Classification of Suicide and Depression using NLP

A Report submitted to the Rajiv Gandhi University of Knowledge Technologies in partial fulfilment of the degree of

Bachelor of Technology

In

Computer Science and Engineering

By

Katyayani Atmakuri

S160022

Prathyusha Rejeti

S160577

Harshitha Bandaru

S160947

Ramya Bhanu Sri Varsha Chakranthi

S160407

SRIKAKULAM - 532402

Andhra Pradesh, India

CERTIFICATE

This is to certify that the report entitled "Classification of Suicide and Depression using NLP" submitted by **Katyayani Atmakuri,** bearing ID. No. S160022, **Harshitha Bandaru,** bearing ID. No. S160947, **Ramya Bhanu Sri Varsha Chakranthi**, bearing ID. No. S160407 and **Prathyusha Rejeti**, bearing ID. No. S160577 in partial fulfilment of the requirements for the award of Bachelor of Technology in Computer Science and Engineering is a bona fide work carried out by them under my supervision and guidance.

The report has not been submitted previously in part or in full to this or any other University or Institution for the award of any degree or diploma.

Ms. Vishnu Priyanka Javvadi,

Project Internal Guide,

Asst. Professor(C), Dept of CSE,

RGUKT, SRIKAKULAM.

Ms. M. Roopa,

Head of the Department,

Department of CSE,

RGUKT, SRIKAKULAM.

DECLARATION

I Katyayani Atmakuri, Harshitha Bandaru, Prathyusha Rejeti and Ramya Bhanu Sri Varsha Chakrathi hereby declare that this report entitled "**Classification of Suicide and Depression using NLP"** submitted by me under the guidance and supervision of **Ms. Vishnu Priyanka mam** is a bona fide work. I also declare that it has not beensubmitted previously in part or in full to this University or other University or Institution for the award of any degree or diploma.

Date: 02-05-2022

Place: Nuzvid.

(Katyayani Atmakuri)

ID: S160022

(Prathyusha Rejeti)

ID: S160577

(Harshitha Bandaru)

ID: S160947

(Ramya Bhanu Sri Varsha Chakranthi)

ID: S160407

Acknowledgments

I would like to express my sincere gratitude to **Javvadi Vishnu Priyanka Mam**, my project Guide, for valuable suggestions and keen interest throughout the progress of my course of research.

I am grateful to **Musidi Roopa Mam**, HOD CSE, for providing excellent computing facilities and a congenial atmosphere for progressing with my project.

At the outset, I would like to thank **Rajiv Gandhi University of Knowledge Technologies, Srikakulam** for providing all the necessary resources for the successful completion of my course work. At last, but not the least I thank my classmates and other students for their physical and moral support.

With Sincere Regards,

Katyayani Atmakuri,

Prathyusha Rejeti,

Harshitha Bandaru,

Ramya Bhanu Sri Varsha Chakranthi.

Abstract

In today's world people are sharing their opinions, feelings in online platforms with strangers as they won't be judged by doing this. So, many well-being applications are being developed and communities in social media applications are getting more, like reddit.

Early detection of suicidal ideation in depressed individuals can allow for adequate medical attention and support, which in many cases is life-saving. There is little research into finding the line where depression turns into suicidal ideation, which is a more difficult clinical and technological task. We use the data related to suicide and depression posts from applications like reddit which are posted by people with depressed thoughts. We classify the posts which are suicidal and depressed. We use lemmatization, sentiment analysis and topic modelling on the collected dataset and apply logistic regression to classify the posts according to the text analysis method outputs.

Keywords: Suicide, depression, reddit, Natural Language Processing, Classification, Topic Modelling, Cosine Similarity, TF-IDF Vectorization, MDS Visualization.

List of Figures

4.3 Classification ……………………………………………………………………………

4.3.1 Confusion matrix without Normalization ………………………………………

4.3.2 Confusion matrix with Normalization ……………………………………………

4.3.3 Unigrams and Bigrams …………………………………………………………

4.3.4 Top five unigrams in suicide …………………………………………………..

4.4 Topic Modelling …………………………………………………………………….

4.4.1 Latent Dirichlet Allocation ……………………………………………………….

5 Result

5.3.1 Topic Modelling Visualization ………………………………………………..

Contents

Title ……………………………………………………………………………………………………

Certificate ……………………………………………………………………………………….... i

Declaration ……………………………………………………………………………………... ii

Acknowledgments …………………………………………………………………………….... iii

Abstract …………………………………………………………………………………………… iv

List of Figures …………………………………………………………………….………. v

1. Introduction…………………………………………………………………...

1.1 Introduction …………………………………………………………………. 1

1.2 Applications ………………………………………………………………… 1

1.3 Problem Statement ………………………………………………………….. 1

1.4 Organization of Report ……………………………………………………… 1

2. Literature Survey……………………………………………………………..

2.1 Web Scraping ……………………………………………………………….. 2

2.2 Sentiment Analysis ……………………………………………………….. 2

2.3 TF-IDF Vectorization ……………………………………………………… 3

2.4 Cosine Similarity …………………………………………………………... 4

2.5 MDS Visualization ………………………………………………………… 4

3. Topic Modelling……………………………………………………………

3.1 Latent Dirichlet Allocation ……………………………………….……… 5

4. Implementation and Design……………………………………………….

4.1 Web Scraping ……………………………………………………………… 7

4.2 Sentiment Analysis…………………………………………………….… 8

4.3 Classification ……………………………………………………………… 10

4.4 Topic Modelling…………………………………………………………..… 11

5. Result……………………………………………………………………

5.1 Topic Modelling Visualization ………………………………………………. 13

6. Conclusion and Future Enhancement……………………………………

6.1 Conclusion……………………..…………………………………………… 15

6.2 Future Enhancement……………………………………………………..…. 15

Bibliography…………………………………………………………………...

Chapter 1

Introduction

1.1 Introduction

Teenagers and young users more often post about mental health on social media. Depression is the most pressing issue worldwide. People who are suffering from depression tend to have suicidal thoughts, if they are left unaddressed, there is a chance that they may commit suicide.

Reddit is a social media platform where people can post about their feelings anonymously. We are considering two subreddits, namely, **r/depression** and **r/suicidewatch** to scrap the data. They are considered benchmark spaces where people frequently post what they are feeling and they are supportive spaces in which people encourage and push each other. We use the scraped dataset to categorize depressed and suicidal thoughts.

1.2 Applications

- Suicide and Depression Detection.
- Classification from reddit posts.

1.3 Problem Statement

To analyse and build a machine learning model, that categorizes the reddit posts into suicide and depression using Natural Language Processing (NLP) which helps in aiding the people suffering with depression and suicidal thoughts.

1.4 Organization of Report

The rest of this thesis is organized as follows: Chapter 2 gives literature survey about sentiment analysis, natural language processing. Chapter 3 is about Topic Modelling. Chapter 4 deals about procedure, implementation and design. The Result of the work is given in Chapter 5 and Conclusion and Future Scope in Chapter 6.

Chapter 2

Literature Survey

2.1 Web Scrapping

Web scraping is an automatic method to obtain large amounts of data from websites. Most of this data is unstructured data in raw format which is then converted into structured data in a spreadsheet or a database so that it can be used in various applications.

There are many different ways to perform web scraping to obtain data from websites. These include using online services, particular API's or even creating your code for web scraping from scratch.

Many large websites, like Reddit, Google, Twitter, Facebook, StackOverflow, etc. have API's that allow you to access their data in a structured format. This is the best option, but there are other sites that don't allow users to access large amounts of data in a structured form or they are simply not that technologically advanced.

In that situation, it's best to use Web Scraping to scrap the website for data.

Data is scrapped from reddit and python's praw (Python Reddit API wrapper) module is used to scrap the data.

2.2 Sentiment Analysis

The process of identifying and categorizing the opinion of a piece of text and determine the context of text, whether it is positive, negative, neutral or compound.

It is an approach to natural language processing that identifies the emotional tone behind a body of text.

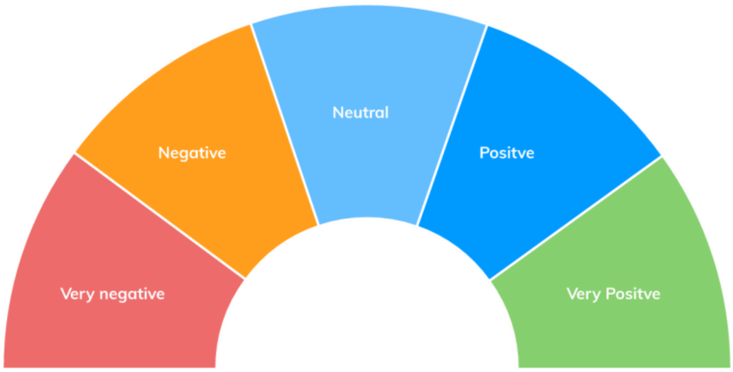
Sentimental Analysis is also known as ***opinion mining or emotional artificial intelligence***.

Sentimental Scoring

A key aspect of sentiment analysis is polarity classification. Polarity refers to the overall sentiment conveyed by a particular text, phrase or word. This polarity can be expressed as a numerical rating known as a "sentiment score". For example, this score can be a number between -100 and 100 with 0 representing neutral sentiment. This score could be calculated for an entire text or just for an individual phrase.

Fine-grained Sentiment Analysis

Sentiment scoring can be as fine-grained as required for a specific use case. Categories can expand beyond just "positive", "neutral" and "negative". For example, you may choose to use five categories.



Steps of sentiment Analysis:

1. Tokenization: Divides statements into set of words.
2. Cleaning: Removes the special characters.
3. Stop word removal: Removes the stop words.
4. Classification: Classifies whether word is positive (+1), negative (-1) or neutral (0) and apply the classification algorithms after training the model.
5. Calculation: Polarity will be calculated.

2.3 TF-IDF Vectorization

- TF-IDF means Term Frequency - Inverse Document Frequency
- It gives a measure that takes the importance of word into consideration depending on how frequently it occurs in a document or a corpus
- TF means we find frequency of a word in the document

$$\text{Term Frequency} \Rightarrow \frac{\text{Number of repetition of word in a sentence}}{\text{Total Number of words in sentence}}$$

- IDF measures the importance of the word in corpus

$$\text{Inverse Document Frequency} \Rightarrow \log\left(\frac{\text{Number of Sentences}}{\text{Number of Sentences Contaning words}}\right)$$

2.4 Cosine Similarity

- It is used to measure the similarity in the text analysis.
- Cosine similarity is represented by cos theta.
- Theta is the angle between points.
- Always ranging between -1 to +1.
- For example, cos 45 =0.53 that means similarity is around 53%.
- cos 90 =0 that means points are not similar.
- If $\theta = 0°$, the ‘x’ and ‘y’ vectors overlap, thus proving they are similar.
- If $\theta = 90°$, the ‘x’ and ‘y’ vectors are dissimilar.
- similarity measure refers to distance with dimensions representing features of the data object, in a dataset. If this distance is less, there will be a high degree of similarity, but when the distance is large, there will be a low degree of similarity.
- In cosine similarity, data objects in a dataset are treated as a vector. The formula to find the cosine similarity between two vectors is –
  Cos (x, y) = x . y / ||x|| * ||y||

  where,

  - x . y = product (dot) of the vectors ‘x’ and ‘y’.
  - ||x|| and ||y|| = length of the two vectors ‘x’ and ‘y’.
  - ||x|| * ||y|| = cross product of the two vectors ‘x’ and ‘y’.
- The cosine similarity is beneficial because even if the two similar data objects are far apart by the Euclidean distance because of the size, they could still have a smaller angle between them. Smaller the angle, higher the similarity.
- When plotted on a multi-dimensional space, the cosine similarity captures the orientation (the angle) of the data objects and not the magnitude.

2.5 MDS Visualization

- MDS (Multidimensional Scaling) is an algorithm that transforms a dataset into another dataset, usually with lower dimensions, keeping the same Euclidean distances between the points.
- MDS, also known as Principal Coordinates Analysis (PCoA), is a statistical technique originating in psychometrics.
- The data used for multidimensional scaling (MDS) are dissimilarities between pair of objects.
- The main objective of MDS is to represent these dissimilarities as distances between points in a lower dimensional space such that the distances correspond as closely as possible to the dissimilarities.

Chapter 3

Topic Modelling

Topic modelling is recognizing the words from the topics present in the document or the corpus of data.
This is useful because extracting the words from a document takes more time and is much more complex than extracting them from topics present in the document.
For example, there are 1000 documents and 500 words in each document. So, to process this it requires 500*1000 = 500000 threads. So, when you divide the document containing certain topics then if there are 5 topics present in it, the processing is just 5*500 words = 2500 threads.

Some of the important points or topics which makes text processing easier in NLP:

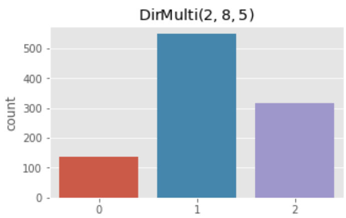
- Removing stop words and punctuation marks
- Stemming
- Lemmatization
- Encoding them to ML language using Count vectorizer or TF-IDF vectorizer

Topic modelling is done using LDA (Latent Dirichlet Allocation).

3.1 Latent Dirichlet Allocation (LDA):

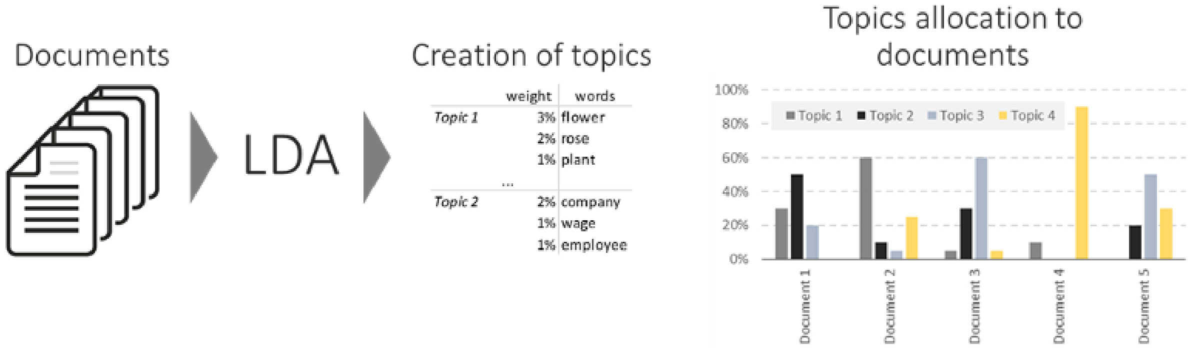
In LDA, latent indicates the hidden topics present in the data then Dirichlet is a form of distribution. Dirichlet distribution is different from the normal distribution. When ML algorithms are to be applied the data has to be normally distributed or follows Gaussian distribution. The normal distribution represents the data in real numbers format whereas Dirichlet distribution represents the data such that the plotted data sums up to 1. It can also be said as Dirichlet distribution is a probability distribution that is sampling over a probability simplex instead of sampling from the space of real numbers as in Normal distribution.

For example,



Normal distribution tells us how the data deviates towards the mean and will differ according to the variance present in the data. When the variance is high then the values in the data would be both smaller and larger than the mean and can form skewed distributions. If the variance is small then samples will be close to the mean and if the variance is zero it would be exactly at the mean.

Topic modelling refers to the task of identifying topics that best describes a set of documents. These topics will only emerge during the topic modelling process (therefore called latent). And one popular topic modelling technique is known as Latent Dirichlet Allocation (LDA).



Chapter 4

Implementation and Design

4.1 Web Scrapping

```python
import pandas as pd
import numpy as np
import praw
from praw.models import MoreComments
import datetime



def pull_Reddit_Posts(subreddit, num_posts):

    reddit = praw.Reddit(client_id='KvRsJ7d8P0y7sQ', client_secret='lp3sUKus_afFOSKcYMtVWDOElso', user_agent='Text1_Scraper'

    subreddit = reddit.subreddit(subreddit)

    posts = []
    comments = []
    comments_fo = []


    #get posts
    for post in subreddit.hot(limit=num_posts):
            posts.append([post.title, post.score, post.id, post.subreddit, post.url, post.num_comments, post.selftext, datetime.

    posts = pd.DataFrame(posts,columns=['title', 'score', 'p_id', 'subreddit', 'url', 'num_comments', 'body', 'p_timestamp']

    #get all comments, identify first-order comments
    for post_id in posts['p_id']:
        submission = reddit.submission(id=post_id)
        for top_level_comment in submission.comments.list(): #get all comments
            if isinstance(top_level_comment, MoreComments):
                continue
            comments.append([post_id, top_level_comment.id, top_level_comment.body, datetime.datetime.fromtimestamp(top_leve

        for top_level_comment in submission.comments: #get all first-order comments
            comments_fo.append([top_level_comment.id, 'Y'])



    comments_fo = pd.DataFrame(comments_fo, columns=['c_id', 'Post_Reply'])
    comments = pd.DataFrame(comments, columns=['p_id', 'c_id', 'comment', 'c_timestamp'])


    df = pd.merge(pd.merge(posts, comments, how='left', on='p_id'), comments_fo, how='left', on='c_id')
    df['Time_to_Comment'] = df['c_timestamp'] - df['p_timestamp']
    df['Post_Reply'] = df['Post_Reply'].fillna('N')

    return df


pull_Reddit_Posts('depression', 10000).to_csv('depression_posts.csv', index=False)
pull_Reddit_Posts('suicidewatch', 10000).to_csv('suicidewatch_posts.csv', index=False)
```

4.2 Sentiment Analysis

In [34]:

```
df_depression_post_direct_reply_time = df_depression[df_depression['Post_Reply']=='Y']
df_depression_post_direct_reply_time.head()

df_depression_post_direct_reply_time['time_to_reply'] = 0

for i in range(len(df_depression_post_direct_reply_time)):
    df_depression_post_direct_reply_time['time_to_reply'].iloc[i] = calc_minutes(df_depression_post_direct_reply_time['Time_



df_depression_post_direct_reply_time = df_depression_post_direct_reply_time.groupby('p_id')['time_to_reply'].aggregate(['med

df_depression_post = pd.merge(df_depression_post, df_depression_post_direct_reply_time, how = 'left', on = 'p_id')
df_depression_post = df_depression_post.rename(columns={'median': 'median_direct_reply_time', 'min': 'min_reply_time'})

df_depression_post.head()
```

```
C:\Users\prath\Anaconda3\lib\site-packages\ipykernel_launcher.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-a-copy
  after removing the cwd from sys.path.
C:\Users\prath\Anaconda3\lib\site-packages\ipykernel_launcher.py:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-a-copy
  import sys
```

Out[34]:

	p_id	score	num_comments	p_timestamp	direct_reply_comments	direct_comments_proportion	median_direct_reply_time	min_reply_time
0	cml6ni	970	5523	2019-08-06 06:12:33	443.0	0.080210	43287.350000	5.933333
1	doqwow	458	45	2019-10-29 17:52:02	20.0	0.444444	8601.425000	51.350000
2	dzkwym	207	37	2019-11-21 17:32:06	29.0	0.783784	106.650000	15.666667
3	dzg28a	543	64	2019-11-21 10:00:35	49.0	0.765625	523.966667	1.100000
4	dzhsg2	115	13	2019-11-21 13:10:35	11.0	0.846154	340.600000	10.366667

In [36]:

```
import nltk
nltk.downloader.download('vader_lexicon')

analyser = SentimentIntensityAnalyzer()

df_depression['Sentiment Scores'] = df_depression['body_sw'].apply(analyser.polarity_scores)
df_depression['P_Sent_Neg'] = np.NaN
df_depression['P_Sent_Neu'] = np.NaN
df_depression['P_Sent_Pos'] = np.NaN
df_depression['P_Sent_Com'] = np.NaN


for i in range(len(df_depression)):
    df_depression['P_Sent_Neg'].iloc[i] = df_depression['Sentiment Scores'].iloc[i]['neg']
    df_depression['P_Sent_Neu'].iloc[i] = df_depression['Sentiment Scores'].iloc[i]['neu']
    df_depression['P_Sent_Pos'].iloc[i] = df_depression['Sentiment Scores'].iloc[i]['pos']
    df_depression['P_Sent_Com'].iloc[i] = df_depression['Sentiment Scores'].iloc[i]['compound']



df_depression_post_sentiments = df_depression.groupby('p_id')[['P_Sent_Neg', 'P_Sent_Neu', 'P_Sent_Pos', 'P_Sent_Com']].mean

df_depression_post = pd.merge(df_depression_post, df_depression_post_sentiments, how = 'left', on = 'p_id')
df_depression_post.head()
```

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

Out[36]:

	p_id	score	num_comments	p_timestamp	direct_reply_comments	direct_comments_proportion	median_direct_reply_time	min_reply_time	P_Sent_Neg	P
0	cml6ni	970	5523	2019-08-06 06:12:33	443.0	0.080210	43287.350000	5.933333	0.063	
1	doqwow	458	45	2019-10-29 17:52:02	20.0	0.444444	8601.425000	51.350000	0.119	
2	dzkwym	207	37	2019-11-21 17:32:06	29.0	0.783784	106.650000	15.666667	0.424	
3	dzg28a	543	64	2019-11-21 10:00:35	49.0	0.765625	523.966667	1.100000	0.122	
4	dzhsg2	115	13	2019-11-21 13:10:35	11.0	0.846154	340.600000	10.366667	0.173	

In [39]:

```
df_depression['Sentiment Scores'] = df_depression['comment_sw'].apply(analyser.polarity_scores)
df_depression['C_Sent_Neg'] = np.NaN
df_depression['C_Sent_Neu'] = np.NaN
df_depression['C_Sent_Pos'] = np.NaN
df_depression['C_Sent_Com'] = np.NaN


for i in range(len(df_depression)):
    df_depression['C_Sent_Neg'].iloc[i] = df_depression['Sentiment Scores'].iloc[i]['neg']
    df_depression['C_Sent_Neu'].iloc[i] = df_depression['Sentiment Scores'].iloc[i]['neu']
    df_depression['C_Sent_Pos'].iloc[i] = df_depression['Sentiment Scores'].iloc[i]['pos']
    df_depression['C_Sent_Com'].iloc[i] = df_depression['Sentiment Scores'].iloc[i]['compound']

df_depression['C_Sent_Neg'] = np.where(df_depression['comment']=='[deleted]', np.NaN, df_depression['C_Sent_Neg'])
df_depression['C_Sent_Neu'] = np.where(df_depression['comment']=='[deleted]', np.NaN, df_depression['C_Sent_Neu'])
df_depression['C_Sent_Pos'] = np.where(df_depression['comment']=='[deleted]', np.NaN, df_depression['C_Sent_Pos'])
df_depression['C_Sent_Com'] = np.where(df_depression['comment']=='[deleted]', np.NaN, df_depression['C_Sent_Com'])


df_depression_comment_sentiments = df_depression.groupby('p_id')[['C_Sent_Neg', 'C_Sent_Neu', 'C_Sent_Pos', 'C_Sent_Com']].m

df_depression_post = pd.merge(df_depression_post, df_depression_comment_sentiments, how = 'left', on = 'p_id')
df_depression_post.head()
```

Out[39]:

	p_id	score	num_comments	p_timestamp	direct_reply_comments	direct_comments_proportion	median_direct_reply_time	min_reply_time	P_Sent_Neg	P
0	cml6ni	970	5523	2019-08-06 06:12:33	443.0	0.080210	43287.350000	5.933333	0.063	
1	doqwow	458	45	2019-10-29 17:52:02	20.0	0.444444	8601.425000	51.350000	0.119	
2	dzkwym	207	37	2019-11-21 17:32:06	29.0	0.783784	106.650000	15.666667	0.424	
3	dzg28a	543	64	2019-11-21 10:00:35	49.0	0.765625	523.966667	1.100000	0.122	
4	dzhsg2	115	13	2019-11-21 13:10:35	11.0	0.846154	340.600000	10.366667	0.173	

In [45]:

```
cols_c = ['C_Neg_Feelings', 'C_Suicide_Act', 'C_Goal', 'C_Medical', 'C_FPS']
cols_p = ['P_Neg_Feelings', 'P_Suicide_Act', 'P_Goal', 'P_Medical', 'P_FPS']

lists = [neg_feelings, suicide_act, goal, medical, fps]


for i in range(len(cols_c)):
    for j in range(len(df_depression['comment_lemmatized'])):

        count = 0
        for k in range(len(df_depression['comment_lemmatized'].iloc[j])):
            if(df_depression['comment_lemmatized'].iloc[j][k].lower() in lists[i]):
                count += 1
        if(len(df_depression['comment_lemmatized'].iloc[j])==0):
            df_depression[cols_c[i]].iloc[j] = 0
        else:
            df_depression[cols_c[i]].iloc[j] = count/len(df_depression['comment_lemmatized'].iloc[j])


for i in range(len(cols_p)):
    for j in range(len(df_depression['body_lemmatized'])):

        count = 0
        for k in range(len(df_depression['body_lemmatized'].iloc[j])):
            if(df_depression['body_lemmatized'].iloc[j][k].lower() in lists[i]):
                count += 1
        if(len(df_depression['body_lemmatized'].iloc[j])==0):
            df_depression[cols_p[i]].iloc[j] = 0
        else:
            df_depression[cols_p[i]].iloc[j] = count/len(df_depression['body_lemmatized'].iloc[j])
```

In [46]:

```
df_depression_frequencies = df_depression.groupby('p_id')['C_Neg_Feelings', 'C_Suicide_Act', 'C_Goal',
                                                          'C_Medical', 'C_FPS', 'P_Neg_Feelings', 'P_Suicide_Act',
                                                          'P_Goal', 'P_Medical', 'P_FPS'].median()

df_depression_post = pd.merge(df_depression_post, df_depression_frequencies, how = 'left', on = 'p_id')
```

In [47]:

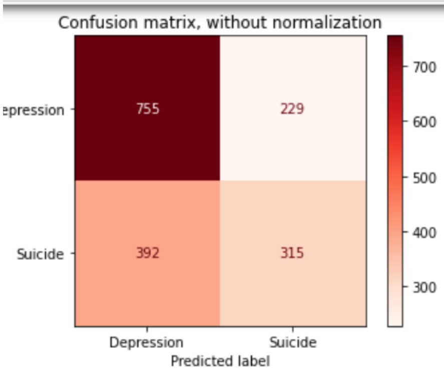
```
df_depression_post.head(5)
```

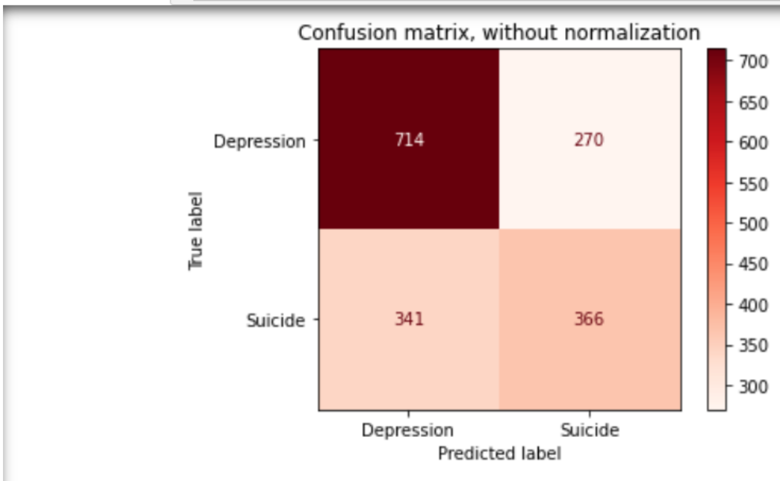
Out[47]:

	p_id	score	num_comments	p_timestamp	direct_reply_comments	direct_comments_proportion	median_direct_reply_time	min_reply_time	P_Sent_Neg	P
0	cml6ni	970	5523	2019-08-06 06:12:33	443.0	0.080210	43287.350000	5.933333	0.063	
1	doqwow	458	45	2019-10-29 17:52:02	20.0	0.444444	8601.425000	51.350000	0.119	
2	dzkwym	207	37	2019-11-21 17:32:06	29.0	0.783784	106.650000	15.666667	0.424	

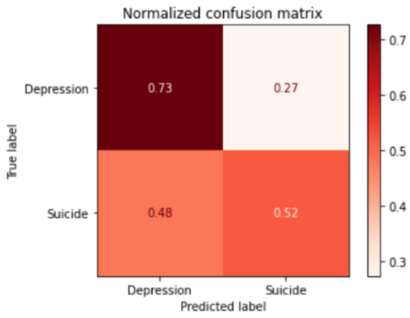
4.3 Classification

4.3.1 Confusion Matrix without Normalization





4.3.2 Confusion Matrix with Normalization



4.3.3 Unigrams and Bigrams

```
LogisticRegression
Unigrams and bigrams
[[755 229]
 [392 315]]
Confusion matrix, without normalization
[[755 229]
 [392 315]]
Normalized confusion matrix
[[0.77 0.23]
 [0.55 0.45]]
```

Output:

```
Top 5 unigrams in suicide set
Suicide most common words:

Out[1]: [('i', 3635),
 ('to', 3517),
 ('you', 3058),
 ('and', 2365),
 ('the', 2222),
 ('a', 1954),
 ('it', 1536),
 ('that', 1462),
 ('of', 1259),
 ('is', 1215),
 ('for', 1112),
 ('but', 1067),
 ('in', 1002),
 ('my', 930),
 ('your', 830),
 ('have', 812),
```

4.4 Topic Modelling

4.4.1 Latent Dirichlet Allocation (LDA)

```
In [21]: # Print the Keyword in the 10 topics
pprint(lda_model.print_topics())
doc_lda = lda_model[corpus]

[(0,
  '0.052*"change" + 0.045*"back" + 0.042*"therapy" + 0.036*"experience" + '
  '0.034*"seem" + 0.030*"person" + 0.025*"put" + 0.024*"able" + 0.022*"maybe" '
  '+ 0.022*"enough"'),
 (1,
  '0.147*"money" + 0.077*"world" + 0.053*"easy" + 0.043*"pretty" + '
  '0.043*"treat" + 0.033*"truly" + 0.032*"exactly" + 0.032*"beautiful" + '
  '0.032*"human" + 0.029*"true"'),
 (2,
  '0.291*"thank" + 0.084*"reach" + 0.067*"whole" + 0.035*"free" + 0.032*"safe" '
  '+ 0.024*"sub" + 0.022*"develop" + 0.017*"clear" + 0.009*"misery" + '
  '0.007*"regard"'),
 (3,
  '0.052*"feel" + 0.035*"know" + 0.033*"get" + 0.030*"thing" + 0.029*"make" + '
  '0.028*"go" + 0.023*"well" + 0.022*"life" + 0.022*"time" + 0.022*"good"'),
 (4,
  '0.065*"big" + 0.058*"option" + 0.055*"case" + 0.048*"include" + '
  '0.041*"chance" + 0.035*"deep" + 0.034*"progress" + 0.030*"share" + '
  '0.029*"useless" + 0.022*"drop"'),
 (5,
  '0.098*"word" + 0.078*"kind" + 0.064*"push" + 0.048*"possible" + '
  '0.047*"appreciate" + 0.046*"instead" + 0.041*"listen" + 0.041*"comment" + '
  '0.036*"close" + 0.029*"example"'),
 (6,
  '0.000*"vote" + 0.000*"storytelle" + 0.000*"sky" + 0.000*"roar" + '
  '0.000*"riding" + 0.000*"horseback" + 0.000*"horse" + 0.000*"comet" + '
  '0.000*"thor" + 0.000*"witness"'),
```

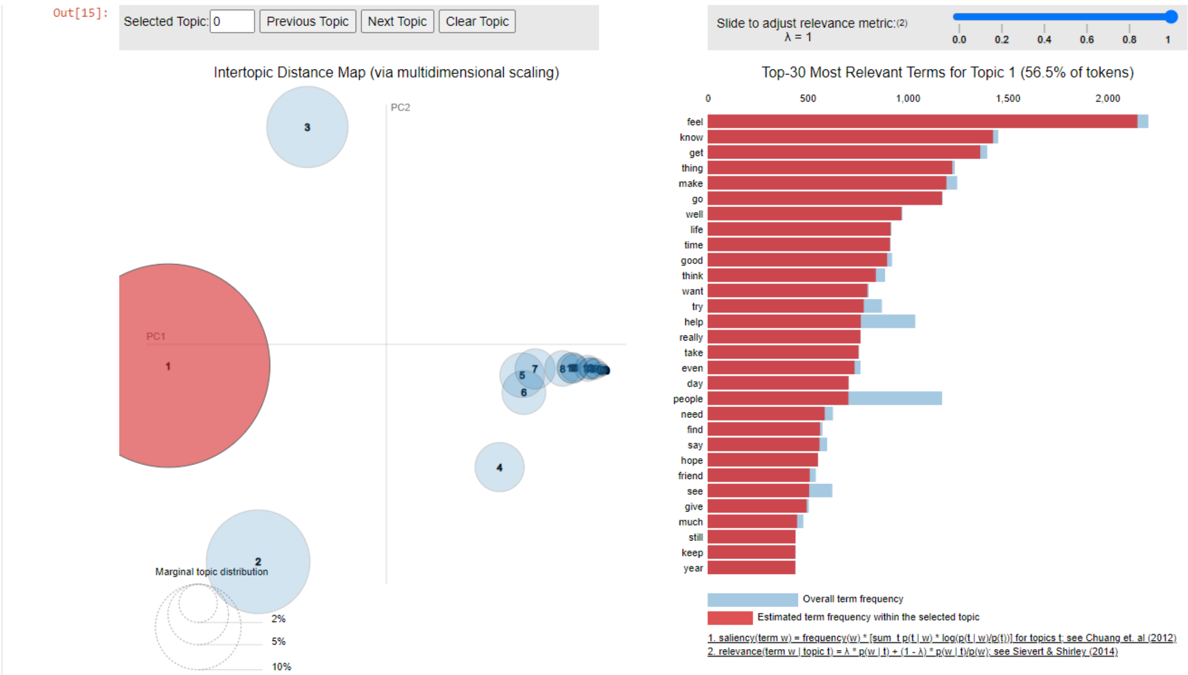
```
(7,
  '0.171*"fuck" + 0.083*"social" + 0.069*"totally" + 0.062*"send" + '
  '0.059*"accept" + 0.034*"expect" + 0.000*"date" + 0.000*"trauma" + '
  '0.000*"walk" + 0.000*"reality"'),
 (8,
  '0.093*"problem" + 0.081*"sorry" + 0.075*"therapist" + 0.074*"sound" + '
  '0.048*"question" + 0.038*"answer" + 0.038*"important" + 0.029*"speak" + '
  '0.029*"allow" + 0.028*"wrong"'),
 (9,
  '0.162*"definitely" + 0.066*"aware" + 0.031*"contact" + 0.018*"private" + '
  '0.005*"paint" + 0.000*"self_esteem" + 0.000*"therapeutic" + 0.000*"kid" + '
  '0.000*"tooth" + 0.000*"d"'),
 (10,
  '0.049*"depression" + 0.045*"way" + 0.041*"people" + 0.035*"tell" + '
  '0.034*"look" + 0.023*"wish" + 0.020*"ask" + 0.019*"use" + 0.018*"depressed" '
  '+ 0.017*"relationship"'),
 (11,
  '0.099*"far" + 0.090*"lol" + 0.053*"high" + 0.047*"absolutely" + '
  '0.036*"hell" + 0.035*"truth" + 0.032*"effect" + 0.031*"hopefully" + '
  '0.016*"eye" + 0.014*"carry"'),
 (12,
  '0.085*"top" + 0.062*"stupid" + 0.025*"welcome" + 0.000*"degree" + '
  '0.000*"school" + 0.000*"sun" + 0.000*"bright" + 0.000*"tomorrow" + '
  '0.000*"shine" + 0.000*"amount"'),
 (13,
  '0.188*"sad" + 0.000*"apparent" + 0.000*"parent" + 0.000*"tear" + '
  '0.000*"bed" + 0.000*"halfway" + 0.000*"constantly" + 0.000*"lowkey" + '
  '0.000*"burst" + 0.000*"entire"'),
 (14,
  '0.196*"new" + 0.064*"literally" + 0.034*"soul" + 0.028*"impossible" + '
  '0.026*"harm" + 0.021*"opinion" + 0.018*"reject" + 0.012*"individual" + '
  '0.009*"necessary" + 0.008*"message"'),
```

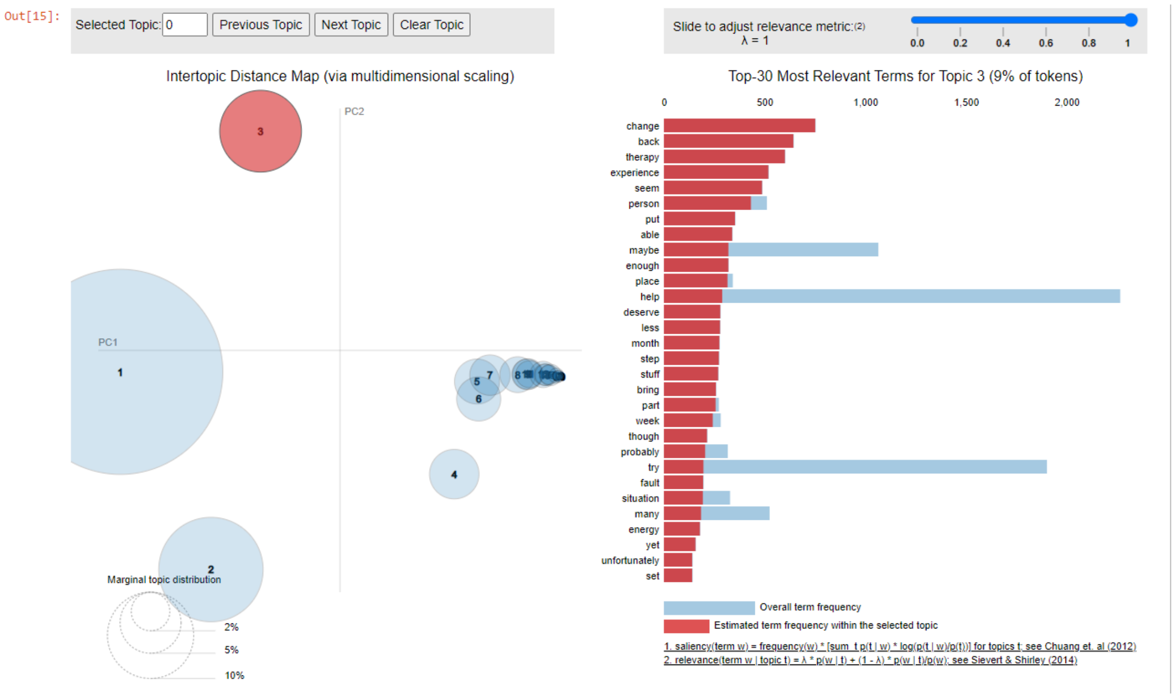
```
  '0.009*"necessary" + 0.008*"message"'),
 (15,
  '0.087*"awful" + 0.069*"fun" + 0.067*"remove" + 0.062*"super" + '
  '0.036*"honest" + 0.033*"sick" + 0.022*"dislike" + 0.020*"number" + '
  '0.013*"internet" + 0.011*"friendly"'),
 (16,
  '0.098*"suck" + 0.096*"depress" + 0.094*"break" + 0.066*"soon" + '
  '0.065*"food" + 0.043*"follow" + 0.040*"ignore" + 0.031*"effort" + '
  '0.021*"perhaps" + 0.019*"otherwise"'),
 (17,
  '0.253*"work" + 0.098*"job" + 0.041*"real" + 0.037*"trust" + '
  '0.031*"eventually" + 0.029*"couple" + 0.028*"easily" + 0.023*"house" + '
  '0.021*"room" + 0.020*"disorder"'),
 (18,
  '0.154*"feeling" + 0.111*"shit" + 0.064*"girl" + 0.054*"notice" + '
  '0.054*"glad" + 0.053*"state" + 0.025*"term" + 0.022*"boat" + '
  '0.020*"side_effect" + 0.017*"track"'),
 (19,
  '0.153*"m" + 0.104*"hour" + 0.082*"completely" + 0.080*"call" + '
  '0.036*"resource" + 0.030*"genuinely" + 0.007*"heavy" + 0.000*"movie" + '
  '0.000*"s" + 0.000*"watch"')]
```

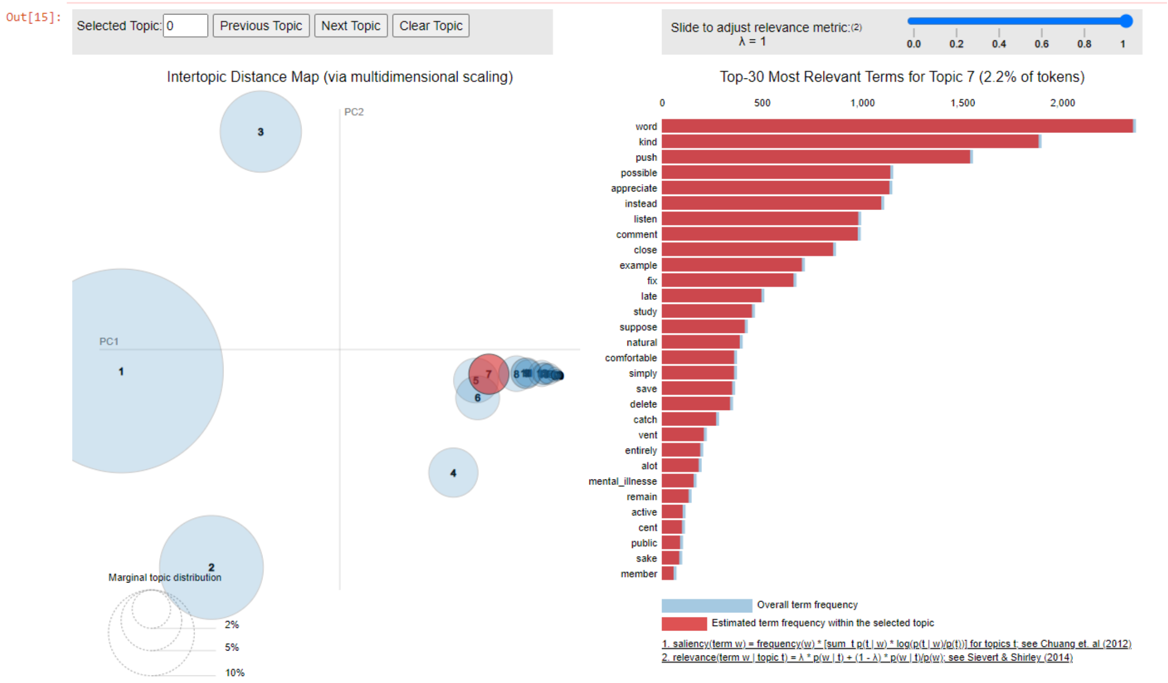
Chapter 5

Result

5.1 Topic Modelling Visualization







5.2 Perplexity and Coherence Score

```
# 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:  -12.950990755787092

Coherence Score:  0.4132638751252635
```

Chapter 6

Conclusion and Future Enhancement

6.1 Conclusion

- Starting with the reddit recent post data by scraping subreddits, we have implemented few methods and classification algorithms like, TF-IDF Vectorization, Lemmatization, Text Modelling, Logistic Regression and MDS Visualization for classifying dataset as suicidal and depressed posts.

- Thus, we used the above methods after learning and researching about existing model to make a proposed system.

6.2 Future Enhancement

- We can channel the classified data, integrate it with mental health counselling apps.
- We can publish a survey on which words are most frequently used when a person has suicidal thoughts, so that it can be useful for research purposes.

Bibliography

[1] Kali Cornn. Identifying Depression on social media, 2020. https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1194/reports/custom/15712307.pdf

[2] Viridiana Romero Martinez. A machine learning approach for the detection of depression and mental illness in Twitter, 2019.

https://medium.datadriveninvestor.com/a-machine-learning-approach-for-detection-of-depression-and-mental-illness-in-twitter-3f3a32a4df60

[3] Scikit learn Developers. Gaussian Mixture Models, 2021

https://scikit-learn.org/stable/modules/mixture.html

[4] Yamini, Analytics Vidya. Topic Modelling, 2021

https://www.analyticsvidhya.com/blog/2021/05/topic-modelling-in-natural-language-processing/