Natural Language Processing

AUTOMATIC IT SUPPORT TICKET ASSIGNMENT

TEAM

RADHIKA KULKARNI
SYED TALIBUDDIN SAIFI
TANVEER QURAISHI
SHANMUGAM SAMPATH
KANWAL RAI

MENTOR

ASHISH VYAS

Table of Content

Interim Report

Final Report

1 Project Planning Phase

Project Background, Data and Objectives

Background

One of the key activities of any IT function is to "Keep the lights on" to ensure there is no impact to the Business operations. IT leverages **Incident Management** process to achieve the above Objective. An incident is something that is unplanned interruption to an **IT service** or reduction in the quality of an IT service that affects the Users and the Business. The main **goal** of Incident Management process is **to provide a quick fix / workarounds or solutions that** resolves **the interruption and restores the service to its full capacity to ensure no business impact**. In most of the organizations, incidents are created by various Business and IT Users, End Users/ Vendors if they have access to ticketing systems, and from the integrated monitoring systems and tools.

Assigning the incidents to the appropriate person or unit in the support team has critical importance to provide improved user satisfaction while ensuring better allocation of support resources. The assignment of incidents to appropriate IT groups **is still a manual process** in many of the IT organizations. Manual assignment of incidents is time consuming and requires human efforts. There may be mistakes due to human errors and resource consumption is carried out ineffectively because of the misaddressing. On the other hand, **manual assignment increases the response and resolution times which result in user satisfaction deterioration / poor customer service.**

In the support process, incoming incidents are analyzed and assessed by organization's support teams to fulfill the request. In many organizations, better allocation and effective usage of the valuable support resources will directly result in substantial cost savings. Currently the **incidents** are created by various stakeholders (Business Users, IT Users and Monitoring Tools) within IT Service Management Tool and **are assigned to Service Desk teams (L1 / L2 teams).** This team will review the incidents for right ticket categorization, priorities and then carry out initial diagnosis to see if they can resolve. Around ~54% of the incidents are resolved by L1 / L2 teams. In case L1 / L2 is unable to resolve, they will then escalate / assign the tickets to Functional teams from Applications and Infrastructure (L3 teams). Some portions of incidents are directly assigned to L3 teams by either Monitoring tools or Callers / Requestors. L3 teams will carry out detailed diagnosis and resolve the incidents. Around ~56% of incidents are resolved by Functional / L3 teams. Incase if vendor support is needed, they will reach out for their support towards incident closure.

L1 / L2 needs to spend time reviewing Standard Operating Procedures (SOPs) before assigning to Functional teams (Minimum ~25-30% of incidents needs to be reviewed for SOPs before ticket assignment). 15 min is being spent for SOP review for each incident. Minimum of ~1 FTE effort needed only for incident assignment to L3 teams. During the process of incident assignments by L1 / L2 teams to functional groups, there were multiple instances of incidents getting assigned to wrong functional groups. Around ~25% of Incidents are wrongly assigned to functional teams. Additional effort needed for Functional teams to re-assign to right functional groups. During this process, some of the incidents are in queue and not addressed timely resulting in poor customer service. Guided by powerful AI techniques that can classify incidents to right functional groups can help organizations to reduce the resolving time of the issue and can focus on more productive tasks.

Data

Details about the data and dataset files are given in below link: https://drive.google.com/file/d/1OZNJm81JXucV3HmZroMq6qCT2m7ez7IJ/view

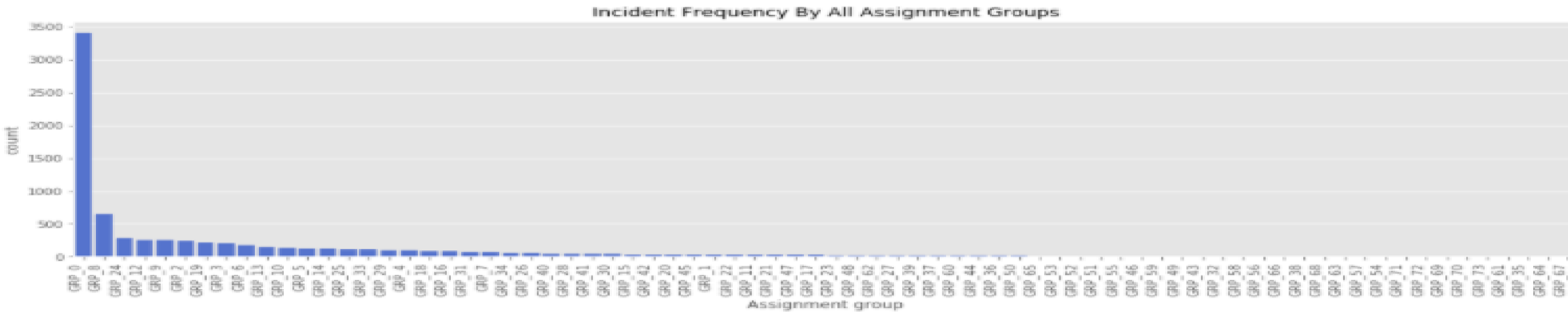
Objectives

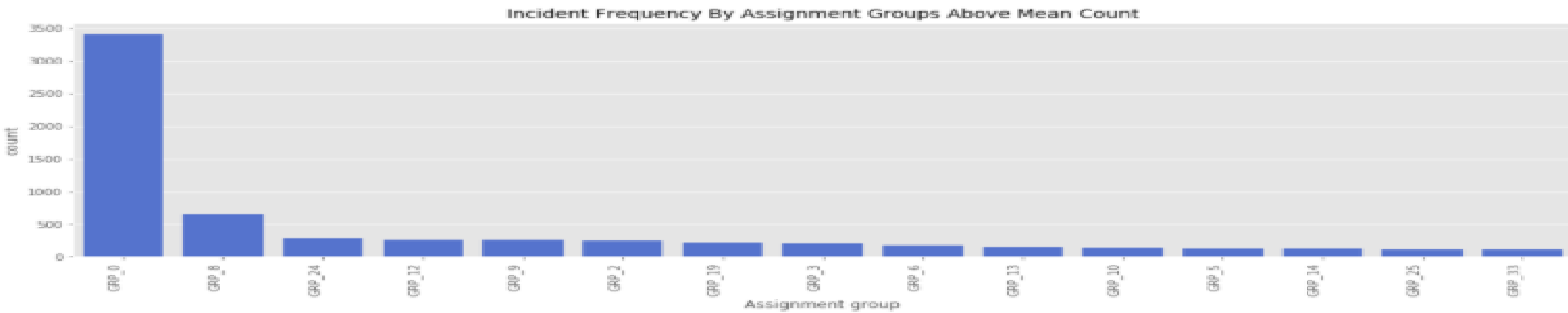
In this capstone project, the goal is to build a classifier that can classify the tickets by analyzing text. The objective of the project is, (a) Learn how to use different classification models. (b) Use transfer learning to use pre-built models. (c) Learn to set the optimizers, loss functions, epochs, learning rate, batch size, checkpointing, early stopping etc. (d) Read different research papers of given domain to obtain the knowledge of advanced models for the given problem. As per the background, the existing system assigns 70-75% of the support tickets effectively. Our objective in terms of effectiveness would be to build AI models to classify these tickets with an accuracy of at lest 80-85%.

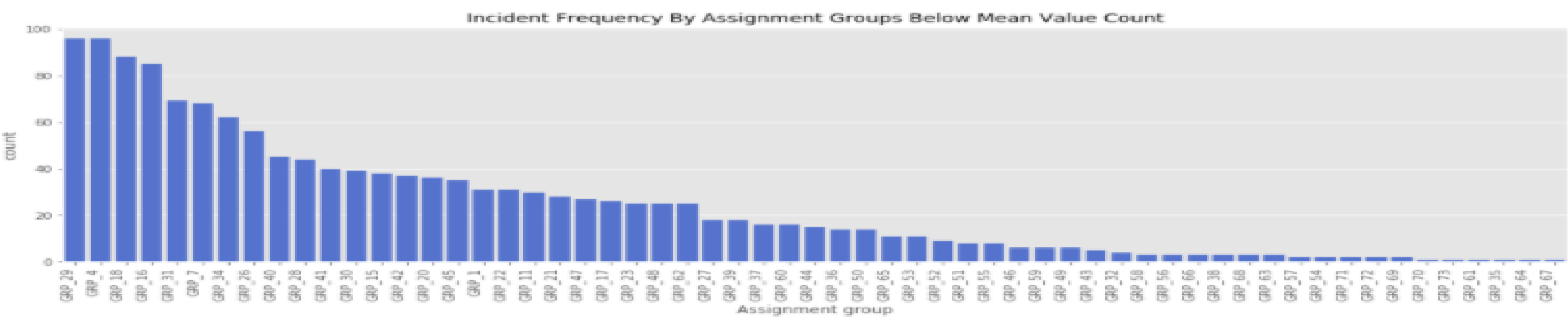
Exploratory Data Analysis

- **Dataset :** Dataset Structure (4 Variables, 8500 Incident Records): Caller, Short Description, Description and Assignment Group
  - Data Features: Caller (2950 unique count), Short Description, Description
  - Label/Target Class: Assignment Group (74)
- **Features**
  - Caller names are ad-hoc and without proper references to any master data. It is not advised to be used as a feature as it will amount to overfitting the data. We expect these user or call names to be anything in the current scope of themes as making a prediction or classification of a ticket based on the caller names is not justifiable rationally. We plan to consider Short Description, Description as the data for feature engineering and ignore Caller related information in the dataset.
  - Description are predominantly in English with words from other European languages as well. We plan to keep the data unmodified as received from the source. However for the sake of learning and experimentation, we plan to detect the languages and try to convert certain words into English.
  - Descriptions are unstructured akin to reviews, chats, e-mails or tweets and its not clean. There are for instance special characters, emails, dates, symbols, hyper links, URL, IP Addresses and missing values or excessive whitespaces and stop words. We plan to remove the stop words and unnecessary noise through regular expression before vectorization of the feature data.
  - We planned to use information in the Short Description along with the information in the Description column by concatenation of the data. Descriptions in some cases are though same as the data in the Short Description.
  - Description contains conjugated words i.e. they are not properly separated and the text is submitted without prior spell corrections (spelling errors) including the fact that there are descriptions that contain sentences which are not grammatically obvious or otherwise meaningful to be analyzed through human intervention.
- **Labels/Target Class**
  - The label data provided in the target class is imbalanced and skewed. Most of the tickets (3976, almost 50%) are being assigned to a single group (GRP_0). There are also assignment groups which have been assigned only a single ticket in the given dataset. We are earlier planned to merge these smaller groups into a single group to reduce the imbalance. This could have solved the imbalance problem. However, we are avoiding add-on group creations as we want the models to predict on the feature data as it is because these assignment groups are dynamic data and would keep changing. Hence using group names as a basis of reclassification will only serve a temporary purpose and even otherwise, it will lead to overfitting of the data. Hence we have kept the data as it is and addressed the imbalance through methods like SMOTE and RESAMPLING. In some cases the number of incident records are less than six in a target class and hence SMOTE can not be applied as it is. We need a suitable combination of sampling methods to prepare the dataset which is balanced and right one to be fed to the models.

Incident Frequency By Assignment Groups

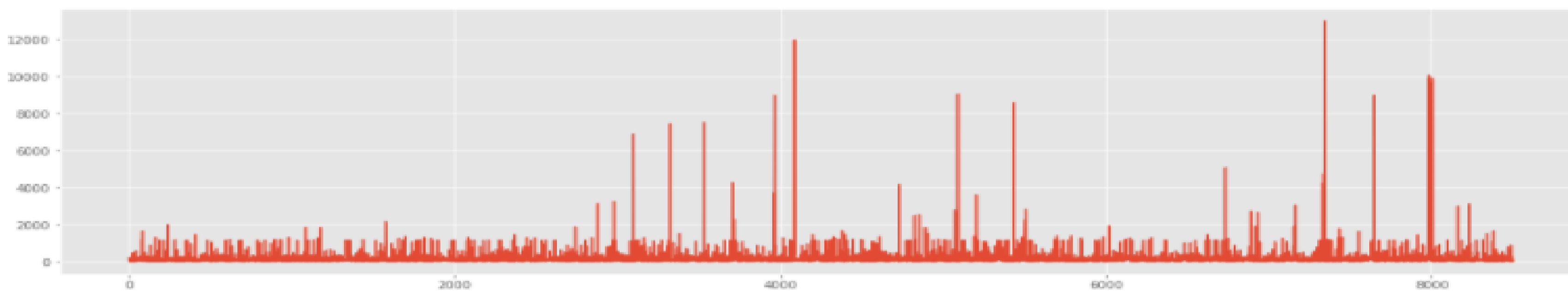


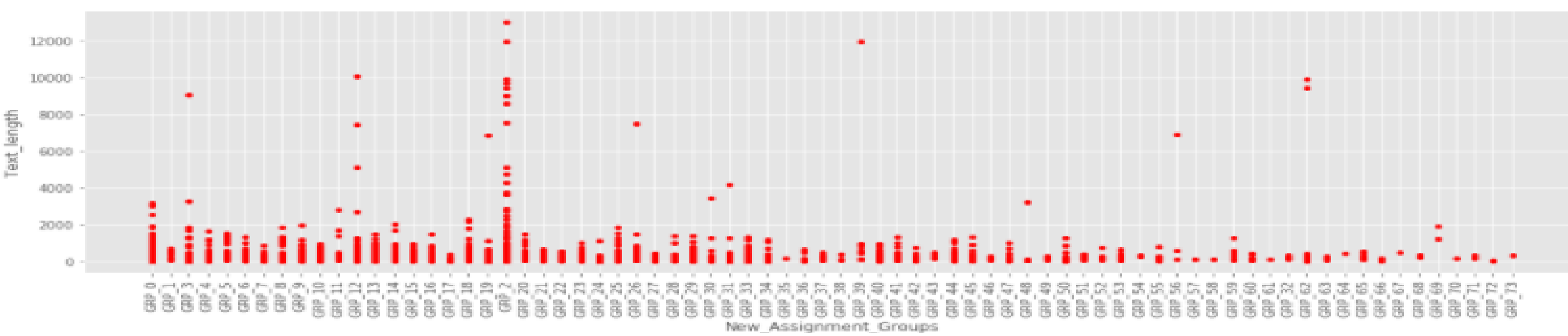




Data Pre-Processing & Input Dataset

- **Feature Set :** We have dropped the Caller information and concatenated the two description columns. Also there were 25 assignment groups having less than 10 incidents. However we not regrouped them and kept the assignment groups intact as provided in the dataset and planned to used resampling methods to make them balanced. We also measured the length of the text used in the incidents across the assignment groups. The length varied from 1 (after the null value being replaced with a space) to 13001 words with an average length of 217 words. There were 5 incidents with less than 3 words. We kept them as included in the dataset as an input to the models.





Data Pre-Processing & Input Dataset

- **Missing Values & Duplicates :** We have dropped duplicate records as they don’t additional information or value for classification. We have replaces the null values a space. There were 83 duplicate records across all columns, 682 duplicate records by the value in Description column and 661 duplicate records having same values in Description and Assignment Group together. This basically means there were 27 incidents which were same but were assigned to different assignment group. After removing duplicate records we had a dataset reduced from 8500 to 7839. There were 5 incidents without any short description and 1 incident without a description. We converted these null values into spaces.

- **Upper & Lower Case Text:** We have converted the feature set in lower case as this helps simplify addressing the cleaning function through regular expression. Otherwise we would land up building more and more filtering criteria to account for the two different cases. Also the feature data based prediction as such does not influence the classification methods in any way.

- **Special Characters & Stop Words :** We have removed various characters like hashtags and kept many of them in the concatenated feature set to subject to the cleaning using regular expression. We planned to use the NLTK for removal of stop words.

- **Language & Spelling Errors & Lemmatization:** We have kept the text as it is. However for experimentation and learning perspective we have tried to translate some of the words from non-English to English. We have also tried correcting some of the typical spelling errors. We have then used lemmatization on the processed or clean text.

- **Word Cloud & Topic Modeling & N-Grams:** We have used word cloud to visualize the frequently existing words and have also performed topic modeling (using Gensim LDA Model) to understand the correlated words in the text across the incident reports i.e. grouping the tickets or incidents based on the words used in the text. We have also tried to plot the number of words used in each of the tickets to study the outliers.

- **Word Embedding & Vectorization:** We plan to use N-Grams (bi and tri) from the bag of words along with Word2Vec and Glove Embedding for vectorization and preparation of dataset.

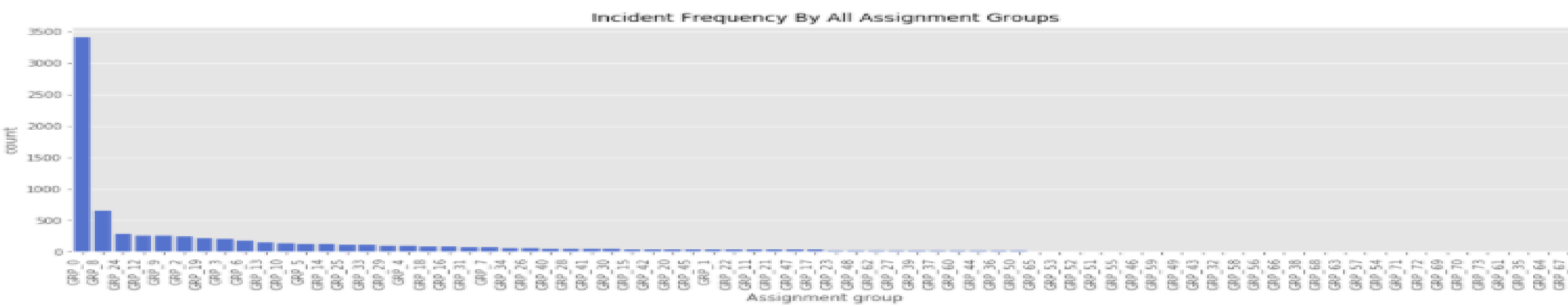
2 Project Execution Phase

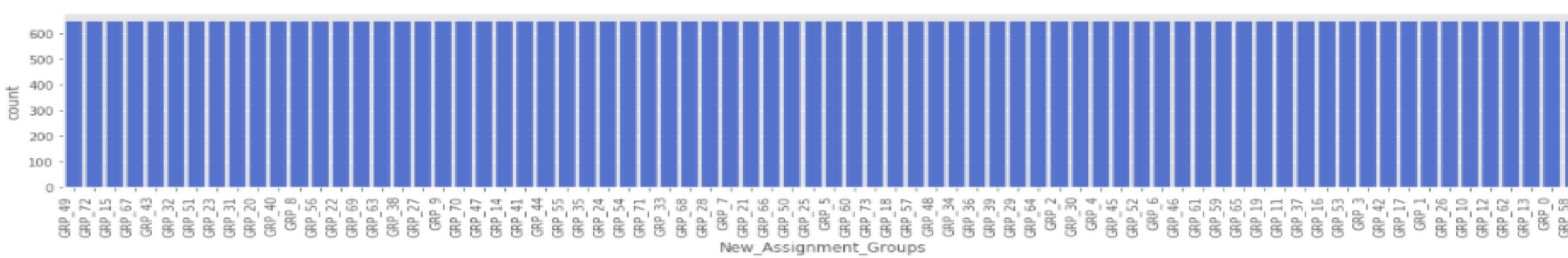
1.1

Dataset Preparation

Data Pre-Processing

The target class is imbalanced. We have many classes with less than 10 incidents. We have avoided the temptation to group such scantly populated classes in order to balance the input dataset. Instead we planned to use the resampling and SMOTE and class weights to address the target class imbalance. Our earlier plan was to build three different dataset. First to have GRP_0 and rest of the records classified into one and use two step classification model and the second one to have non-GRP_0 groups resampled for the second order classification and third dataset in which we keep all the groups (74) as it is and address the imbalance issue. After due considerations and discussions, we finally decided to pursue only the third dataset so that we don't tend to force fit the data into model. In future, these group names could be changed any time and fulcrum of imbalance could shift to some other group. In such a situation, this hard coded approach as followed in case of the first two datasets, we fail to deliver the right classification. It could fail miserably. However, all the data related views are presented below.





Data Cleaning

- We created a function to clean the dataset using regular expressions after converting the feature data into lower case. We removed various unwanted text or characters as identified noise in the information. We initially thought to use translation and spell correction function as well as there were many non-English words and typing errors. However we decided not to touch the actual data and leave these actions to be undertaken at the source. At the same time converting the data from one form to another, wont have any significant impact on the classification model. At the same time, we planned to use BERT and try the same dataset with it compare the performance of the models on the uniform or same dataset. We also did not filter the records having less than certain threshold value. However we analyzed the same and the details are provided below. The code used a variable to define this threshold value and hence could be used in future in case one wants to filter the records based on the number of words in the feature data. We used NLTK to remove the stop words and later analyzed the same to see these are not appearing the word cloud.

- Lemmatization was carried out subsequently before tokenization for vectorization using word embeddings. We limited ourselves to use lemmatization instead of stemming based on the quality of the data to avoid further dilution of the input feeds. Spacy was used for lemmatization. PyLDAvis was used to plot the topics and analyze the texts using N-Gram models for clustering relevant data using Gensim.

- We also restricted usage of spelling errors on the same ground as there are multi-lingual texts which might as well are not spelled accurately. Hence both the translation and spelling corrections might not be effective at the stage from classification perspective. Based on our research, we have found that these two aspect of language processing wont provide any substantial improvement in the model accuracy.

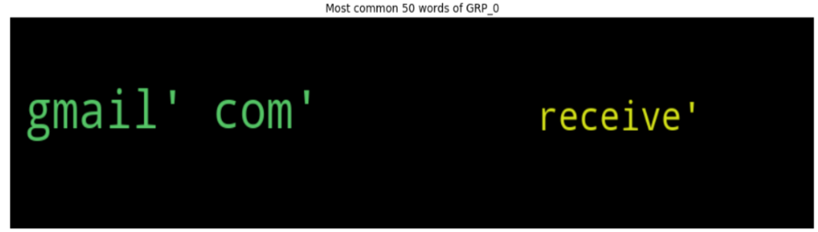
Word Cloud

We used word cloud visualization for checking if the stop words are removed effectively as well as identify the words that represent the particular target class based on the frequency or importance. We generated the word cloud for the groups, however we are sharing a snapshot in the report for few of them having texts with certain threshold value as set as a criteria while generating them.

BI-GRAMS

TRI-GRAMS

Most common 50 words of GRP_0



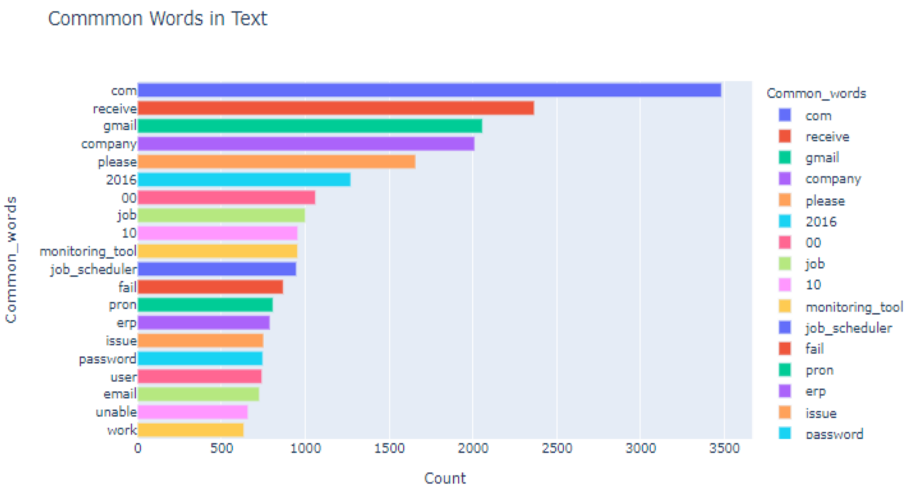
Most common 50 words of GRP_8

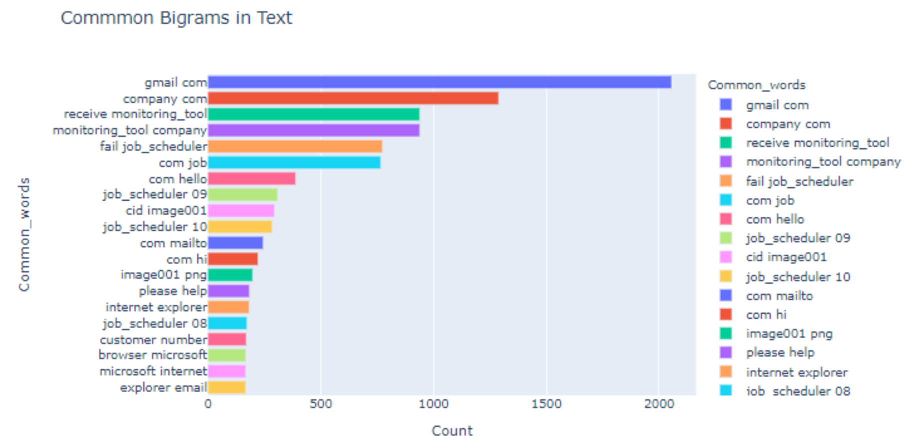
Most common 50 words of GRP_24

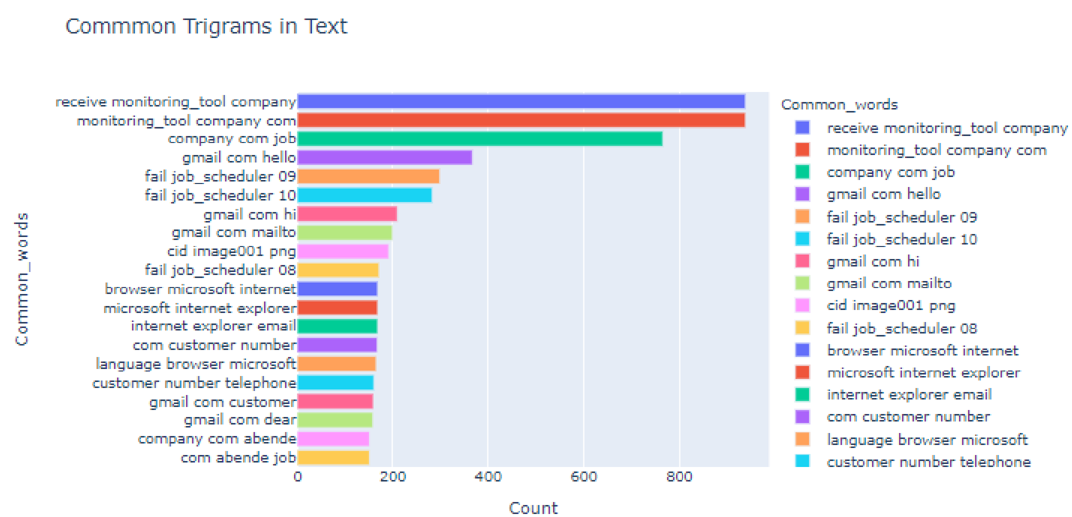
Most common 50 words of GRP_12

N-Gram Visualization

COMMON WORDS, BI-GRAMS & TRI-GRAMS

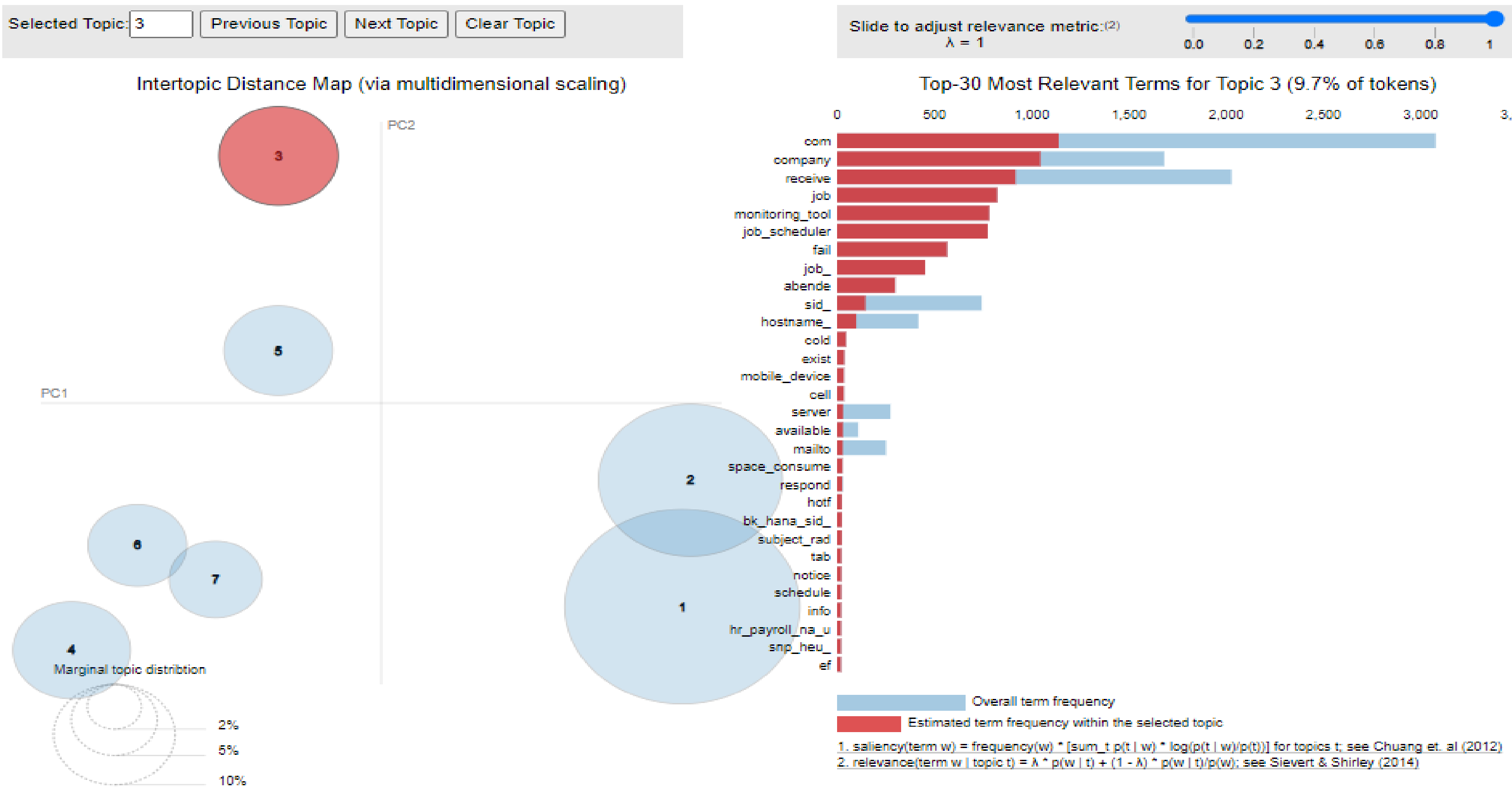






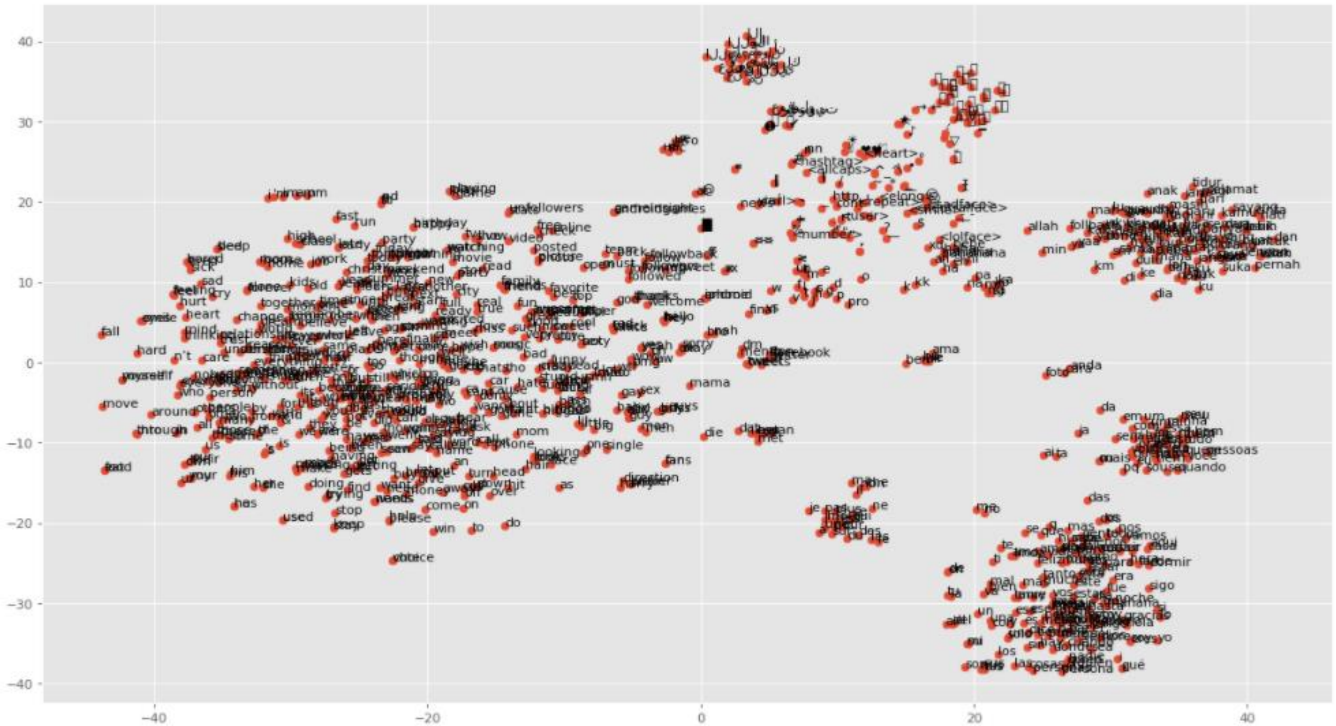
Topic Modeling

Typically in a text domain, EDA can have many meanings: What are the topics? How frequent they are? The process will involve some level of preprocessing steps. We analyzed the incidents based on the number of words present in them. We plan to extend its utility in dimensionality reduction in next iteration.



Word Embedding

Word Embedding is the efficient way for text data vectorization and we have used both Word2Vec and GloVe embedding to experiment with the models. The first one is shallow two layer NN trained on large corpus of text to produce a vector for a text (each word) in the corpus. GloVe is unsupervised learning based vector representation of words. The training is performed on an aggregated global word-word co-occurrence statistics from a corpus and the resulting representations have linear substructures of the words in the vector space. We have explored both the types of embedding with LSTM model. We have not encountered significant difference in terms of its impact on the model performance.



1.2

Model Development

Model Selection & Development

We planned to use both Deep Learning and Machine Learning models for classifying the incidents by assignment groups with the existing and same dataset after suitable cleaning and vectorization.:

- **Deep Learning**
  - **Bidirectional LSTM – With Word2Vec and Glove Embedding**
  - **GRU**
  - **RNN**
  - **BERT**
- **Machine Learning Models**
  - **Random Forest Classifier – With and Without Class Weight**
  - **Support Vector Machine**
  - **Naïve Bayes**
  - **Adaboost**
  - **Bagging**
  - **GradientBoost**
  - **KNN**

1.3

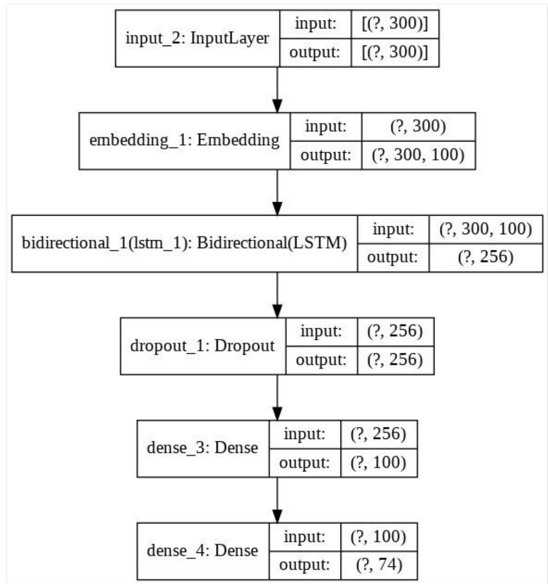
Model Evaluation

Bi-Directional LSTM Model [Word2Vec]

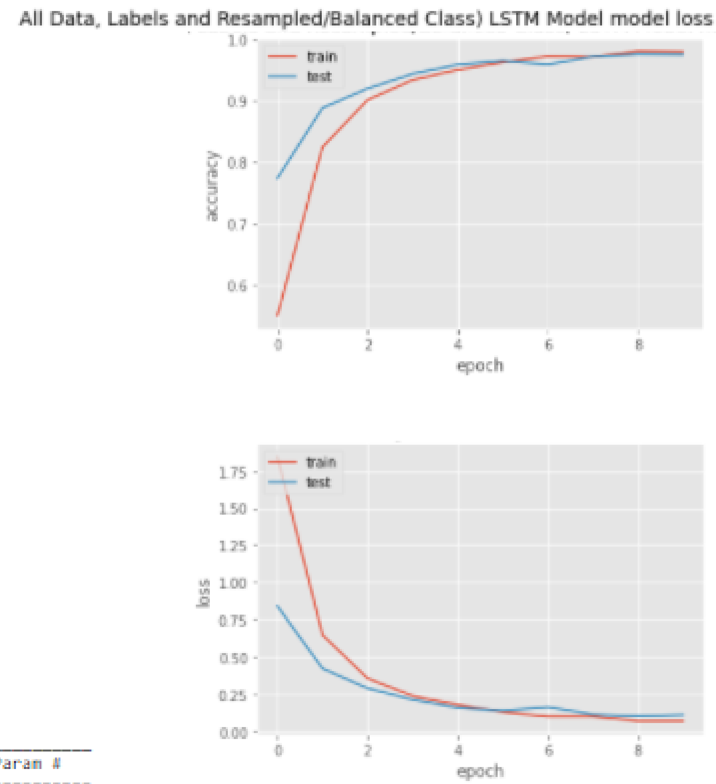
LSTM stands for “long short-term memory” and it’s a type of recurrent unit that has become very popular in recent years due to its superior performance and the fact that it doesn’t as easily succumb to the vanishing gradient problem.

Bi-Directional LSTM are a variant of traditional LSTM but improve the model performance by using the training inputs (texts/words) in a sequence (sentence) first as a normal sentence and then reading the tokens in the sentence in a reverse order. In short, for every token, in the sentence, there is a data about the tokens ahead as well as behind it with a memory parameters as defined in the model. This improves the effectiveness of the classification.

We have received an accuracy of 97% in first iteration with Word2Vec based embeddings for vectorization of the cleaned dataset.



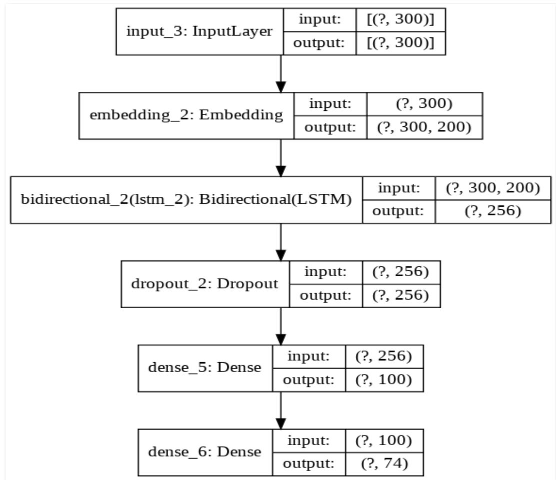

```
Number of Samples: 47730
Number of Labels:  47730
Number of train Samples: 38184
Number of val Samples: 9546
Model: "functional_9"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_5 (InputLayer)         [(None, 300)]             0
_________________________________________________________________
embedding_4 (Embedding)      (None, 300, 100)          900100
_________________________________________________________________
bidirectional_4 (Bidirection (None, 256)               234496
_________________________________________________________________
dropout_4 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_8 (Dense)              (None, 100)               25700
_________________________________________________________________
dense_9 (Dense)              (None, 74)                7474
=================================================================
Total params: 1,167,770
Trainable params: 1,167,770
Non-trainable params: 0
```



Bi-Directional LSTM Model [GloVe]

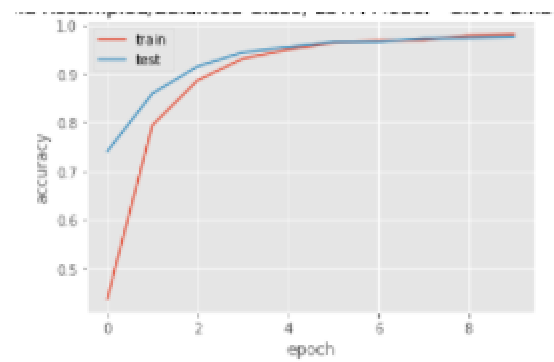
LSTM stands for "long short-term memory" and it's a type of recurrent unit that has become very popular in recent years due to its superior performance and the fact that it doesn't as easily succumb to the vanishing gradient problem. Bi-Directional LSTM are a variant of traditional LSTM but improve the model performance by using the training inputs (texts/words) in a sequence (sentence) first as a normal sentence and then reading the tokens in the sentence in a reverse order. In short, for every token, in the sentence, there is a data about the tokens ahead as well as behind it with a memory parameters as defined in the model.

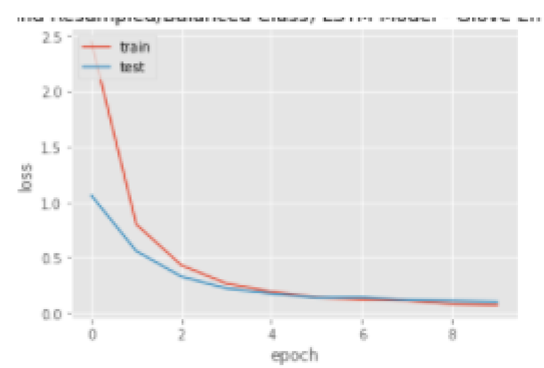
The accuracy of the model was 97.2 % which is not much different from the same model fed with Word2Vec vectors for training.




```
Number of Samples: 47730
Number of Labels:  47730
Number of train Samples: 30547
Number of val Samples: 7637
Model: "functional_11"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_6 (InputLayer)         [(None, 300)]             0
_________________________________________________________________
embedding_5 (Embedding)      (None, 300, 200)          1800200
_________________________________________________________________
bidirectional_5 (Bidirection (None, 256)               336896
_________________________________________________________________
dropout_5 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_10 (Dense)             (None, 100)               25700
_________________________________________________________________
dense_11 (Dense)             (None, 74)                7474
=================================================================
Total params: 2,170,270
Trainable params: 2,170,270
Non-trainable params: 0
```

All Data, Labels and Resampled/Balanced Class) LSTM Model - Glove Embedding model loss

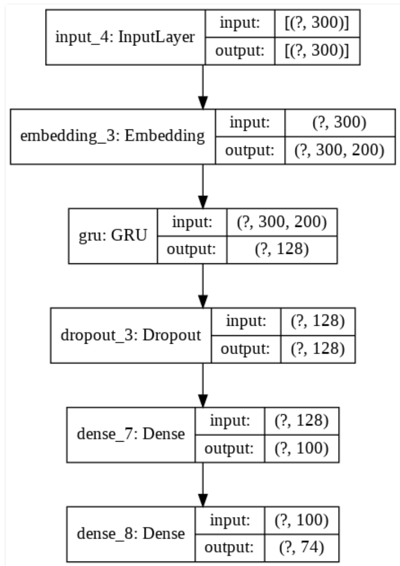




GRU

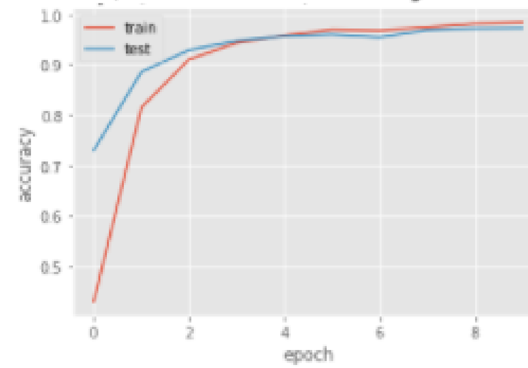
GRU (Gated Recurrent Unit) aims to solve the vanishing gradient problem which comes with a standard recurrent neural network. It can also be considered as a variation on the LSTM because both are designed similarly and, in some cases, produce equally excellent results.

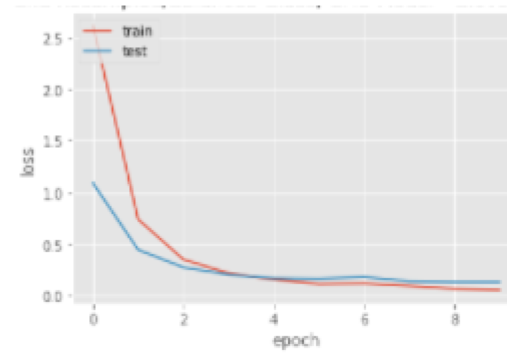
This model got rid of the cell state and used the hidden state to transfer information. It uses only 2 gates, update gate and reset gate. Basically, these are two vectors which decide what information should be passed to the output. The update gate acts similar to the forget and input gate of an LSTM. It decides what information to throw away and what new information to add. The reset gate is another gate used to decide how much past information to forget. GRU's has fewer tensor operations; therefore, they are a little speedier to train than LSTM's. We have received an accuracy of 97 % which is way above RNN model architecture.




```
Number of Samples: 47730
Number of Labels:  47730
Number of train Samples: 38547
Number of val Samples: 7637
Model: "functional_15"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_8 (InputLayer)         [(None, 300)]             0
_________________________________________________________________
embedding_7 (Embedding)      (None, 300, 200)          1800200
_________________________________________________________________
gru_1 (GRU)                  (None, 128)               126720
_________________________________________________________________
dropout_7 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_14 (Dense)             (None, 100)               12900
_________________________________________________________________
dense_15 (Dense)             (None, 74)                7474
=================================================================
Total params: 1,947,294
Trainable params: 1,947,294
Non-trainable params: 0
_________________________________________________________________
```

All Data, Labels and Resampled/Balanced Class) GRU Model - Glove Embedding model accuracy

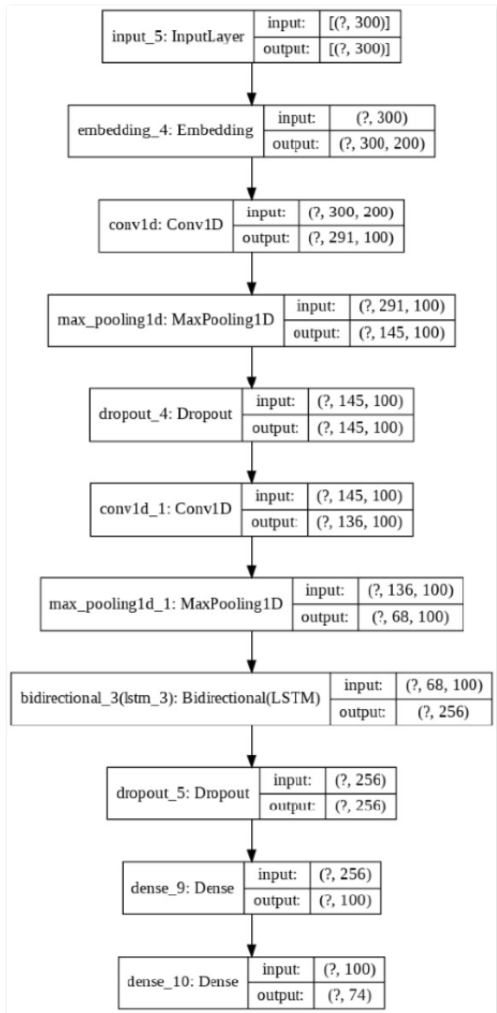




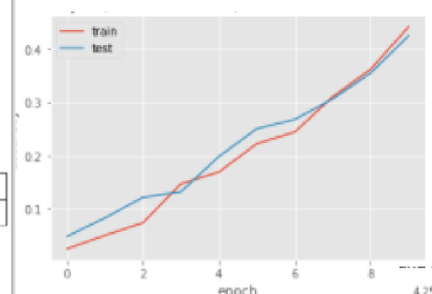
RNN

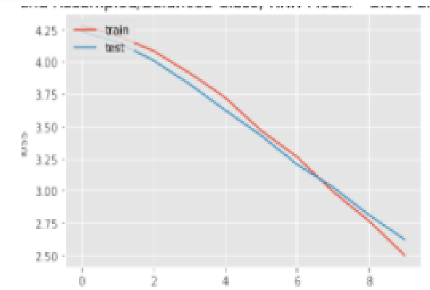
Recurrent Neural Networks (RNNs) are a family of neural networks designed specifically for sequential data processing. The RNN model does prediction of the next word in a sequence based on the previous ones. This operation is performed recurrently which is why it is called as Recurrent Neural Networks. It repetitively performs the same task for every element of a sequence, with the output being dependent on the previous computations. In short it has a memory of the text used in a sequence used for classification of the text both from the type of texts and the sequence of appearance as well. There are known inherent issues in RNN which are addressed in the subsequent model of LSTM.

We have received an accuracy of 42% but we would like further tweak with its architecture in the next iteration.




```
Number of Samples: 1432
Number of Labels:  1432
Number of train Samples: 1145
Number of val Samples: 287
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_8 (Embedding)      (None, 300, 200)          1800200
_________________________________________________________________
conv1d (Conv1D)              (None, 291, 100)          200100
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 145, 100)          0
_________________________________________________________________
dropout_8 (Dropout)          (None, 145, 100)          0
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 136, 100)          100100
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 68, 100)           0
_________________________________________________________________
bidirectional_6 (Bidirection (None, 256)               234496
_________________________________________________________________
dropout_9 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_16 (Dense)             (None, 100)               25700
_________________________________________________________________
dense_17 (Dense)             (None, 74)                7474
=================================================================
Total params: 2,368,070
Trainable params: 2,368,070
Non-trainable params: 0
```





All Data, Labels and Resampled/Balanced Class) RNN Model - Glove Embedding model accuracy

Random Forest Classifier

Tree based models work by learning in hierarchical manner. Random forests is a supervised learning algorithm which can be used both for classification as well as regression. It is also the most flexible and easy to use algorithm. It creates decision trees on randomly selected data samples, gets prediction from each tree and selects the best solution by means of voting. It also provides a pretty good indicator of the feature importance. We have received good accuracy using RFC. Also there is not much difference between with and without assigning the class weights which we plan to explore further.

RFC

	precision	recall	f1-score	support
0	0.0103	0.0137	0.0117	146
1	0.7609	0.3043	0.4348	115
2	0.3529	0.0472	0.0833	127
3	0.1686	0.2283	0.1940	127
4	0.0642	0.0511	0.0569	137
5	0.1575	0.1639	0.1606	122
6	0.0593	0.1810	0.0894	116
7	0.1677	0.2074	0.1854	135
8	0.1168	0.1185	0.1176	135
9	0.3043	0.0986	0.1489	142
10	0.5714	0.1515	0.2395	132
11	0.1667	0.0367	0.0602	109
12	0.0600	0.1282	0.0817	117
13	0.1316	0.4318	0.2018	132
14	0.7000	0.1500	0.2471	140
15	0.0357	0.0085	0.0137	118
16	0.1224	0.2266	0.1589	128
17	0.0587	0.7913	0.1092	115
18	0.6364	0.0569	0.1045	123
19	0.2764	0.2656	0.2709	128
20	0.5672	0.3065	0.3979	124
21	0.2577	0.2137	0.2336	117
22	0.5385	0.0952	0.1618	147
23	0.0769	0.0853	0.0809	129
24	0.0874	0.3597	0.1406	139
25	0.1212	0.0889	0.1026	135
26	0.6733	0.5354	0.5965	127
27	0.1765	0.1538	0.1644	117
28	0.1517	0.1803	0.1648	122
29	0.9565	1.0000	0.9778	132
30	0.8214	0.1901	0.3087	121
31	1.0000	0.0483	0.0921	145
32	0.8667	0.3023	0.4483	129
33	0.9815	0.4077	0.5761	130
34	0.7778	0.0614	0.1138	114
35	0.3488	0.1172	0.1754	128
36	0.1958	0.2276	0.2105	123
37	0.4500	0.0672	0.1169	134
38	0.7605	0.8639	0.8089	147
39	0.5165	0.3507	0.4178	134
40	0.8261	0.1557	0.2621	122
41	0.0000	0.0000	0.0000	122
42	0.9630	0.2000	0.3312	130
43	1.0000	0.0472	0.0902	127
44	0.2812	0.3025	0.2915	119
45	0.1166	0.3562	0.1757	146
46	0.6230	0.2879	0.3938	132
47	0.0000	0.0000	0.0000	124
48	0.2826	0.2185	0.2464	119
49	0.0512	0.2303	0.0838	152
50	0.0000	0.0000	0.0000	117
51	0.5060	0.3387	0.4058	124
52	0.9921	1.0000	0.9960	126
53	0.0000	0.0000	0.0000	141
54	0.8889	0.3279	0.4790	122
55	0.9818	0.4122	0.5806	131
56	1.0000	0.0072	0.0144	138
57	1.0000	0.1141	0.2048	149
58	0.9795	1.0000	0.9896	143
59	1.0000	0.1228	0.2188	114
60	0.9512	0.2955	0.4509	132
61	1.0000	1.0000	1.0000	119
62	1.0000	0.1716	0.2930	134
63	0.1786	0.2734	0.2160	128
64	1.0000	1.0000	1.0000	115
65	1.0000	1.0000	1.0000	127
66	1.0000	1.0000	1.0000	133
67	0.6250	0.0725	0.1299	138
68	0.9122	1.0000	0.9541	135
69	0.9683	0.4388	0.6040	139
70	0.0000	0.0000	0.0000	133
71	1.0000	1.0000	1.0000	128
72	0.1621	0.3361	0.2187	122
73	1.0000	0.0236	0.0462	127
accuracy			0.2981	9546
macro avg	0.5073	0.2980	0.3099	9546
weighted avg	0.5099	0.2981	0.3104	9546

RFC With Class Weights

	precision	recall	f1-score	support
0	0.8929	0.5137	0.6522	146
1	0.9746	1.0000	0.9871	115
2	0.9695	1.0000	0.9845	127
3	0.9922	1.0000	0.9961	127
4	0.9137	0.9270	0.9203	137
5	0.9030	0.9918	0.9453	122
6	0.9431	1.0000	0.9707	116
7	1.0000	1.0000	1.0000	135
8	0.9507	1.0000	0.9747	135
9	0.9660	1.0000	0.9827	142
10	0.9851	1.0000	0.9925	132
11	0.8966	0.9541	0.9244	109
12	0.8651	0.9316	0.8971	117
13	0.9925	1.0000	0.9962	132
14	1.0000	1.0000	1.0000	140
15	1.0000	1.0000	1.0000	118
16	1.0000	1.0000	1.0000	128
17	1.0000	0.9478	0.9732	115
18	0.9531	0.9919	0.9721	123
19	1.0000	1.0000	1.0000	128
20	0.9688	1.0000	0.9841	124
21	0.9669	1.0000	0.9832	117
22	1.0000	1.0000	1.0000	147
23	0.9444	0.9225	0.9333	129
24	0.9929	1.0000	0.9964	139
25	0.9640	0.9926	0.9781	135
26	1.0000	1.0000	1.0000	127
27	0.9915	0.9915	0.9915	117
28	1.0000	0.9754	0.9876	122
29	1.0000	1.0000	1.0000	132
30	1.0000	1.0000	1.0000	121
31	1.0000	1.0000	1.0000	145
32	1.0000	1.0000	1.0000	129
33	0.9924	1.0000	0.9962	130
34	0.9732	0.9561	0.9646	114
35	1.0000	1.0000	1.0000	128
36	1.0000	1.0000	1.0000	123
37	0.9926	1.0000	0.9963	134
38	1.0000	1.0000	1.0000	147
39	1.0000	1.0000	1.0000	134
40	1.0000	1.0000	1.0000	122
41	1.0000	1.0000	1.0000	122
42	1.0000	1.0000	1.0000	130
43	1.0000	1.0000	1.0000	127
44	1.0000	1.0000	1.0000	119
45	0.9295	0.9932	0.9603	146
46	0.9925	1.0000	0.9962	132
47	1.0000	1.0000	1.0000	124
48	1.0000	1.0000	1.0000	119
49	1.0000	1.0000	1.0000	152
50	1.0000	1.0000	1.0000	117
51	1.0000	1.0000	1.0000	124
52	1.0000	1.0000	1.0000	126
53	1.0000	1.0000	1.0000	141
54	1.0000	1.0000	1.0000	122
55	1.0000	1.0000	1.0000	131
56	0.9704	0.9493	0.9597	138
57	0.9933	1.0000	0.9967	149
58	1.0000	1.0000	1.0000	143
59	0.9913	1.0000	0.9956	114
60	1.0000	1.0000	1.0000	132
61	1.0000	1.0000	1.0000	119
62	1.0000	1.0000	1.0000	134
63	1.0000	1.0000	1.0000	128
64	1.0000	1.0000	1.0000	115
65	1.0000	1.0000	1.0000	127
66	1.0000	1.0000	1.0000	133
67	0.9718	1.0000	0.9857	138
68	1.0000	1.0000	1.0000	135
69	1.0000	1.0000	1.0000	139
70	0.9779	1.0000	0.9888	133
71	1.0000	1.0000	1.0000	128
72	0.9903	0.8361	0.9067	122
73	0.9680	0.9528	0.9603	127
accuracy			0.9837	9546
macro avg	0.9834	0.9842	0.9828	9546
weighted avg	0.9835	0.9837	0.9826	9546

SVM

A support vector machine (SVM) is a supervised machine learning model that uses classification algorithms for two-group classification problems. After giving an SVM model sets of labeled training data for each category, they're able to categorize the new text.

It is a fast and dependable classification algorithm that performs very well with a limited amount of data. A popular algorithm for this technique is penalized SVM.

We plan to further explore this model in next iteration for hyper parameter based optimization. As this model was taking more time for training, we have prioritized development on this front to second iteration.

	precision	recall	f1-score	support
0	0.0000	0.0000	0.0000	146
1	0.0000	0.0000	0.0000	115
2	0.0000	0.0000	0.0000	127
3	0.0000	0.0000	0.0000	127
4	0.0000	0.0000	0.0000	137
5	0.0000	0.0000	0.0000	122
6	0.0000	0.0000	0.0000	116
7	0.0000	0.0000	0.0000	135
8	0.0000	0.0000	0.0000	135
9	0.0000	0.0000	0.0000	142
10	0.0000	0.0000	0.0000	132
11	0.0153	1.0000	0.0301	109
12	0.0000	0.0000	0.0000	117
13	0.0000	0.0000	0.0000	132
14	0.0000	0.0000	0.0000	140
15	0.0000	0.0000	0.0000	118
16	0.0000	0.0000	0.0000	128
17	0.0000	0.0000	0.0000	115
18	0.0000	0.0000	0.0000	123
19	0.0000	0.0000	0.0000	128
20	0.0000	0.0000	0.0000	124
21	0.0000	0.0000	0.0000	117
22	0.0000	0.0000	0.0000	147
23	0.0000	0.0000	0.0000	129
24	0.0000	0.0000	0.0000	139
25	0.0000	0.0000	0.0000	135
26	1.0000	1.0000	1.0000	127
27	0.0000	0.0000	0.0000	117
28	0.0000	0.0000	0.0000	122
29	1.0000	1.0000	1.0000	132
30	0.0000	0.0000	0.0000	121
31	0.0000	0.0000	0.0000	145
32	1.0000	1.0000	1.0000	129
33	0.0000	0.0000	0.0000	130
34	0.0000	0.0000	0.0000	114
35	0.0000	0.0000	0.0000	128
36	0.0000	0.0000	0.0000	123
37	0.0000	0.0000	0.0000	134
38	1.0000	0.5918	0.7436	147
39	0.0000	0.0000	0.0000	134
40	0.0000	0.0000	0.0000	122
41	0.0000	0.0000	0.0000	122
42	0.0000	0.0000	0.0000	130
43	0.0000	0.0000	0.0000	127
44	0.0000	0.0000	0.0000	119
45	0.0000	0.0000	0.0000	146
46	0.0000	0.0000	0.0000	132
47	0.0000	0.0000	0.0000	124
48	0.0000	0.0000	0.0000	119
49	0.0000	0.0000	0.0000	152
50	1.0000	1.0000	1.0000	117
51	0.0000	0.0000	0.0000	124
52	1.0000	1.0000	1.0000	126
53	1.0000	1.0000	1.0000	141
54	1.0000	1.0000	1.0000	122
55	0.0000	0.0000	0.0000	131
56	0.0000	0.0000	0.0000	138
57	0.0000	0.0000	0.0000	149
58	1.0000	1.0000	1.0000	143
59	0.0000	0.0000	0.0000	114
60	1.0000	1.0000	1.0000	132
61	1.0000	1.0000	1.0000	119
62	0.0000	0.0000	0.0000	134
63	1.0000	1.0000	1.0000	128
64	1.0000	1.0000	1.0000	115
65	1.0000	1.0000	1.0000	127
66	1.0000	1.0000	1.0000	133
67	0.0000	0.0000	0.0000	138
68	1.0000	1.0000	1.0000	135
69	1.0000	1.0000	1.0000	139
70	0.9852	1.0000	0.9925	133
71	1.0000	1.0000	1.0000	128
72	0.0000	0.0000	0.0000	122
73	0.0000	0.0000	0.0000	127
accuracy			0.2642	9546
macro avg	0.2568	0.2648	0.2536	9546
weighted avg	0.2590	0.2642	0.2554	9546

Adaboost, Bagging, KNN & NB

Adaboost

	precision	recall	f1-score	support
0	0.0599	0.3904	0.1038	146
1	0.0000	0.0000	0.0000	115
2	0.0000	0.0000	0.0000	127
3	0.0000	0.0000	0.0000	127
4	0.0314	0.1241	0.0501	137
5	0.0183	0.0820	0.0299	122
6	0.0252	0.0690	0.0370	116
7	0.0000	0.0000	0.0000	135
8	0.0162	0.0296	0.0209	135
9	0.0000	0.0000	0.0000	142
10	0.0558	0.0909	0.0692	132
11	0.1250	0.0183	0.0320	109
12	0.0429	0.1197	0.0632	117
13	0.0000	0.0000	0.0000	132
14	0.0000	0.0000	0.0000	140
15	0.0335	0.2119	0.0578	118
16	0.0000	0.0000	0.0000	128
17	0.0000	0.0000	0.0000	115
18	0.0000	0.0000	0.0000	123
19	0.0316	0.0703	0.0436	128
20	0.0000	0.0000	0.0000	124
21	0.0000	0.0000	0.0000	117
22	0.2500	0.0204	0.0377	147
23	0.0000	0.0000	0.0000	129
24	0.0000	0.0000	0.0000	139
25	0.0000	0.0000	0.0000	135
26	0.7564	0.4646	0.5756	127
27	0.0000	0.0000	0.0000	117
28	0.0000	0.0000	0.0000	122
29	0.0000	0.0000	0.0000	132
30	0.0000	0.0000	0.0000	121
31	0.1228	0.0483	0.0693	145
32	0.0000	0.0000	0.0000	129
33	0.1064	0.1538	0.1258	130
34	0.0000	0.0000	0.0000	114
35	0.0328	0.0156	0.0212	128
36	0.0364	0.1545	0.0589	123
37	0.0000	0.0000	0.0000	134
38	0.3333	0.6327	0.4366	147
39	0.0000	0.0000	0.0000	134
40	0.0000	0.0000	0.0000	122
41	0.1802	0.4918	0.2637	122
42	0.0000	0.0000	0.0000	130
43	0.0000	0.0000	0.0000	127
44	0.2137	0.8151	0.3386	119
45	0.1612	0.6781	0.2605	146
46	0.0000	0.0000	0.0000	132
47	0.0000	0.0000	0.0000	124
48	0.0000	0.0000	0.0000	119
49	0.1224	0.1184	0.1204	152
50	0.0000	0.0000	0.0000	117
51	0.0000	0.0000	0.0000	124
52	0.8200	0.6508	0.7257	126
53	0.0000	0.0000	0.0000	141
54	0.0000	0.0000	0.0000	122
55	0.0000	0.0000	0.0000	131
56	0.1121	0.2754	0.1593	138
57	0.0000	0.0000	0.0000	149
58	0.0000	0.0000	0.0000	143
59	0.0000	0.0000	0.0000	114
60	0.3771	1.0000	0.5477	132
61	0.0000	0.0000	0.0000	119
62	0.0852	0.4552	0.1435	134
63	0.3647	0.7578	0.4924	128
64	0.0000	0.0000	0.0000	115
65	0.0000	0.0000	0.0000	127
66	0.0000	0.0000	0.0000	133
67	0.0446	0.1087	0.0633	138
68	0.0000	0.0000	0.0000	135
69	0.7596	1.0000	0.8634	139
70	0.0000	0.0000	0.0000	133
71	0.0000	0.0000	0.0000	128
72	0.0000	0.0000	0.0000	122
73	0.0000	0.0000	0.0000	127
accuracy			0.1256	9546
macro avg	0.0719	0.1223	0.0785	9546
weighted avg	0.0738	0.1256	0.0807	9546

Bagging

	precision	recall	f1-score	support
0	0.8533	0.4384	0.5792	146
1	0.9746	1.0000	0.9871	115
2	0.9845	1.0000	0.9922	127
3	1.0000	1.0000	1.0000	127
4	0.8741	0.9124	0.8929	137
5	0.9528	0.9918	0.9719	122
6	0.9667	1.0000	0.9831	116
7	0.9926	1.0000	0.9963	135
8	0.9643	1.0000	0.9818	135
9	0.9861	1.0000	0.9930	142
10	0.9851	1.0000	0.9925	132
11	0.8898	0.9633	0.9251	109
12	0.9478	0.9316	0.9397	117
13	0.9925	1.0000	0.9962	132
14	0.9929	1.0000	0.9964	140
15	0.9833	1.0000	0.9916	118
16	0.9846	1.0000	0.9922	128
17	0.9910	0.9565	0.9735	115
18	0.9606	0.9919	0.9760	123
19	0.9846	1.0000	0.9922	128
20	0.9688	1.0000	0.9841	124
21	0.9915	1.0000	0.9957	117
22	0.9866	1.0000	0.9932	147
23	0.9453	0.9380	0.9416	129
24	0.9929	1.0000	0.9964	139
25	0.9571	0.9926	0.9745	135
26	1.0000	1.0000	1.0000	127
27	0.9667	0.9915	0.9789	117
28	0.9835	0.9754	0.9794	122
29	1.0000	1.0000	1.0000	132
30	1.0000	1.0000	1.0000	121
31	1.0000	1.0000	1.0000	145
32	0.9923	1.0000	0.9961	129
33	0.9848	1.0000	0.9924	130
34	0.9820	0.9561	0.9689	114
35	0.9846	1.0000	0.9922	128
36	0.9919	1.0000	0.9960	123
37	1.0000	1.0000	1.0000	134
38	1.0000	1.0000	1.0000	147
39	1.0000	1.0000	1.0000	134
40	0.9919	1.0000	0.9959	122
41	1.0000	1.0000	1.0000	122
42	0.9848	1.0000	0.9924	130
43	1.0000	1.0000	1.0000	127
44	1.0000	1.0000	1.0000	119
45	0.9732	0.9932	0.9831	146
46	0.9925	1.0000	0.9962	132
47	1.0000	1.0000	1.0000	124
48	1.0000	1.0000	1.0000	119
49	0.9806	1.0000	0.9902	152
50	1.0000	1.0000	1.0000	117
51	1.0000	1.0000	1.0000	124
52	1.0000	1.0000	1.0000	126
53	1.0000	1.0000	1.0000	141
54	0.9919	1.0000	0.9959	122
55	0.9924	1.0000	0.9962	131
56	0.9640	0.9710	0.9675	138
57	0.9868	1.0000	0.9933	149
58	1.0000	1.0000	1.0000	143
59	0.9744	1.0000	0.9870	114
60	0.9925	1.0000	0.9962	132
61	1.0000	1.0000	1.0000	119
62	1.0000	1.0000	1.0000	134
63	1.0000	1.0000	1.0000	128
64	0.9914	1.0000	0.9957	115
65	1.0000	1.0000	1.0000	127
66	1.0000	1.0000	1.0000	133
67	0.9718	1.0000	0.9857	138
68	1.0000	1.0000	1.0000	135
69	1.0000	1.0000	1.0000	139
70	0.9779	1.0000	0.9888	133
71	1.0000	1.0000	1.0000	128
72	0.9533	0.8361	0.8908	122
73	0.9839	0.9606	0.9721	127
accuracy			0.9831	9546
macro avg	0.9823	0.9838	0.9820	9546
weighted avg	0.9823	0.9831	0.9816	9546

KNN

	precision	recall	f1-score	support
0	0.6364	0.1438	0.2346	146
1	0.8582	1.0000	0.9237	115
2	0.6496	0.5984	0.6230	127
3	0.8194	1.0000	0.9007	127
4	0.3111	0.1022	0.1538	137
5	0.5000	0.3361	0.4020	122
6	0.4754	0.5000	0.4874	116
7	0.7895	1.0000	0.8824	135
8	0.6357	0.6074	0.6212	135
9	0.7634	1.0000	0.8659	142
10	0.7752	0.7576	0.7663	132
11	0.3810	0.1468	0.2119	109
12	0.5581	0.2051	0.3000	117
13	0.8199	1.0000	0.9010	132
14	0.8861	1.0000	0.9396	140
15	0.8429	1.0000	0.9147	118
16	0.8828	1.0000	0.9377	128
17	0.8116	0.4870	0.6087	115
18	0.4646	0.3740	0.4144	123
19	0.7971	0.8594	0.8271	128
20	0.8611	1.0000	0.9254	124
21	0.7355	0.9744	0.8382	117
22	0.6542	0.4762	0.5512	147
23	0.5000	0.2016	0.2873	129
24	0.8968	1.0000	0.9456	139
25	0.7342	0.8593	0.7918	135
26	0.9845	1.0000	0.9922	127
27	0.6146	0.5043	0.5540	117
28	0.7303	0.9098	0.8102	122
29	0.9851	1.0000	0.9925	132
30	0.9237	1.0000	0.9603	121
31	0.8683	1.0000	0.9295	145
32	0.9923	1.0000	0.9961	129
33	0.9091	1.0000	0.9524	130
34	0.5766	0.5614	0.5689	114
35	0.8067	0.9453	0.8705	128
36	0.7961	0.9837	0.8800	123
37	0.8874	1.0000	0.9404	134
38	0.9735	1.0000	0.9866	147
39	0.9116	1.0000	0.9537	134
40	0.8133	1.0000	0.8971	122
41	0.9839	1.0000	0.9919	122
42	0.9155	1.0000	0.9559	130
43	0.9137	1.0000	0.9549	127
44	0.9835	1.0000	0.9917	119
45	0.5714	0.5753	0.5734	146
46	0.9103	1.0000	0.9531	132
47	0.9920	1.0000	0.9960	124
48	0.9597	1.0000	0.9794	119
49	0.8994	1.0000	0.9470	152
50	0.9915	1.0000	0.9957	117
51	0.9466	1.0000	0.9725	124
52	0.9474	1.0000	0.9730	126
53	0.9792	1.0000	0.9895	141
54	0.9919	1.0000	0.9959	122
55	0.9097	1.0000	0.9527	131
56	0.5323	0.4783	0.5038	138
57	0.9085	1.0000	0.9521	149
58	0.9931	1.0000	0.9965	143
59	0.8906	1.0000	0.9421	114
60	0.9635	1.0000	0.9814	132
61	1.0000	1.0000	1.0000	119
62	0.9241	1.0000	0.9606	134
63	0.9697	1.0000	0.9846	128
64	1.0000	1.0000	1.0000	115
65	0.9621	1.0000	0.9807	127
66	0.9568	1.0000	0.9779	133
67	0.6471	0.7174	0.6804	138
68	0.9926	1.0000	0.9963	135
69	0.9720	1.0000	0.9858	139
70	0.9779	1.0000	0.9888	133
71	0.9922	1.0000	0.9961	128
72	0.5926	0.3934	0.4729	122
73	0.4948	0.3780	0.4286	127
accuracy			0.8409	9546
macro avg	0.8173	0.8389	0.8188	9546
weighted avg	0.8185	0.8409	0.8204	9546

NB

	precision	recall	f1-score	support
0	0.0000	0.0000	0.0000	146
1	0.4038	0.1826	0.2515	115
2	0.1081	0.0315	0.0488	127
3	1.0000	0.1102	0.1986	127
4	0.1667	0.0073	0.0140	137
5	0.0952	0.0820	0.0881	122
6	0.3333	0.0086	0.0168	116
7	0.3297	0.2222	0.2655	135
8	0.1176	0.0296	0.0473	135
9	0.0555	0.2887	0.0931	142
10	0.3171	0.0985	0.1503	132
11	0.0417	0.0092	0.0150	109
12	0.6000	0.0256	0.0492	117
13	0.7778	0.1061	0.1867	132
14	0.5455	0.1286	0.2081	140
15	0.0312	0.0085	0.0133	118
16	0.5556	0.1172	0.1935	128
17	0.0667	0.4348	0.1156	115
18	0.0000	0.0000	0.0000	123
19	0.1628	0.0547	0.0819	128
20	0.3095	0.1048	0.1566	124
21	0.0000	0.0000	0.0000	117
22	0.2653	0.0884	0.1327	147
23	0.0000	0.0000	0.0000	129
24	0.0000	0.0000	0.0000	139
25	1.0000	0.0148	0.0292	135
26	0.3864	0.8031	0.5217	127
27	0.1818	0.0171	0.0313	117
28	0.0000	0.0000	0.0000	122
29	0.5077	1.0000	0.6735	132
30	0.6522	0.1240	0.2083	121
31	0.7209	0.2138	0.3298	145
32	0.6500	0.3023	0.4127	129
33	0.3836	0.2154	0.2759	130
34	0.0000	0.0000	0.0000	114
35	0.6250	0.0391	0.0735	128
36	1.0000	0.0163	0.0320	123
37	0.0000	0.0000	0.0000	134
38	0.5167	0.6327	0.5688	147
39	0.6912	0.3507	0.4653	134
40	0.6000	0.0738	0.1314	122
41	0.1658	0.8033	0.2749	122
42	0.9600	0.1846	0.3097	130
43	0.0000	0.0000	0.0000	127
44	0.1822	0.6555	0.2852	119
45	0.1579	0.0411	0.0652	146
46	1.0000	0.1970	0.3291	132
47	0.2418	0.2984	0.2671	124
48	0.8571	0.1008	0.1805	119
49	0.3750	0.1184	0.1800	152
50	0.3535	1.0000	0.5223	117
51	1.0000	0.1290	0.2286	124
52	1.0000	0.6508	0.7885	126
53	0.2889	1.0000	0.4483	141
54	0.1920	0.7459	0.3054	122
55	0.6389	0.1756	0.2754	131
56	0.0882	0.0435	0.0583	138
57	0.4957	0.3893	0.4361	149
58	0.5417	1.0000	0.7027	143
59	0.0000	0.0000	0.0000	114
60	0.3426	0.6515	0.4491	132
61	0.9154	1.0000	0.9558	119
62	0.3333	0.1716	0.2266	134
63	0.1422	0.4844	0.2199	128
64	0.9583	1.0000	0.9787	115
65	0.6865	1.0000	0.8141	127
66	0.8526	1.0000	0.9204	133
67	0.0256	0.0145	0.0185	138
68	0.5315	1.0000	0.6941	135
69	0.7316	1.0000	0.8450	139
70	0.1557	1.0000	0.2695	133
71	0.6667	1.0000	0.8000	128
72	0.6170	0.2377	0.3432	122
73	0.2500	0.3465	0.2904	127
accuracy			0.3176	9546
macro avg	0.4047	0.3160	0.2644	9546
weighted avg	0.4055	0.3176	0.2664	9546

Comparative Model Performance

	model	Pred_Accuracy	descriptions
1	LSTM_Model_data_3_resamp1ed	0.974963	LSTM Model + Word2Vec Embedding on data_3_resa...
1	LSTM_Model_data_3_resamp1ed_Glove	0.972449	LSTM Model + Glove Embedding on data_3_resampl...
1	LSTM_Model_data_3_resamp1ed_Glove_GRU	0.972449	GRU Model + Glove Embedding on data_3_resample...
1	RNN_Model_data_3_resamp1ed_Glove	0.425087	RNN Model + Glove Embedding on data_3_resample...

	Model	accuracy
1	Random Forest	0.983658
1	Random Forest Classifier - Weighted	0.984706
1	Adaboost Classifier	0.125602
1	Bagging Classifier	0.983134
2	MultinomialNB Classifier	0.317620
2	ComplementNB Classifier	0.136706
1	KNN Classifier	0.840876

1.4

What Are Next Steps?

Potential Model Performance Improvements

- We have focused on various aspects of the NLP process starting from studying the data to selecting all possible or feasible models and then processing the features and training the models. The initial results have been documented and presented in this interim report. As a next step between now and the final submission of reports and findings, we plan to further explore some of the areas where we want to go deeper to see a potential benefit:
  - **Imbalanced Data:** We have already explored resampling and SMOTE options, however we would like to get deeper in these two as well as other potential sampling and data augmentation methods.
  - **BERT:** We have worked on BERT for multi-label classification and the model is showing its initial results. As a next step we want to fine tune this model. Running these models on the dataset has been a time consuming activity for the team. This model being new with multiple possible implementation scenarios and libraries, is research intensive topic for us. We would like to spend some time on it subsequently.
  - **Fine Tuning DL and ML Model Based on Hyper Parameters:** We have played with the hyper parameters initially while building them. There are certain models which have yielded results beyond our expectations both on training and testing data. However some to the models have not performed as we hoped for. We would like to explore the possibility of further tuning them.
  - **Dimensionality Reduction:** We have done extensive analysis of the text data in the tickets. We have used topic modelling from Gensim to categorize and cluster the words that appear together most often and hence indicates substitutability of these words by the corresponding topics. This will reduce the number of dimensions and the vector space for computation. We would like to explore this aspect as a next step and document our findings with respect to its impact on the model performance.
  - **Language Translation and Spelling Checks**: We have peripherally explore this aspect. We have not find this as a critical aspect of the classification process as some models are yielding good results. However we would like to still explore this perspective in whatever limited way possible. On the same lines we want improve the effectiveness of the cleaning function using regular expression.

thank you!

RADHIKA KULKARNI[Radhikakulkarni844@gmail.com]
SYED TALIBUDDIN SAIFI [letters.syed@gmail.com]
TANVEER QURAISHI[tanveer.quraishi@gmail.com]
SHANMUGAM SAMPATH [shanmugam.Sampath@gmail.com]
KANWAL RAI [raikanwalrai@gmail.com]