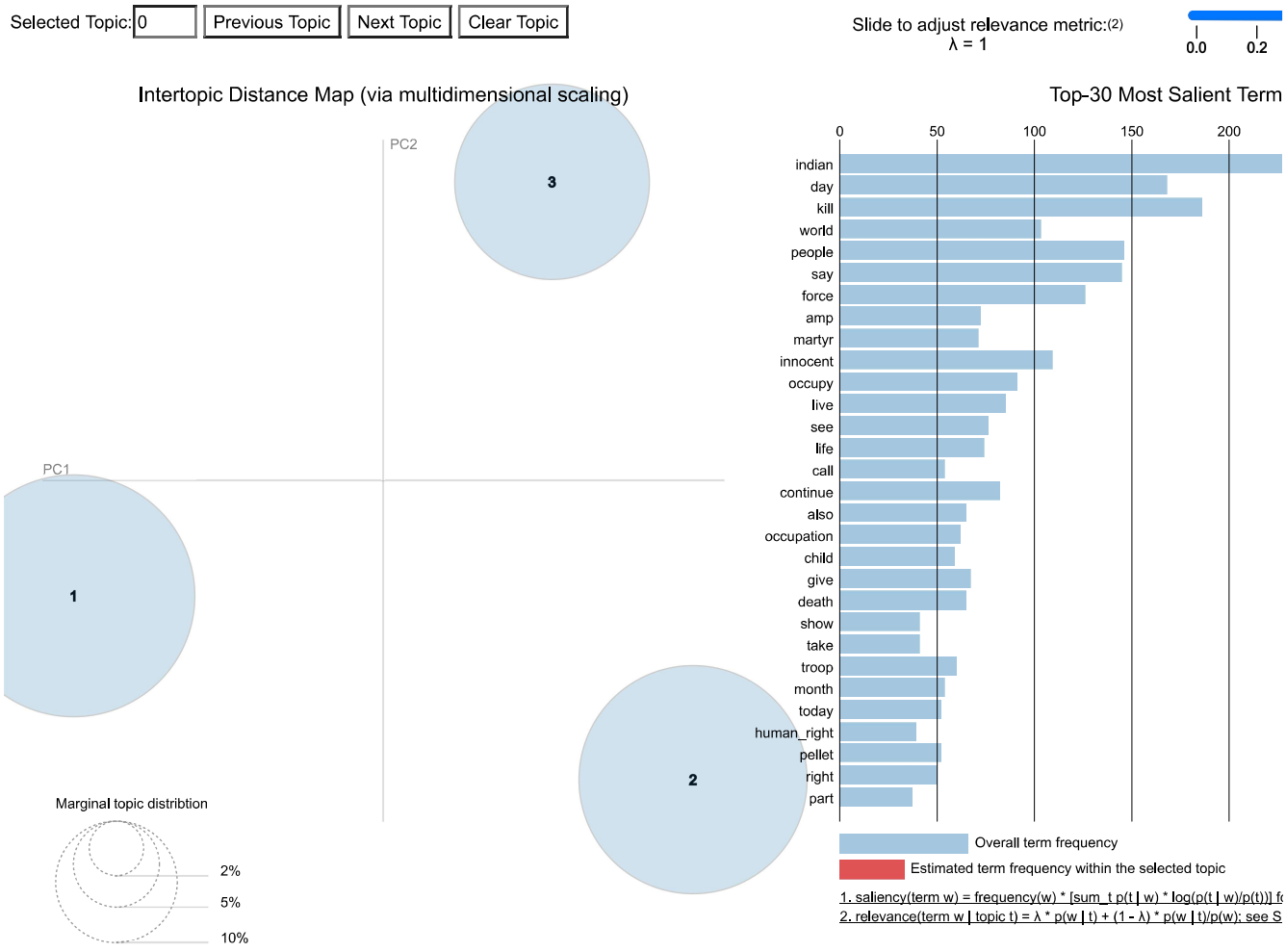
```
[(0,
  '0.026*"world" + 0.018*"amp" + 0.018*"martyr" + 0.014*"call" + 0.010*"show" '
  '+ 0.010*"take" + 0.010*"human_right" + 0.009*"part" + 0.009*"humanity" + '
  '0.008*"save"'),
 (1,
  '0.050*"indian" + 0.031*"kill" + 0.024*"people" + 0.024*"say" + '
  '0.021*"force" + 0.018*"innocent" + 0.015*"occupy" + 0.013*"continue" + '
  '0.011*"give" + 0.011*"death"'),
 (2,
  '0.031*"day" + 0.016*"live" + 0.014*"see" + 0.014*"life" + 0.012*"also" + '
  '0.011*"occupation" + 0.011*"child" + 0.010*"month" + 0.010*"today" + '
  '0.010*"pellet"')]
length of doc_lda: 2278
```

In [182]:
```
# 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:  -7.595207532654055

Coherence Score:  0.4676174627848224
```

In [183]:
```
pyLDAvis.enable_notebook()
vis = pyLDAvis.gensim.prepare(lda_model, corpus, id2word)
vis
```

Out[183]:

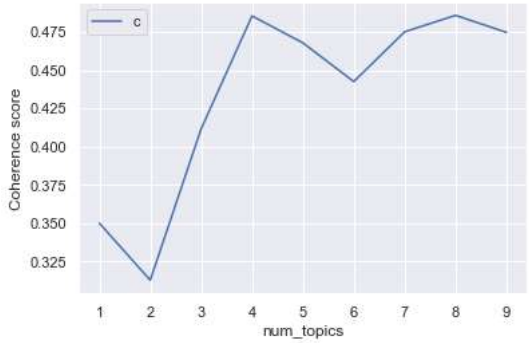


Finding out the optimal number of topics

Topic coherence in essence measures the human interpretability of a topic model. Traditionally perplexity has been used to evaluate topic models however this does not correlat
Topic coherence is another way to evaluate topic models with a much higher guarantee on human interpretability. Thus this can be used to compare different topic models amon

In [184]:
```
def evaluate_graph(dictionary, corpus, texts, limit):
    """
    Function to display num_topics - LDA graph using c_v coherence

    Parameters:
    ----------
    dictionary : Gensim dictionary
    corpus : Gensim corpus
    limit : topic limit

    Returns:
    -------
    lm_list : List of LDA topic models
    c_v : Coherence values corresponding to the LDA model with respective number of topics
    """
    c_v = []
    lm_list = []
    for num_topics in range(1, limit):
        lm = gensim.models.ldamodel.LdaModel(corpus=corpus, num_topics=num_topics, id2word=dictionary)
        lm_list.append(lm)
        cm = CoherenceModel(model=lm, texts=texts, dictionary=dictionary, coherence='c_v')
        c_v.append(cm.get_coherence())

    # Show graph
    x = range(1, limit)
    plt.plot(x, c_v)
    plt.xlabel("num_topics")
    plt.ylabel("Coherence score")
    plt.legend(("c_v"), loc='best')
    plt.show()

    return lm_list, c_v
```

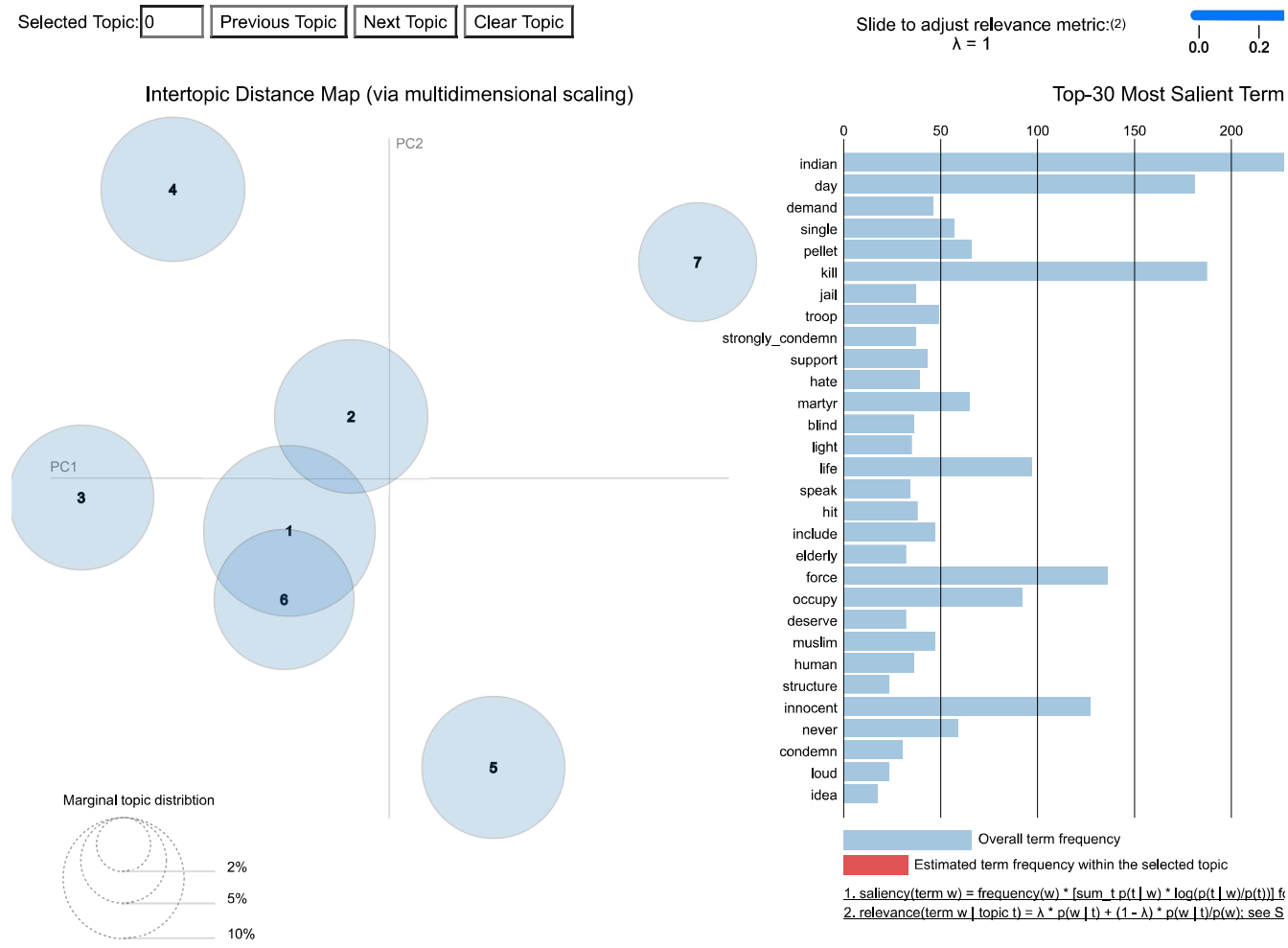
In [185]:
```
%%time
lmlist, c_v = evaluate_graph(dictionary=id2word, corpus=corpus, texts=data_lemmatized, limit=10)
```




```
Wall time: 1min 15s
```

In [186]:
```
pyLDAvis.gensim.prepare(lmlist[6], corpus, id2word)
```

Out[186]:



Hierarchical Dirichlet Process (HPD)

Topic models such as LDA and LSI helps in summarising and organising large archives of texts that is not possible to analyse by hand. Apart from LDA and LSI, one other powe (Hierarchical Dirichlet Process). It's basically a mixed-membership model for unsupervised analysis of grouped data. Unlike LDA (its's finite counterpart), HDP infers the number

In [187]:
```
# Hierarchical Dirichlet Process (HDP) topic model with regards to Gensim
from gensim.models.hdpmodel import HdpModel
#hdp = HdpModel(corpus=corpus,id2word=id2word)
hdpmodel = HdpModel(corpus=corpus, id2word=id2word)
```

Viewing Topics

The HDP model (Hdp_model) can be used to view the topics from the documents. It can be done with the help of following script

In [188]:
```
pprint(hdpmodel.show_topics())
```

```
[(0,
  '0.003*super + 0.003*prophet + 0.002*behaviour + 0.002*house + 0.002*battle '
  '+ 0.002*inseparable + 0.002*chooser + 0.002*anticipate + 0.002*page + '
  '0.002*exchange + 0.002*whisper + 0.002*suspect + 0.002*talk + 0.002*pow + '
  '0.002*normalcy + 0.002*intolerable + 0.002*white + 0.002*jawan + '
  '0.002*alive + 0.002*ear'),
 (1,
  '0.003*demonstrate + 0.003*zone + 0.003*surface + 0.003*family + 0.003*word '
  '+ 0.003*eligible + 0.002*send + 0.002*dispossessed + 0.002*utterly + '
  '0.002*quiet + 0.002*address + 0.002*much + 0.002*god + 0.002*object + '
  '0.002*span + 0.002*campaign + 0.002*possession + 0.002*connected + '
  '0.002*matyrdom + 0.002*entry'),
 (2,
  '0.003*bhutto + 0.002*complicit + 0.002*ask + 0.002*grace + 0.002*meanwhile '
  '+ 0.002*shoot + 0.002*proof + 0.002*screen + 0.002*commander_junaid + '
  '0.002*compliant + 0.002*silence + 0.002*protest + 0.002*government + '
  '0.002*relative + 0.002*takfiri + 0.002*yardstick + 0.002*extremist + '
  '0.002*inseparable + 0.002*humanity + 0.002*humiliate'),
 (3,
  '0.003*minister + 0.003*college + 0.003*reportedly + 0.003*becau + '
  '0.002*indifference + 0.002*check + 0.002*migrant + 0.002*arson + '
  '0.002*phone + 0.002*takfiri + 0.002*decade + 0.002*online + 0.002*illegal + '
  '0.002*bhat + 0.002*integral + 0.002*strange + 0.002*truth + 0.002*breath + '
  '0.002*bed + 0.002*prize'),
 (4,
  '0.003*skip + 0.003*socalled + 0.003*lord + 0.003*communit + 0.003*job + '
  '0.002*punishment + 0.002*adjoin + 0.002*stay + 0.002*toy + 0.002*wipe + '
  '0.002*legendary + 0.002*home + 0.002*hang + 0.002*butcher + 0.002*commit + '
  '0.002*indifference + 0.002*superpower + 0.002*nawakadl + 0.002*succeed + '
  '0.002*experience'),
 (5,
  '0.003*sonmarg + 0.003*curfew + 0.003*unaffected + 0.002*effort + 0.002*beef '
  '+ 0.002*illustrate + 0.002*tension + 0.002*policy + 0.002*dispossessed + '
  '0.002*injured + 0.002*letter + 0.002*ago + 0.002*violent + 0.002*lawyer + '
  '0.002*soldier + 0.002*regime + 0.002*criminal + 0.002*characteristic + '
  '0.002*bre + 0.002*work'),
 (6,
  '0.003*mercy + 0.003*security_official + 0.003*sane + 0.002*raid + '
  '0.002*revocation + 0.002*buy + 0.002*use + 0.002*leisurely + 0.002*speech + '
  '0.002*care + 0.002*related + 0.002*policy + 0.002*asleep + 0.002*minority + '
  '0.002*inshaallah + 0.002*tyrant + 0.002*rss_bjp + 0.002*covid + 0.002*early '
  '+ 0.002*jiofibre'),
 (7,
  '0.004*determine + 0.003*expell + 0.003*positive + 0.003*lung + '
  '0.002*happiness + 0.002*liberty + 0.002*truth + 0.002*sight + '
  '0.002*indulgence + 0.002*stabbi + 0.002*design + 0.002*racism + '
  '0.002*intonate + 0.002*anyt + 0.002*motto + 0.002*wipe + 0.002*dark + '
  '0.002*slumber + 0.002*leadership + 0.002*arthritis'),
 (8,
  '0.003*buy + 0.002*earth + 0.002*raid + 0.002*slight + 0.002*affect + '
  '0.002*demography + 0.002*asking + 0.002*emergency + 0.002*socalled + '
  '0.002*build + 0.002*apart + 0.002*arrogance + 0.002*commit + 0.002*future + '
  '0.002*surf + 0.002*unheard + 0.002*maqbool + 0.002*inch + 0.002*indial + '
  '0.002*arthritis'),
 (9,
  '0.004*discriminate + 0.003*love + 0.003*basim_aijaz + 0.003*conference + '
  '0.002*steroid + 0.002*s + 0.002*infect + 0.002*illegal + 0.002*causality + '
  '0.002*earth + 0.002*fearless + 0.002*fighter + 0.002*loud + 0.002*lung + '
  '0.002*trace + 0.002*see + 0.002*able + 0.002*disrespect + 0.002*ball + '
  '0.002*value'),
 (10,
  '0.003*painful + 0.003*invoke + 0.003*roof + 0.002*opinion + 0.002*joke + '
  '0.002*service + 0.002*bad + 0.002*political + 0.002*steal + '
  '0.002*mainstream + 0.002*humanity + 0.002*bright + 0.002*super + 0.002*deal '
  '+ 0.002*teller + 0.002*incompetency + 0.002*gas + 0.002*die + 0.002*blame + '
  '0.002*terrorism'),
 (11,
  '0.003*also + 0.002*know + 0.002*spo + 0.002*mobile + 0.002*loose + '
  '0.002*medium + 0.002*supremacist + 0.002*vital + 0.002*photo + '
  '0.002*bulletproof + 0.002*guide + 0.002*shutter + 0.002*therefore + '
  '0.002*depend + 0.002*hang + 0.002*give + 0.002*minister + 0.002*disrespect '
  '+ 0.002*gutter + 0.002*superpower'),
 (12,
  '0.003*outside + 0.003*tum + 0.002*brother + 0.002*minister + 0.002*bare + '
  '0.002*unfortunately + 0.002*tactic + 0.002*atleast + 0.002*caso + '
  '0.002*murder + 0.002*building + 0.002*believe + 0.002*access + '
  '0.002*resilient + 0.002*hum + 0.002*emotionally + 0.002*skin + '
  '0.002*replace + 0.002*sol + 0.002*hoist'),
 (13,
  '0.003*cue + 0.003*supplication + 0.003*narrative + 0.003*exchange + '
  '0.002*negligence + 0.002*fierce + 0.002*talk + 0.002*decease + 0.002*allahs '
  '+ 0.002*failure + 0.002*dead + 0.002*prangnat + 0.002*experiment + '
  '0.002*scare + 0.002*necessary + 0.002*entity + 0.002*guarantee + 0.002*blee '
  '+ 0.002*national_security + 0.002*acceptance'),
 (14,
  '0.003*result + 0.003*surprised + 0.003*ignore + 0.002*wither + 0.002*quran '
```

```
 '+ 0.002*territory + 0.002*work + 0.002*american + 0.002*godness + '
 '0.002*commitment + 0.002*security_official + 0.002*copy + 0.002*painful + '
 '0.002*good + 0.002*resource + 0.002*explode + 0.002*atrocity + '
 '0.002*jealousy + 0.002*respond + 0.002*save'),
(15,
 '0.003*arm + 0.002*initiate + 0.002*smhs + 0.002*suppose + 0.002*aim + '
 '0.002*immediately + 0.002*sound + 0.002*scar + 0.002*dance + 0.002*suffer + '
 '0.002*lest + 0.002*prepare + 0.002*wise + 0.002*great + 0.002*relief + '
 '0.002*silence + 0.002*arrogance + 0.002*amit + 0.002*stat + 0.002*concept'),
(16,
 '0.003*fight + 0.003*forget + 0.002*destruction + 0.002*suit + 0.002*way + '
 '0.002*send + 0.002*stabbi + 0.002*promote + 0.002*hurriyat_leader + '
 '0.002*bed + 0.002*bhat + 0.002*deployment + 0.002*worry + 0.002*rage + '
 '0.002*friend + 0.002*innumerable + 0.002*sanitize + 0.002*conflate + '
 '0.002*province + 0.002*stop'),
(17,
 '0.003*parent + 0.003*right + 0.003*ideology + 0.003*write + 0.003*produce + '
 '0.002*attend + 0.002*suit + 0.002*vulnerable + 0.002*inve + '
 '0.002*machiavellian + 0.002*tweeting + 0.002*tribal + 0.002*employee + '
 '0.002*gas + 0.002*magic + 0.002*sorry + 0.002*quit + 0.002*god + '
 '0.002*careful + 0.002*hour'),
(18,
 '0.003*maintain + 0.003*constructive + 0.003*enjoy + 0.002*state_terrorism + '
 '0.002*regret + 0.002*present + 0.002*restriction + 0.002*freedom_fighter + '
 '0.002*encounter + 0.002*injust + 0.002*moral + 0.002*supplication + '
 '0.002*judgement + 0.002*relative + 0.002*sever + 0.002*detrimental + '
 '0.002*playbook + 0.002*convent + 0.002*mass + 0.002*join'),
(19,
 '0.003*racist + 0.003*mourner + 0.003*hatred + 0.003*fund + 0.002*failure + '
 '0.002*describe + 0.002*deteriorate + 0.002*freedome + 0.002*believe + '
 '0.002*co + 0.002*punishment + 0.002*session + 0.002*jawan + 0.002*colony + '
 '0.002*routinely + 0.002*baby + 0.002*afghan + 0.002*paki + 0.002*let + '
 '0.002*hold_accountable')]
```