Topic modelling using Latent Dirichlet Allocation (LDA)

Algorithm and design decision:

Data is extracted from the Galway Bay FM twitter account. GetOldTweets3 python library is used to get tweets. These tweets are saved in a data frame and pre-processed by removing English stop words along with other words which is very frequent in Galway Bay FM tweets and all the words are converted into lower case. A matrix of token counts is created from the pre-processed list of words using sci-kit learn's countVectorizer function. This matrix of the token count is used as an input data for Latent Dirichlet Allocation algorithm. Latent Dirichlet Allocation algorithm is used from Scikit learn's decomposition class.

Latent Dirichlet Allocation (LDA) is used for topic modelling. It is the process of identifying the abstract topics is the collection of the documents. LDA is an unsupervised learning algorithm that considers a set of documents as a bag of words. This algorithm gives Dirichlet distributions, a topic per document model and words per topic model.

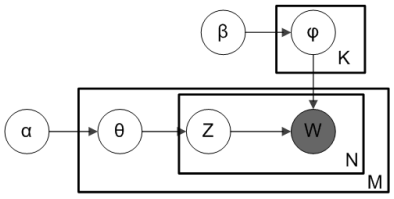

Figure 1: Plate diagram of LDA [1]

Figure 1 represents the plate diagram of the LDA model, where

M represents the number of documents and N presents the number of words in a document.

Φ represents the word distribution for topic k and θ represents the topic distribution for document m.

α is the parameter of the Dirichlet prior on the per-document topic distributions.

β is the parameter of the Dirichlet prior on the per-topic word distribution.

M is the specific word and Z is a topic for N^{th} word in document m.

Process:

- Assign a topic to words across document *m* by covering all the *k* topic.
- For each word *w* in a document *m*, it is assumed that all the other words are assigned with the correct topic except that word *m.*
- Assignment of word *w* to a topic is based on two things, one is based on what all topics are present in the document *m*, and across all of the documents how many times this word *w* is assigned to a particular topic.
- Repeat these processes to build the distributions.

In symmetric distribution, if α is high, then documents contain more topics and If β is high, topics are made up of most of the words.

LatentDirichletAllocation algorithm is used from the sci-kit learn. Set parameters such as *maximum number of iterations* to 50 and the *number of topics* to 20 in LDA.

LDA provides topics which belong to different category. Among all these 20 topics, few and relevant topics are selected by manual inspection. Policy or topic keywords (such as funding, hospital, traffic

etc) are manually determined by referring public data source. These keywords are used to match and filter clustered topics. Six number of topics are selected after filtering all the clustered topics.

LDA returns the topic per document probability distribution, this probability distribution is used to remove the documents having the same probability for all the topics. If the tweet has the same probability for all the topics then the tweet is removed. By removing these tweets, we will have only relevant documents/tweets to analyse. The number of tweets per topic is calculated by adding each tweet to a topic. A topic with high probability value in a tweet is considered as a topic of that particular tweet. Topic name and number of tweets per topic is saved in a table and saved to CSV file.

Visualisation of the result:

Matplotlib's bar chart is used to visualise the counts of the tweets per topic as shown in figure 2. Okabe and ito's colour palette is used to visualise the bar's colour. Okabe and ito's palette colour are suitable for both standard and colour vision deficiency people.

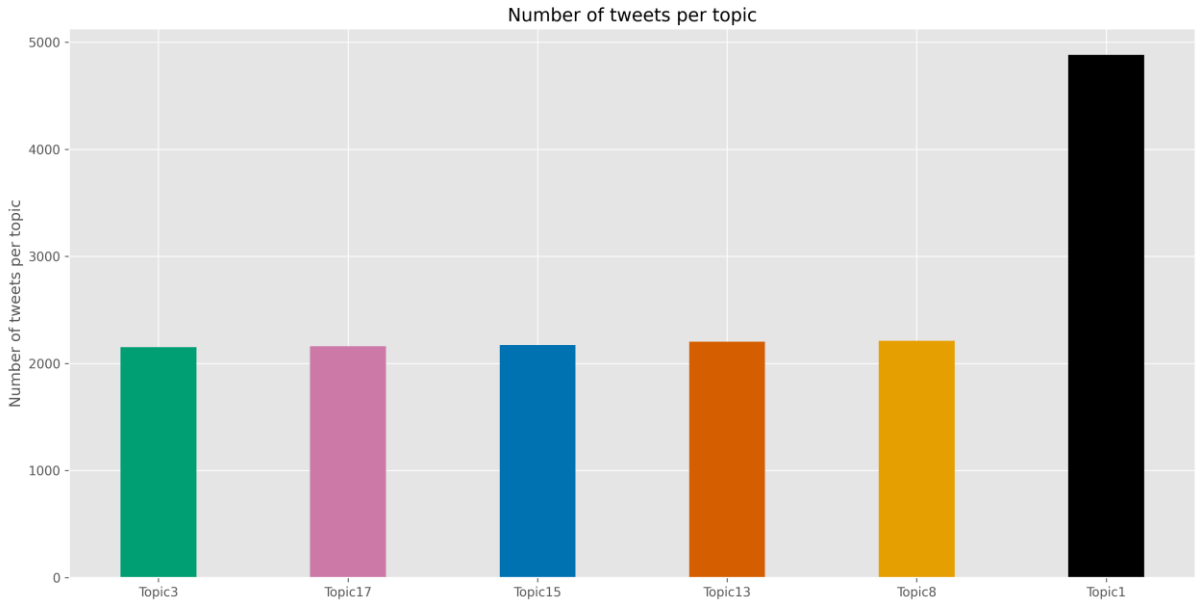

Figure 2

By manually inspecting, 6 number of topics are selected. *Topic3* represents health policy, keywords like hospital, million, and support has weightage in this topic. *Topic17* represents social/estate policy, keywords like housing and high has high weightage in this topic. *Topic15* represents transport policy keywords like traffic and funding has high weightage in this topic. *Topic13* represents a topic related to the general election and Brexit. *Topic8* represents a topic related to coronavirus confirmed cases. *Topic1* represent topic related to climate change, keywords like climate, water, and the day has high weightage.

The output of the LDA algorithm can be visualised using pyLDAvis package. pyLDAvis helps to interpret the topics in a topic model as shown in figure 3. The labelled topic number shown in the below figure is different because this package executes separately by using LDA model and token counts independent of the previously shown model. pyLDAvis provide interactive visualisation.

In the plot created by using pyLDAvis is shown in figure 3, topic 15 in this figure represents the topic related to health policy, which is *topic3* in figure 2. The horizontal bar chart shows how much each term contributed to a specific topic. Terms like hospital, support, million have a major contribution to the topic 15. The interactive visualisation HTML file is provided in the assignment zip folder.

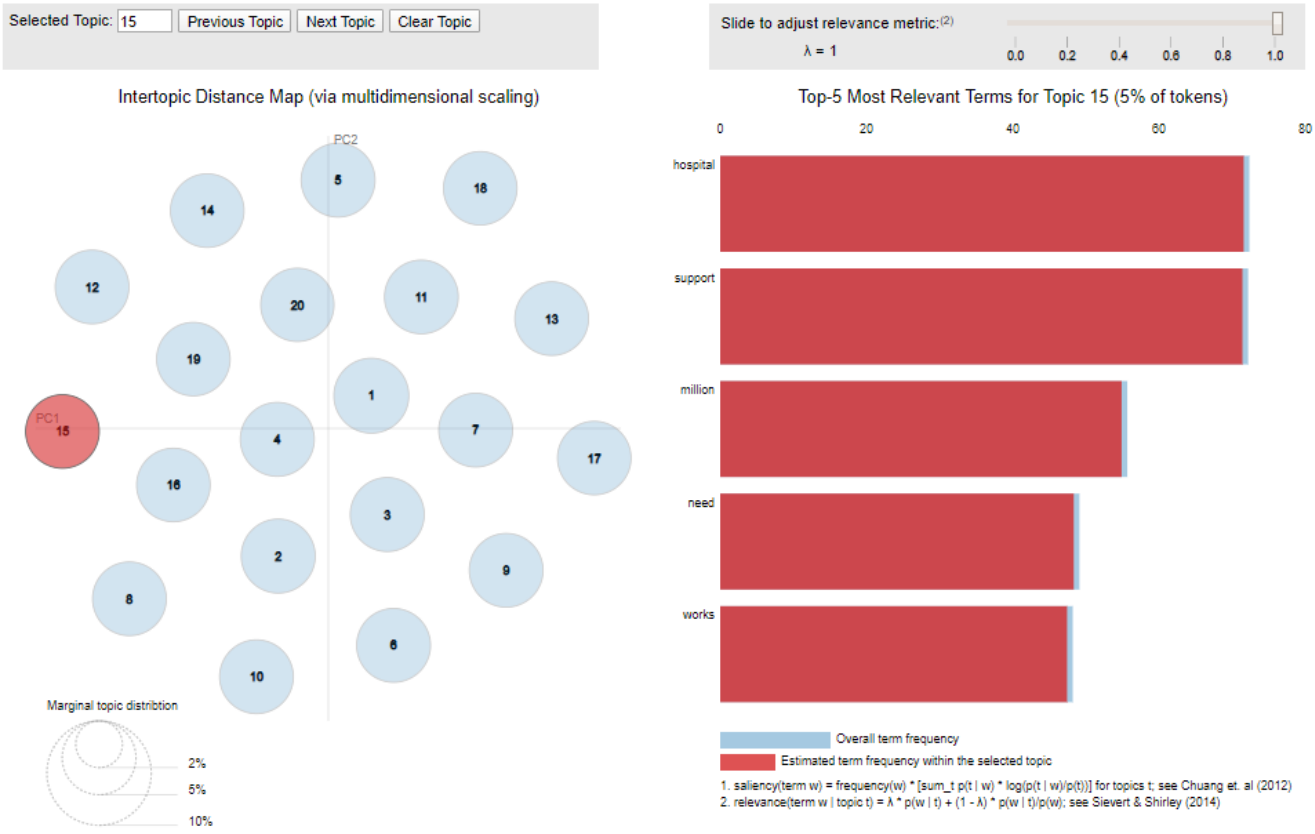

Figure 3

Conclusion and observation:

From figure 2, it is observed that *topic1* which is related to weather is popular in Galway which is around 5000 tweets. All the remaining topics are in the same range which is around 2200 number of tweets per topic. *Topic3* related to health policy has the lowest number of tweets among other topics. It is also observed that Galway Bay FM tweets don't contain much about government policies rather tweets are about daily news such as weather warning, coronavirus outbreak. Topics like Brexit and the general election is also popular in Galway. Policies related to health and transportation is quite treading in Galway.

References:

[1]"Latent Dirichlet allocation", *En.wikipedia.org*, 2020. [Online]. Available: https://en.wikipedia.org/wiki/Latent_Dirichlet_allocation. [Accessed: 10- Apr- 2020].

Appendix:

```
# install package
pip install GetOldTweets3
# install package
pip install pyLDAvis
# part 1 - importing tweeter data
# import
import GetOldTweets3 as getOTw3
import pandas as pd
# import tweets from the galwaybayfmnews using GetOldTweets3
galwaybayfmTweetObj = getOTw3.manager.TweetCriteria().setQuerySearch('Galwa
ybayfmnews')\
                                            .setSince("2019-08-7")\
                                            .setUntil("2020-04-7")


# get tweets
galwayBayFmTweets = getOTw3.manager.TweetManager.getTweets(galwaybayfmTweet
Obj)


# preprocessing
# create dataframe from imported tweets
galwayBay_df = pd.DataFrame()


# store tweet into data frame
for tweet in galwayBayFmTweets:
   galwayBay_df = galwayBay_df.append(pd.Series(tweet.text), ignore_index=T
rue)
   print(tweet.text)


import re
# remove all the punctuation in the tweets
galwayBay_df['process'] = galwayBay_df[0].map(lambda x: re.sub('[,\\.!?]',
'', x))
# split string into array by space
galwayBay_df['process'] = galwayBay_df['process'].str.lower().str.split()


# build list of words from data frame
words = []
```

```
for word in galwayBay_df.loc[:,'process']:
  for sword in word:
    words.append(sword)


from sklearn.feature_extraction.stop_words import ENGLISH_STOP_WORDS


# define prefix words to remove from list of words
prefixes = ('http', 'https', 'news', 'galway','uhg','oughterard','new','say
s','it', 'galwayad','know','connemara', 'tuam' ,'we' ,'one', 'us','say','go
','thanks','td','well','next','back','loughrea', 'see', 're', '&amp','may',
've','like','johnson', 'three' ,'two','near','that','na','ugh','galwayad','
boris','galwaycoco','fyi','12','please','help')
# define suffix words to remove from list of words
suffix = ('galway', 'galwaybayfmnews', 'news',  'galwayad', '&amp')
# remove prefix and suffix
words = [x for x in words if not x.startswith(prefixes)]
words = [x for x in words if not x.endswith(suffix)]


# remove stopwords
words = [item for item in words if item not in ENGLISH_STOP_WORDS]


# to get matrix of token counts
from sklearn.feature_extraction.text import CountVectorizer
# part 2 - Topic modelling
# create tokenizer
countVectObj = CountVectorizer()


# get matrix of token counts of imported tweets
tokenData = countVectObj.fit_transform(words)
# Model building
# Import LDA model from scikit learn
from sklearn.decomposition import LatentDirichletAllocation


# total number of topics
num_of_topic = 20


# create lda model with 50 iterations and learning method is online
lda_model = LatentDirichletAllocation(n_components=num_of_topic, max_iter=5
0, learning_method='online')
```

```
lda_topics = lda_model.fit_transform(tokenData)


# visualise the topics in the cluster
def Visulise_topics(lda, features, numOfword):
    for top_idx, topic in enumerate(lda.components_):
        print("Topic %d:" % (top_idx))
        print(" ".join([features[i] for i in topic.argsort()[:-numOfword - 1:-1]]))


# get feature names from tokenizer
features = countVectObj.get_feature_names()
# visulise the topics
Visulise_topics(lda_model, features, 5)


# remove the irrelevant posts from the model's output
topicsPerDocu = pd.DataFrame(lda_topics, columns=["Topic"+str(i+1) for i in range(num_of_topic)])


# part 3 - tweets per topic count
# to compare all the topic probablity is equal in a document
val = 1/num_of_topic
# remove rows having all the topic probablities are equal
topicsPerDocu = topicsPerDocu.loc[(topicsPerDocu != val).all(axis=1),]


# select a topic with high probability as a topic of the perticular documen t
highProbTopic = topicsPerDocu.idxmax(axis=1)
# count number of documents per topic
docCountPerTopic = highProbTopic.groupby(highProbTopic).count()


# convert series into dataframe
docCountPerTopic_df = pd.DataFrame(docCountPerTopic)
docCountPerTopic_df.reset_index( inplace=True)
docCountPerTopic_df.columns = ["Topic Name", 'Number of Tweets']


# sort the dataframe in ascending order of the number of tweets
docCountPerTopic_df.sort_values(by=['Number of Tweets'], inplace=True)
```

```
# By manual selecting topics topics
docCountPerTopic_df = docCountPerTopic_df.loc[docCountPerTopic_df['Topic Na
me'].isin(['Topic1','Topic3','Topic8','Topic13','Topic15','Topic17'])]


docCountPerTopic_df


# save table to csv
docCountPerTopic_df.to_csv("tweetsPerTopic.csv", index=False)
# part 4 - Visualise the tweets per topic
# visualise document per topics
# bar plot
import matplotlib.pyplot as plot
plot.style.use('ggplot')


# define figure object
figure = plot.figure()
# define axes object
axes = figure.add_axes([0,0,1,1])


# visulise bar chart
plot.bar(docCountPerTopic_df['Topic Name'], docCountPerTopic_df['Number of
Tweets'], width=0.4, color=['#009E73', "#CC79A7", "#E69F00", "#000000"])


# set title and ylabel
plot.ylabel('Number of tweets per topic')


# to visualise the LDA topic model using pyLDAvis
import pyLDAvis.sklearn
pyLDAvis.enable_notebook()
panel = pyLDAvis.sklearn.prepare(lda_model, tokenData, countVectObj, mds='t
sne', R=5)
panel
```