Capstone Project Final Report - AIML Online Batch- 2021-22

Automatic Ticket Assignment

Prepared By:

1. Jagriti Srivastava
2. Rishi Kumar
3. Sree Bhargava
4. Sree Ashrit Dande
5. Shyamlapriya Pudur Raamamurthy

Supervised by:
Abdul Manaf

1. Summary of problem statement, data and findings

1.1. Understanding the Business

In any IT industry, Incident Management plays an important role in delivering quality support to customers. An incident ticket is created by various groups of people within the organization to resolve an issue as quickly as possible based on its severity. Whenever an incident is created, it reaches the Service desk team and then it gets assigned to the respective teams to work on the incident.

The Service Desk team (L1/L2) will perform basic analysis on the user's requirement, identify the issue based on given descriptions and assign it to the respective teams.
The manual assignment of these incidents might have below disadvantages:

- More resource usage and expenses.
- Human errors - Incidents get assigned to the wrong assignment groups
- Delay in assigning the tickets
- More resolution times
- If a particular ticket takes more time in analysis, other productive tasks get affected for the Service Desk

If this ticket assignment is automated, it can be more cost-effective, less resolution time and the Service Desk team can focus on other productive tasks.

1.2. Objective

From the given problem description, we could see that the existing system is able to assign 75% of the tickets correctly.So our objective here is to build an AI-based classifier model to assign the tickets to right functional groups by analyzing the given description.

1.3. Observations from the given Dataset

- Four columns – Short Description, Description, Caller and Assignment group
- 74 Assignment groups found - Target classes
- Caller names are random (may not be useful for training data)
- Non-English language also found in the data
- Email/chat format in description
- Symbols & other characters in the description
- Hyperlinks, URLS & few image data found in the description
- Blanks found either in the short description or description field
- Few descriptions same as the short description

- Few words were combined together
- Spelling mistakes and typo errors are found

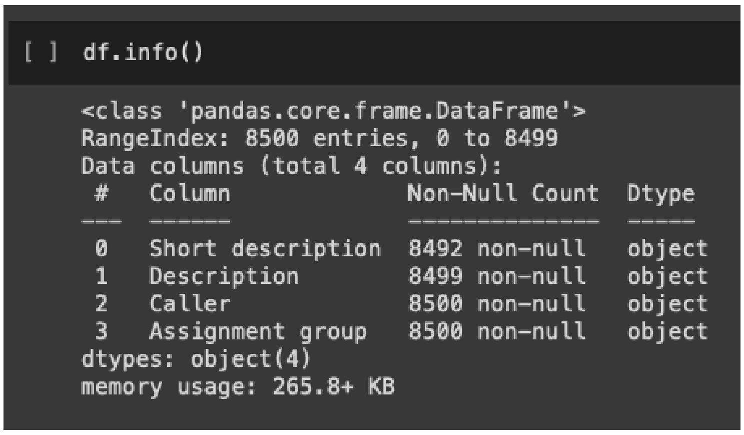
2. Overview of the final process

2.1. Observations from Target Class

- The Target class distribution is extremely skewed
- A large no of entries for GRP_0 (mounting to 3976) which account for ~50% of the data
- There are groups with 1 entry also. We could merge all groups with small entries to a group to reduce the imbalance in the target. This may reduce the imbalance to some extent.

2.2. Data Pre-processing

Below steps have been performed for initial pre-processing and clean up of data.

- Dropped the caller field as the data was not found to be useful for analysis
- Replaced Null values in Short description & description with space.
    - There were 8 null values in the short description field:
    - 

```
[ ]  df.info()

     <class 'pandas.core.frame.DataFrame'>
     RangeIndex: 8500 entries, 0 to 8499
     Data columns (total 4 columns):
      #   Column             Non-Null Count  Dtype
     ---  ------             --------------  -----
      0   Short description  8492 non-null   object
      1   Description        8499 non-null   object
      2   Caller             8500 non-null   object
      3   Assignment group   8500 non-null   object
     dtypes: object(4)
     memory usage: 265.8+ KB
```

- Merged Short Description & Description fields for analysis
- Contraction words found in the merged Description are removed for ease of word modeling
- Changed the case sensitivity of words to the common one
- Removed Hashtags and kept the words, Hyperlinks, URLs, HTML tags & non-ASCII symbols from merged fields.
- Translating all languages (German) to English
- Tokenization of merged data
- Removal of Stop words
- Lemmatization
- WordCloud created for all available 50 groups to have more information specific to Assignment groups
- Attempted to do spell check

- Created Plot to understand the distribution of words

2.3. Algorithms used

We have tried below pre-modelling and modeling techniques

- Built LDA model to understand the data by creating Topic-based clustering
- Word2Vector Embedding
- Glove Embedding
- Bidirectional LSTM Models using both embedding and compared
- GRU model
- RNN model
- Random Forest classifier
- SVM Classifier model

We used the following approaches for modeling:

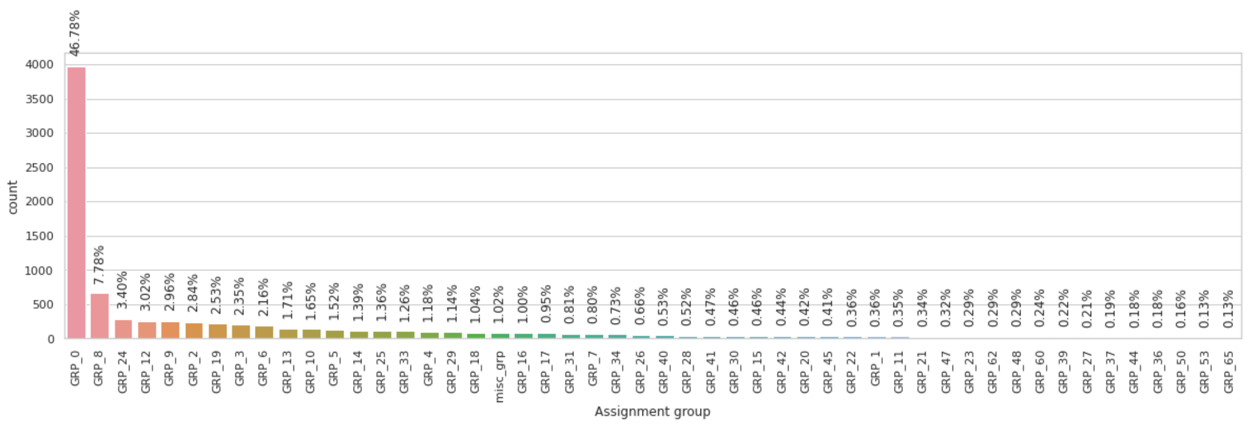
- Raw text was cleaned to remove unnecessary elements which would not have been useful for modeling exercises
- Performed the pre-modeling and modeling steps on this cleaned dataset
- Resampled the cleaned dataset to achieve balance between the incident groups
- We used 2 separate models. Here one model was used to classify the GRP0 & a second model was used to classify the other groups. The dataset from the 1st model contains GRP0 data & all the remaining data combined to a single group. The dataset for the 2nd model contained all groups other than GRP0. The dataset here was resampled again to address the target imbalance if any.

3. Step-by-step walk through the solution

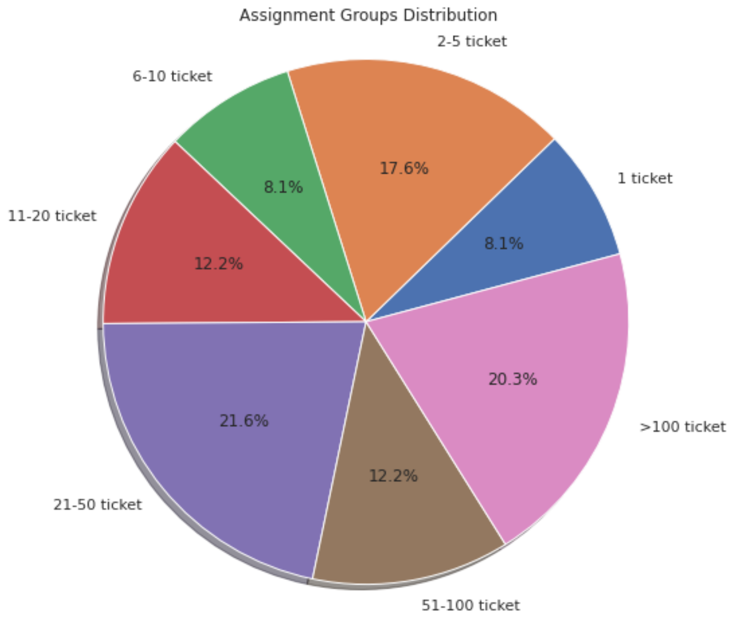
3.1. Data Pre-processing

- The target class is extremely skewed data. The target class was filtered for less than 10 entries and grouped together as misc_grp as there is no information with the groups individually.

Distribution of Target class for all groups (after adding misc_grp)



- There are many groups where the number of tickets assigned is less than 100. The following graph shows the distribution of groups based on number of tickets assigned



3.2. Data cleaning

- A function clean_text() has been created to clean up the unwanted information from initial observations. All words have been converted to lowercase. The email headers and sender information are removed. All the numbers, non-dictionary characters, newline characters, hashtag, HTML entities, hyperlinks, extra spaces and unreadable characters have been added to the function. Ensured to remove any caller names included in the description column. The clean_data function is applied to the Description column and cleaned up data is generated for further analysis.

3.3. Translation

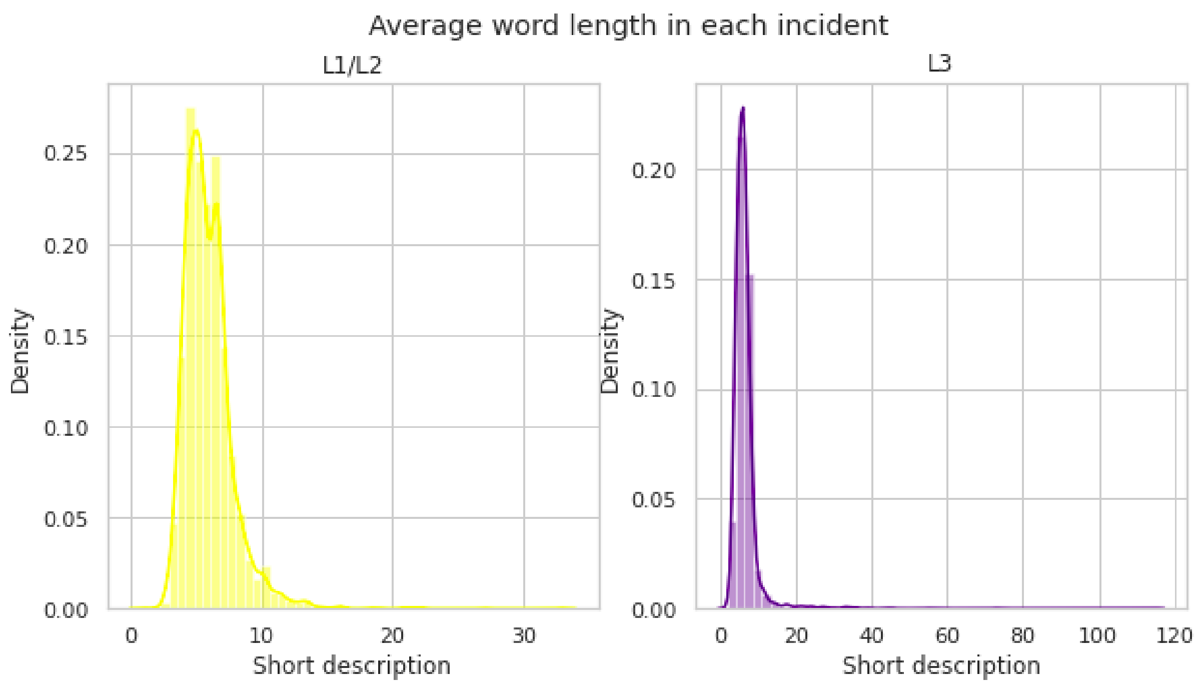
- The German language is found from the dataset. Attempted using many libraries such as googletrans, textblob, goslate, etc for translation of non-english entries to english, but found that all of them had size limitations & was unable to proceed with translation. To overcome this limitation, a wordlist of non-english words was formed from the dataset. All the rows from the Description column filtered using German wordlist have been translated to English language by passing to a Google translator.

3.4. Lemmatization & Stop words removal

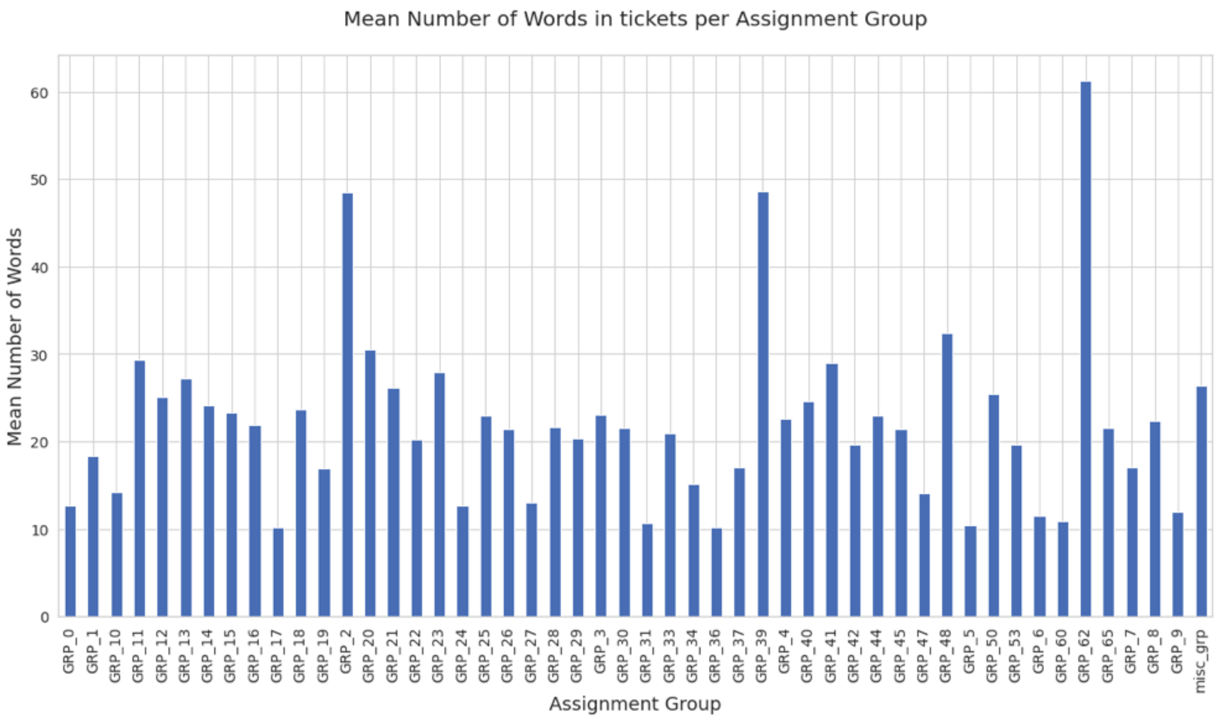
- Stop words have been removed using nltk corpus modules. Lemmatization is the process of grouping together the different inflected forms of a word so they can be analyzed as a single item. Lemmatization is similar to stemming but it brings context to the words. So, it links words with similar meanings to one word.
  Here we have preferred Lemmatization over Stemming because lemmatization does morphological analysis of the words.

3.5. Average Word Length

- Average length of the word of the short description is determined for different categories of incidents
- This metric is distributed to form a density plot

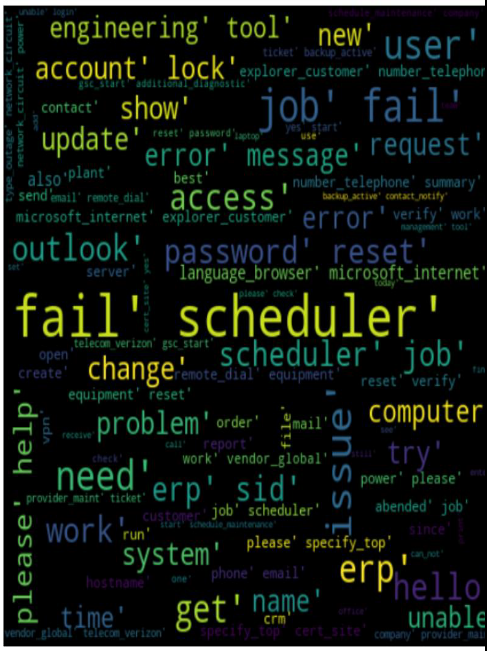


3.6. Mean Number of words in tickets per assignment group

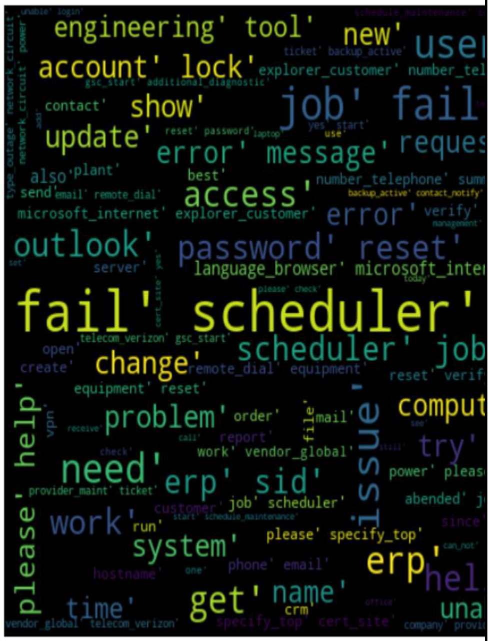


3.7. Analysis using WordCloud

- WordCloud is a data visualization technique used for representing text data in which the size of each word indicates its frequency or importance. Significant textual data points can be highlighted using a word cloud. WordClouds have been generated with All available words & top 100 words.



Word clouds for all words



Word cloud for top 100 words

4. Model Evaluation - Deciding Models and Model Building

4.1. Word Embedding

As all our Machine Learning and Deep learning algorithms are incapable of processing strings or plain text in their raw form, word embeddings are used to convert the texts into numbers. There may be different numerical representations of the same text. It tries to map a word using a dictionary to a vector.
We have experimented below 2 types of embedding in our models with the dimension as 100.

4.1.1. Word2Vector Embedding:

Word2Vec models are shallow, two-layer neural networks that are trained to reconstruct linguistic contexts of words. Word2vec takes as its input a large corpus of text and produces a vector space, typically of several hundred dimensions, with each unique word in the corpus being assigned a corresponding vector in the space.

4.1.2. GloVe (Global Vectors) Embedding:

GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space.

4.2. Bi-directional LSTM Model

Bidirectional LSTMs are an extension of traditional LSTMs that can improve model performance on classification problems.
In problems where all timesteps of the input sequence are available, Bidirectional LSTMs train two instead of one LSTMs on the input sequence. The first on the input sequence as-is and the second on a reversed copy of the input sequence. This can provide additional context to the network and result in faster and even fuller learning on the problem.

4.3. GRU Model

GRU (Gated Recurrent Unit) aims to solve the vanishing gradient problem which comes with a standard recurrent neural network. GRU can also be considered as a variation on the LSTM because both are designed similarly and, in some cases, produce equally excellent results.
To solve the vanishing gradient problem of a standard RNN, GRU's got rid of the cell state and used the hidden state to transfer information. It has only 2 gates, an update gate and a reset gate. Basically, these are two vectors which decide what information should be passed to the output. The special thing about them is that they can be trained to keep information from long ago, without washing it through time or removing information which is irrelevant to the prediction.

4.4. RNN Model

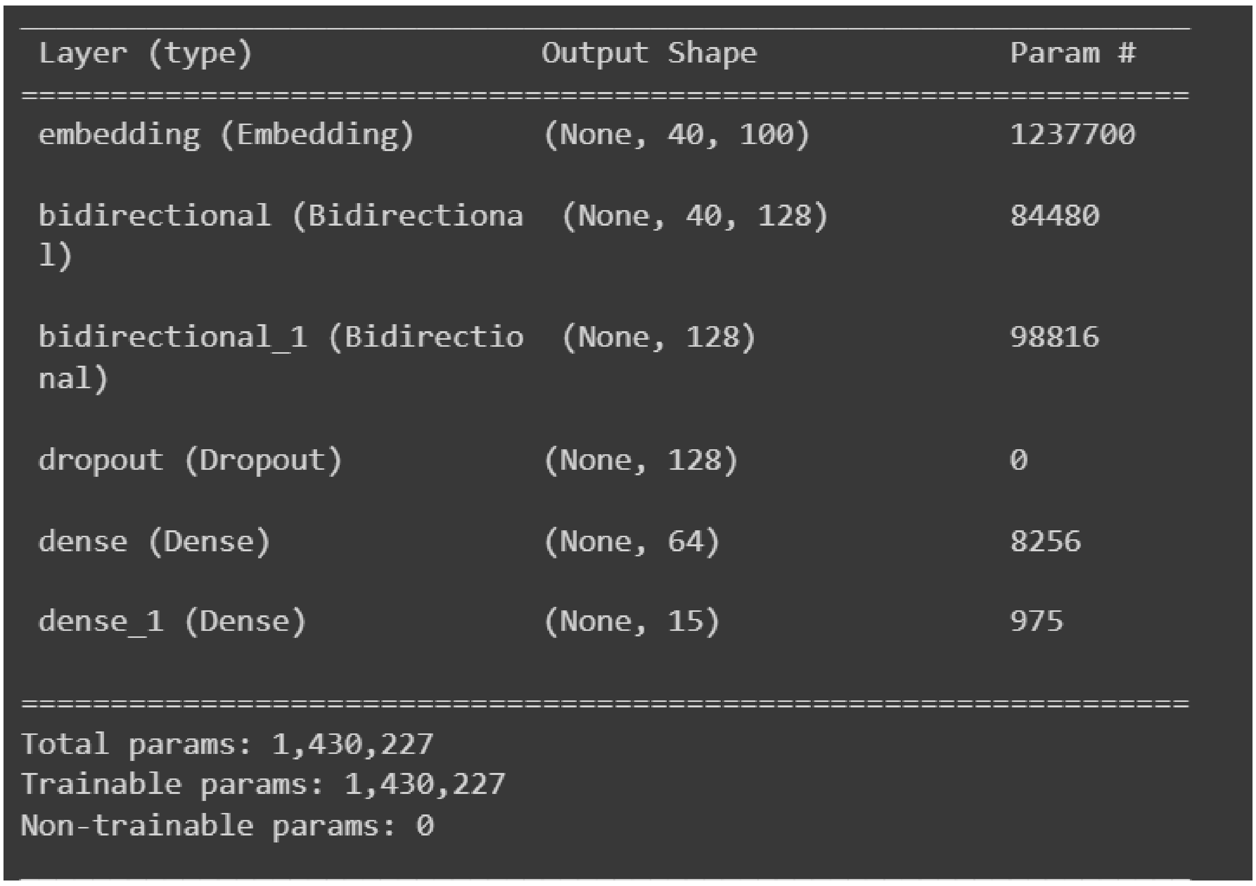
Recurrent Neural Networks (RNNs) are a family of neural networks designed specifically for sequential data processing. The RNN model will predict the next word in a sequence based on the previous ones. The same operation is performed recurrently which is why it is called Recurrent Neural Networks.
RNNs perform the same task for every element of a sequence, with the output being dependent on the previous computations. Another way to think about RNNs is that they have a "memory" which captures information about what has been calculated so far.

5. Model Performance and Results:

5.1. Model building approach 1 : Raw data

5.1.1. Bidirectional LSTM + Word2Vec:

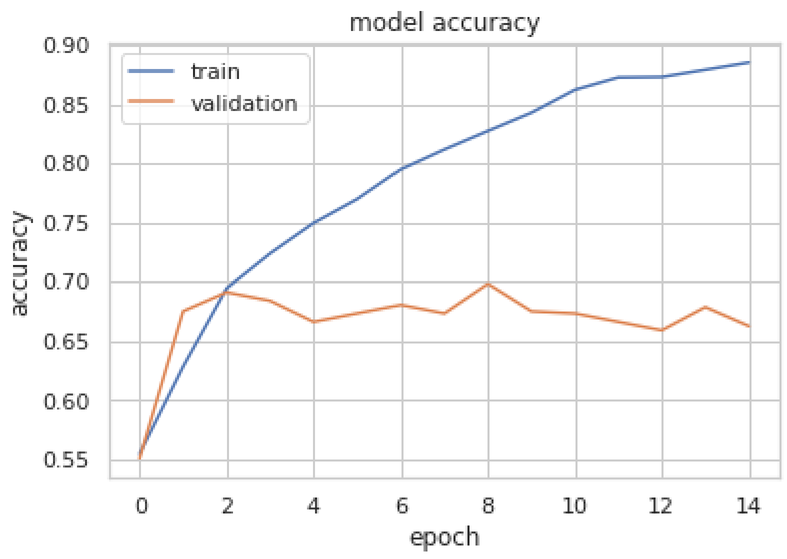
The model summary is shown below:



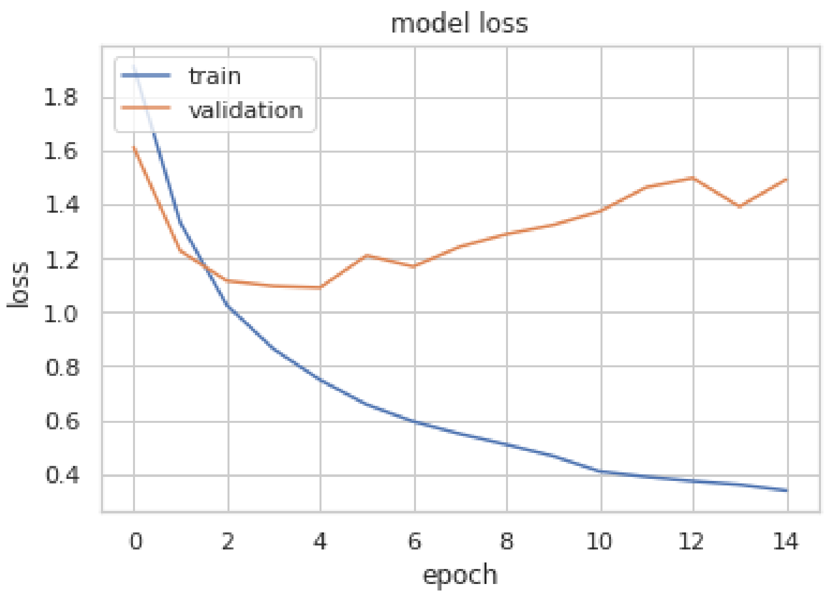
```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 40, 100)           1237700

 bidirectional (Bidirectiona  (None, 40, 128)          84480
 l)

 bidirectional_1 (Bidirectio  (None, 128)              98816
 nal)

 dropout (Dropout)           (None, 128)               0

 dense (Dense)               (None, 64)                8256

 dense_1 (Dense)             (None, 15)                975

=================================================================
Total params: 1,430,227
Trainable params: 1,430,227
Non-trainable params: 0
_________________________________________________________________
```

LSTM Model Observations:

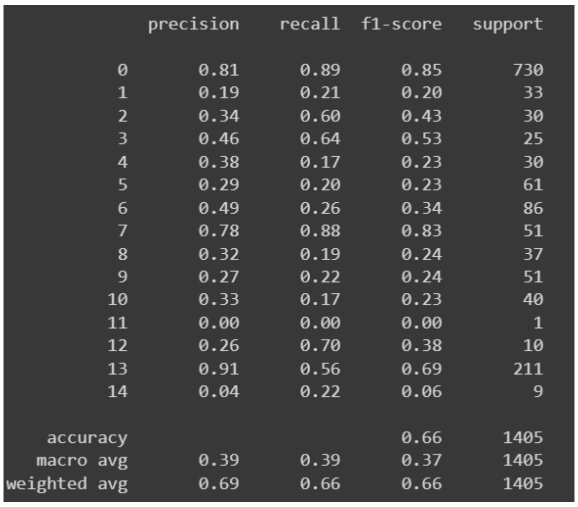
Unsampled Data - Model Accuracy



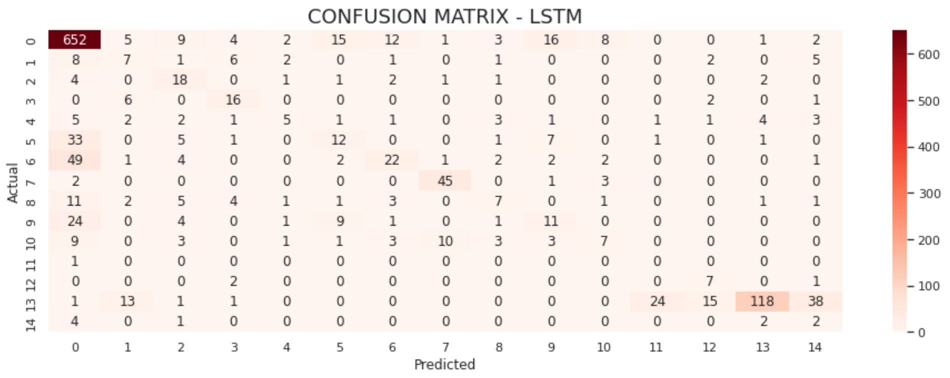
Unsampled Data - Model Loss



Classification - report and Confusion Matrix



```
              precision    recall  f1-score   support

           0       0.81      0.89      0.85       730
           1       0.19      0.21      0.20        33
           2       0.34      0.60      0.43        30
           3       0.46      0.64      0.53        25
           4       0.38      0.17      0.23        30
           5       0.29      0.20      0.23        61
           6       0.49      0.26      0.34        86
           7       0.78      0.88      0.83        51
           8       0.32      0.19      0.24        37
           9       0.27      0.22      0.24        51
          10       0.33      0.17      0.23        40
          11       0.00      0.00      0.00         1
          12       0.26      0.70      0.38        10
          13       0.91      0.56      0.69       211
          14       0.04      0.22      0.06         9

    accuracy                           0.66      1405
   macro avg       0.39      0.39      0.37      1405
weighted avg       0.69      0.66      0.66      1405
```



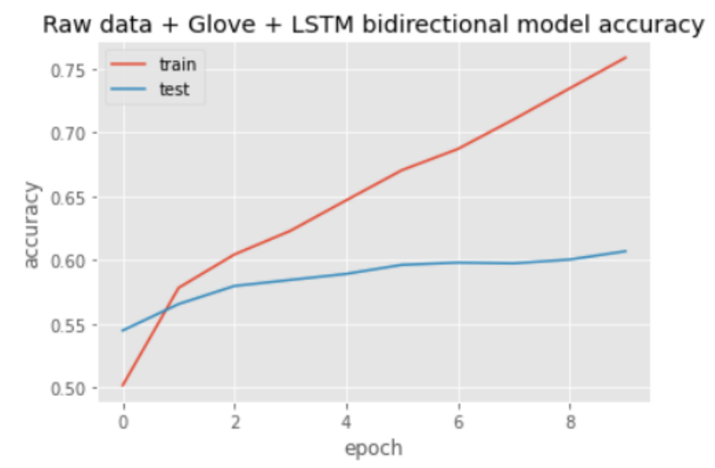
Actual \ Predicted	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	652	5	9	4	2	15	12	1	3	16	8	0	0	1	2
1	8	7	1	6	2	0	1	0	1	0	0	0	2	0	5
2	4	0	18	0	1	1	2	1	1	0	0	0	0	2	0
3	0	6	0	16	0	0	0	0	0	0	0	0	2	0	1
4	5	2	2	1	5	1	1	0	3	1	0	1	1	4	3
5	33	0	5	1	0	12	0	0	1	7	0	1	0	1	0
6	49	1	4	0	0	2	22	1	2	2	2	0	0	0	1
7	2	0	0	0	0	0	0	45	0	1	3	0	0	0	0
8	11	2	5	4	1	1	3	0	7	0	1	0	0	1	1
9	24	0	4	0	1	9	1	0	1	11	0	0	0	0	0
10	9	0	3	0	1	1	3	10	3	3	7	0	0	0	0
11	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	2	0	0	0	0	0	0	0	0	7	0	1
13	1	13	1	1	0	0	0	0	0	0	0	24	15	118	38
14	4	0	1	0	0	0	0	0	0	0	0	0	0	2	2

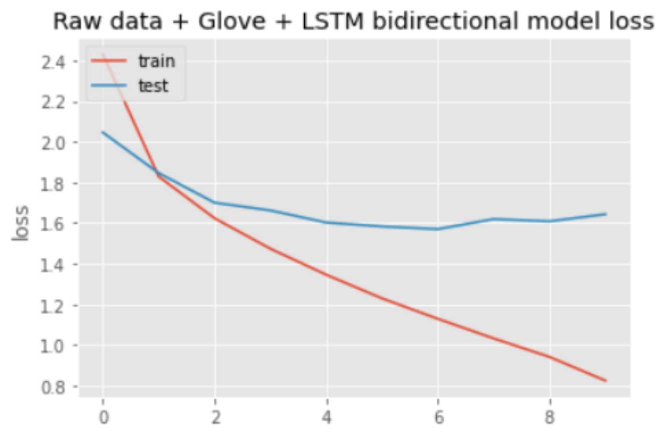
5.1.2. LSTM Bidirectional Model with Glove Embeddings:

Model Summary:

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_4 (InputLayer)        [(None, 300)]             0

 embedding_3 (Embedding)     (None, 300, 100)          900100

 bidirectional_2 (Bidirectio  (None, 256)              234496
 nal)

 dropout_4 (Dropout)         (None, 256)               0

 dense_6 (Dense)             (None, 100)               25700

 dense_7 (Dense)             (None, 49)                4949

=================================================================
Total params: 1,165,245
Trainable params: 1,165,245
Non-trainable params: 0
```

Model Performance:



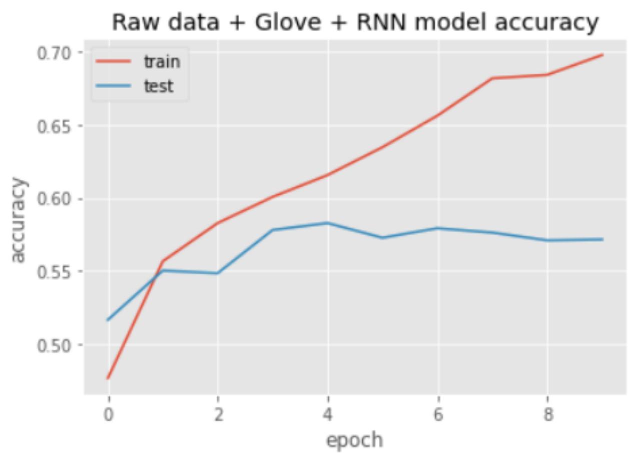


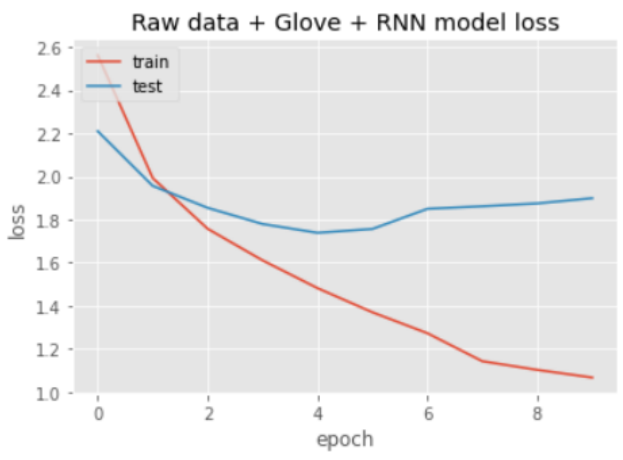
5.1.3. RNN Model with Glove Embeddings:

Model Summary:

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 embedding_2 (Embedding)     (None, 300, 100)          900100    
                                                                 
 conv1d (Conv1D)             (None, 291, 100)          100100    
                                                                 
 max_pooling1d (MaxPooling1D  (None, 145, 100)         0         
 )                                                               
                                                                 
 dropout_2 (Dropout)         (None, 145, 100)          0         
                                                                 
 conv1d_1 (Conv1D)           (None, 136, 100)          100100    
                                                                 
 max_pooling1d_1 (MaxPooling  (None, 68, 100)          0         
 1D)                                                             
                                                                 
 bidirectional_1 (Bidirectio  (None, 256)              234496    
 nal)                                                            
                                                                 
 dropout_3 (Dropout)         (None, 256)               0         
                                                                 
 dense_4 (Dense)             (None, 100)               25700     
                                                                 
 dense_5 (Dense)             (None, 49)                4949      
                                                                 
=================================================================
Total params: 1,365,445
Trainable params: 1,365,445
Non-trainable params: 0
_________________________________________________________________
```

Model Performance:



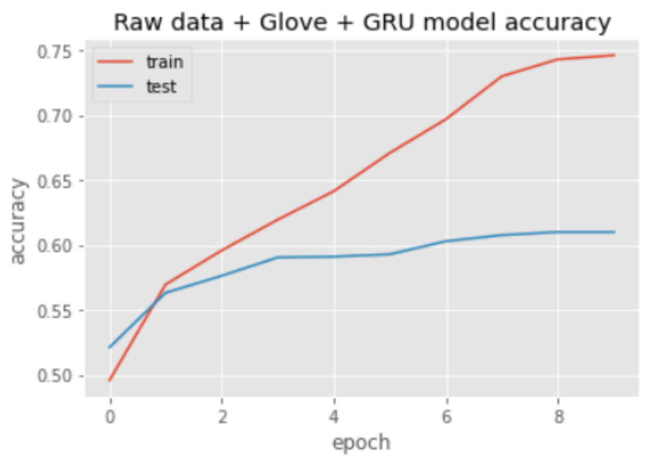


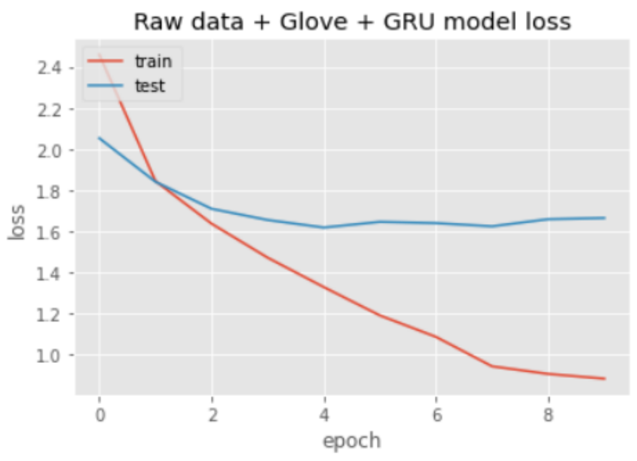
5.1.4. GRU Model with Glove Embeddings:

Model Summary:

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 input_5 (InputLayer)        [(None, 300)]             0         
                                                                 
 embedding_4 (Embedding)     (None, 300, 100)          900100    
                                                                 
 gru_1 (GRU)                 (None, 128)               88320     
                                                                 
 dropout_5 (Dropout)         (None, 128)               0         
                                                                 
 dense_8 (Dense)             (None, 100)               12900     
                                                                 
 dense_9 (Dense)             (None, 49)                4949      
                                                                 
=================================================================
Total params: 1,006,269
Trainable params: 1,006,269
Non-trainable params: 0
_________________________________________________________________
```

Model Performance:

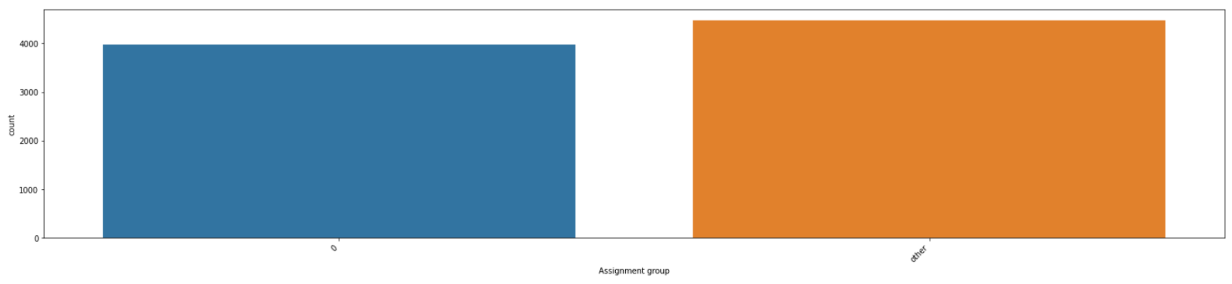




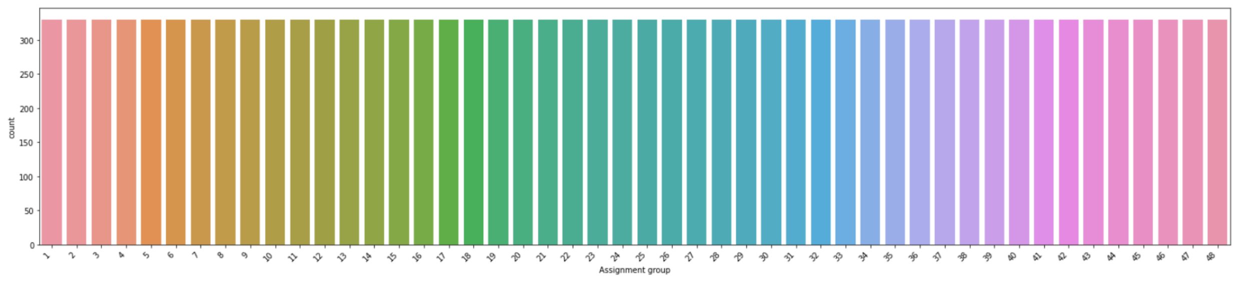
5.2. Model building approach 2 : Resampled data

Since the data for Group-0 is more in numbers compared to other groups, resampling is done in two ways.

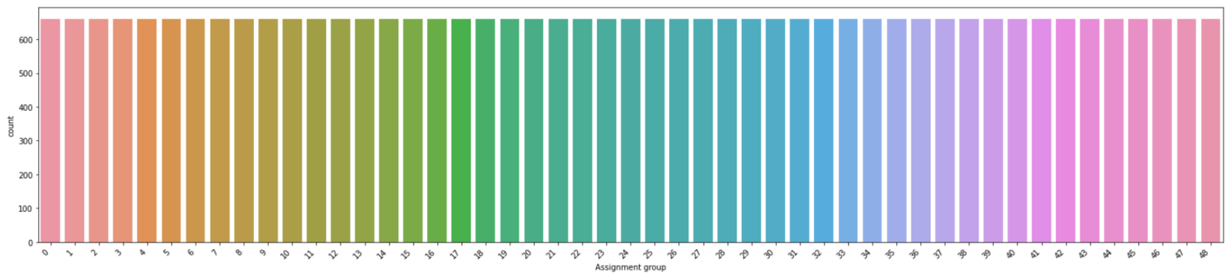
1. Resampling the other group (other than group-0) for a two-model approach.
2. Resampling the whole dataset to simultaneously apply upsampling for other group and downsampling for group 0



```
plot_groups(df_cleaned_others_resampled)
```



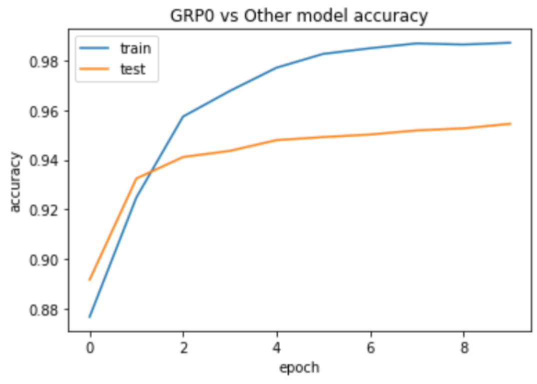
```
plot_groups(df_cleaned_resampled)
```

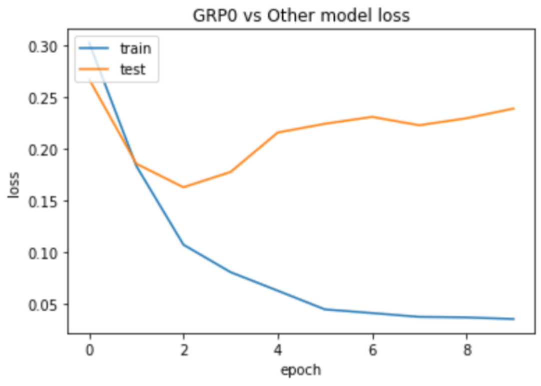


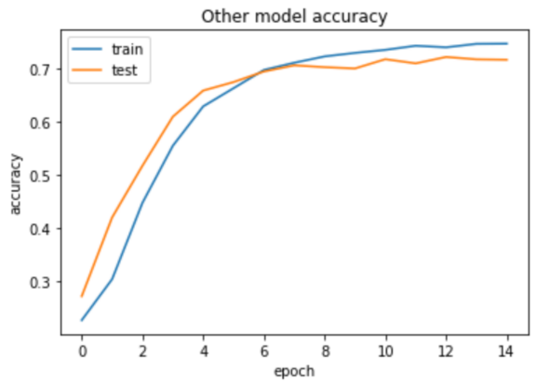
Two Part model:

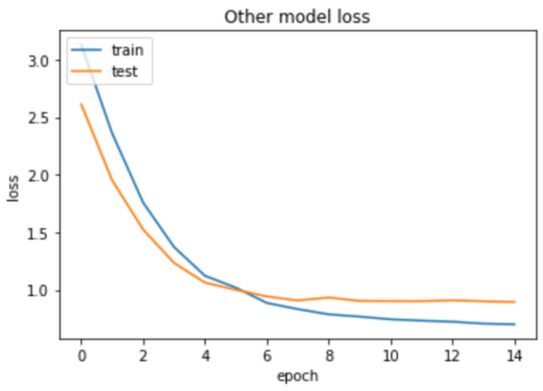
- **Model 1 - Identifies if the target is 'Group 0' or 'Others'**
- **Model 2 - Identifies the actual target from other groups**

5.2.1. LSTM Model with Word2Vec Embeddings:

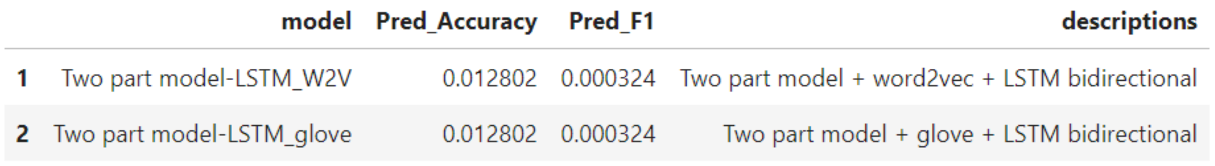








Model performance for Two model Approach using LSTM model:

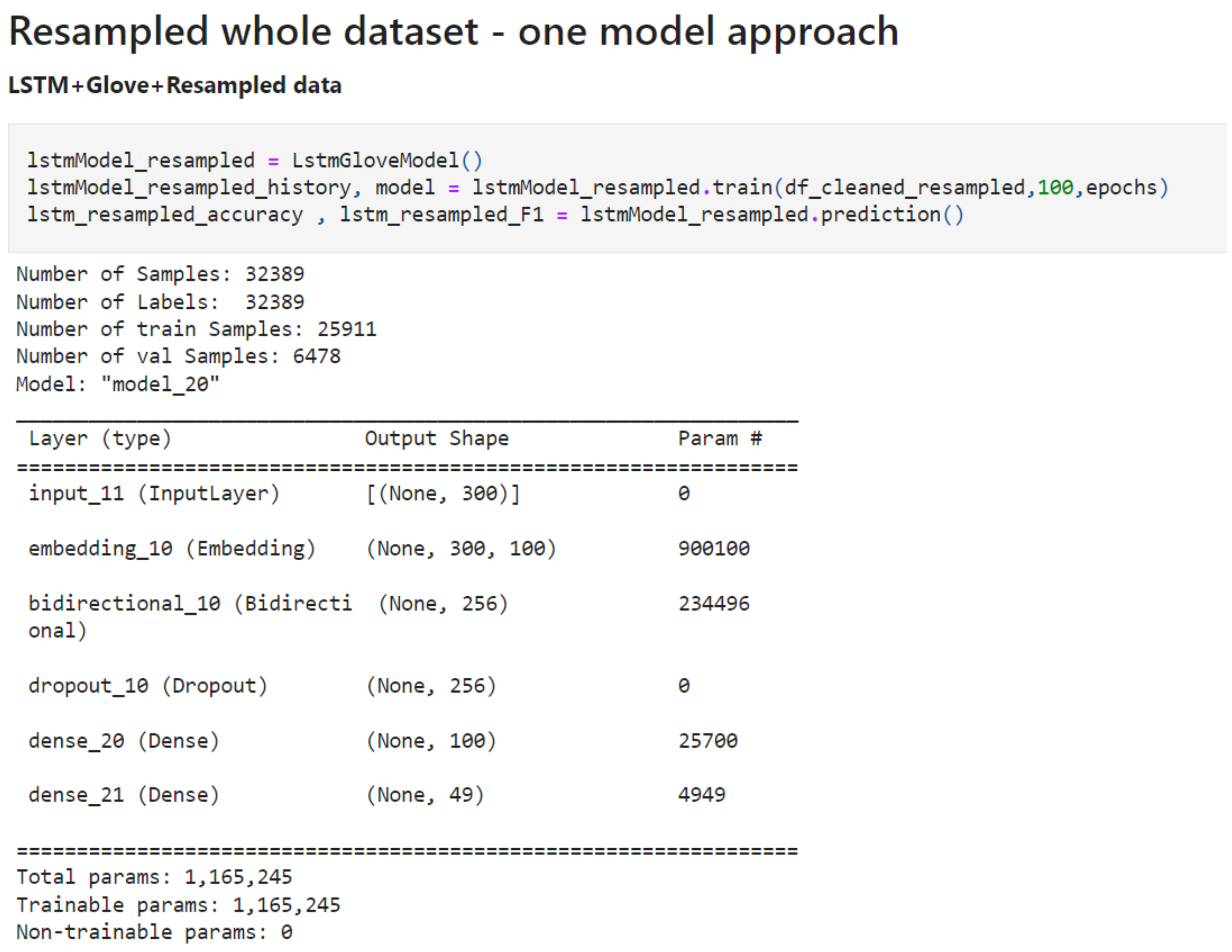


	model	Pred_Accuracy	Pred_F1	descriptions
1	Two part model-LSTM_W2V	0.012802	0.000324	Two part model + word2vec + LSTM bidirectional
2	Two part model-LSTM_glove	0.012802	0.000324	Two part model + glove + LSTM bidirectional

Since, the two-part model for both types of embedding is not giving good results, not proceeding with other model+embedding combinations for this approach.

Resampled data for single model:

Model summary:



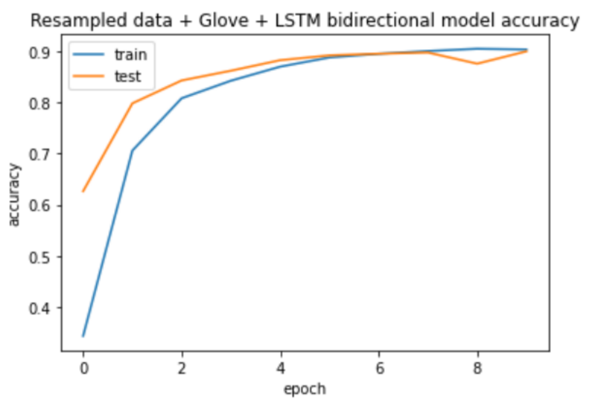
Resampled whole dataset - one model approach

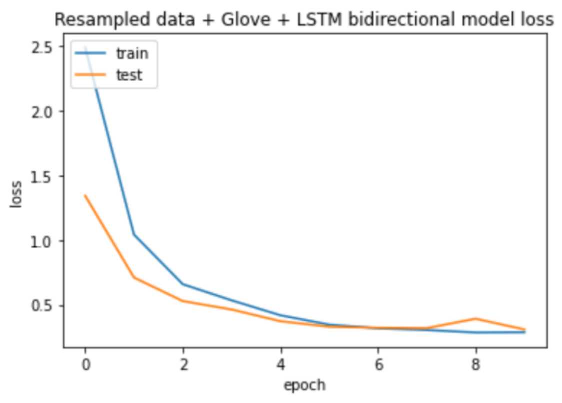
LSTM+Glove+Resampled data

```
lstmModel_resampled = LstmGloveModel()
lstmModel_resampled_history, model = lstmModel_resampled.train(df_cleaned_resampled,100,epochs)
lstm_resampled_accuracy , lstm_resampled_F1 = lstmModel_resampled.prediction()
```

```
Number of Samples: 32389
Number of Labels:  32389
Number of train Samples: 25911
Number of val Samples: 6478
Model: "model_20"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 input_11 (InputLayer)       [(None, 300)]             0         
                                                                 
 embedding_10 (Embedding)    (None, 300, 100)          900100    
                                                                 
 bidirectional_10 (Bidirecti  (None, 256)              234496    
 onal)                                                           
                                                                 
 dropout_10 (Dropout)        (None, 256)               0         
                                                                 
 dense_20 (Dense)            (None, 100)               25700     
                                                                 
 dense_21 (Dense)            (None, 49)                4949      
                                                                 
=================================================================
Total params: 1,165,245
Trainable params: 1,165,245
Non-trainable params: 0
```

Model performance:





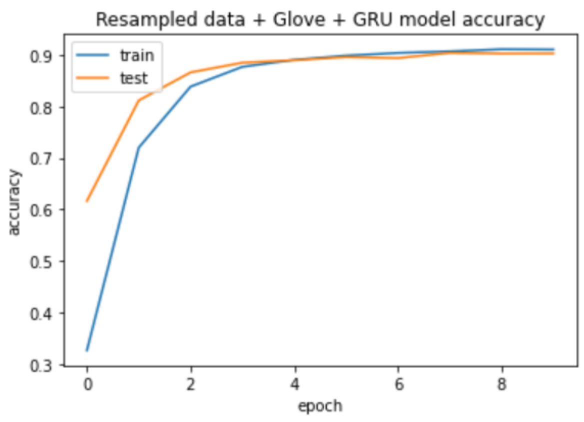
Model Summary:

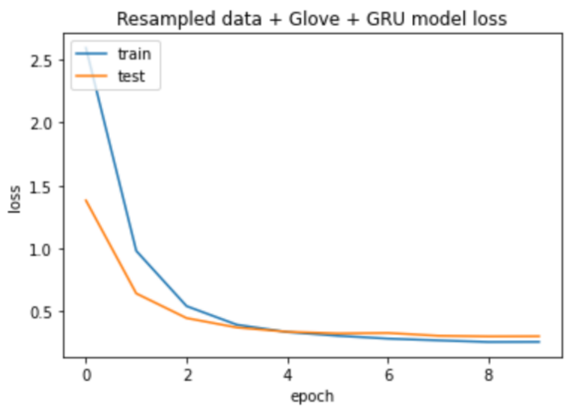
GRU+Glove+Resampled data

```
gruModel_resampled = GruGloveModel()
gruModel_resampled_history, model = gruModel_resampled.train(df_cleaned_resampled,100,epochs)
gruModel_resampled_accuracy, gruModel_resampled_f1 = gruModel_resampled.prediction()
```

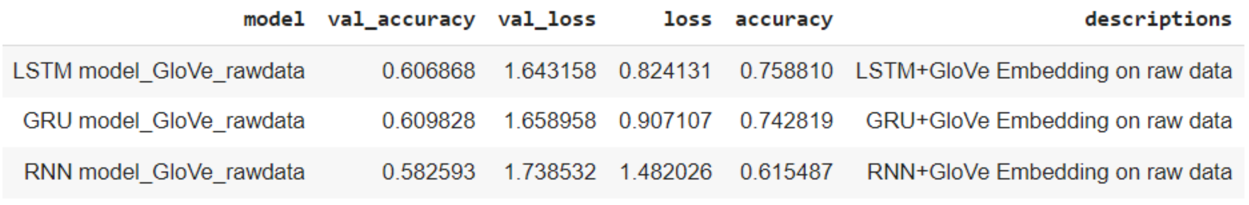
```
Number of Samples: 32389
Number of Labels:  32389
Number of train Samples: 25911
Number of val Samples: 6478
Model: "model_21"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_12 (InputLayer)       [(None, 300)]             0

 embedding_11 (Embedding)    (None, 300, 100)          900100

 gru (GRU)                   (None, 128)               88320

 dropout_11 (Dropout)        (None, 128)               0

 dense_22 (Dense)            (None, 100)               12900

 dense_23 (Dense)            (None, 49)                4949

=================================================================
Total params: 1,006,269
Trainable params: 1,006,269
Non-trainable params: 0
```

Model performance:

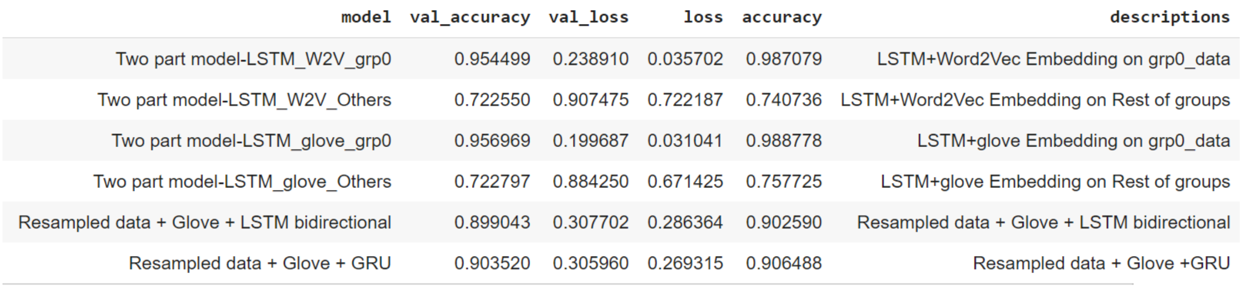




Model performance comparison:

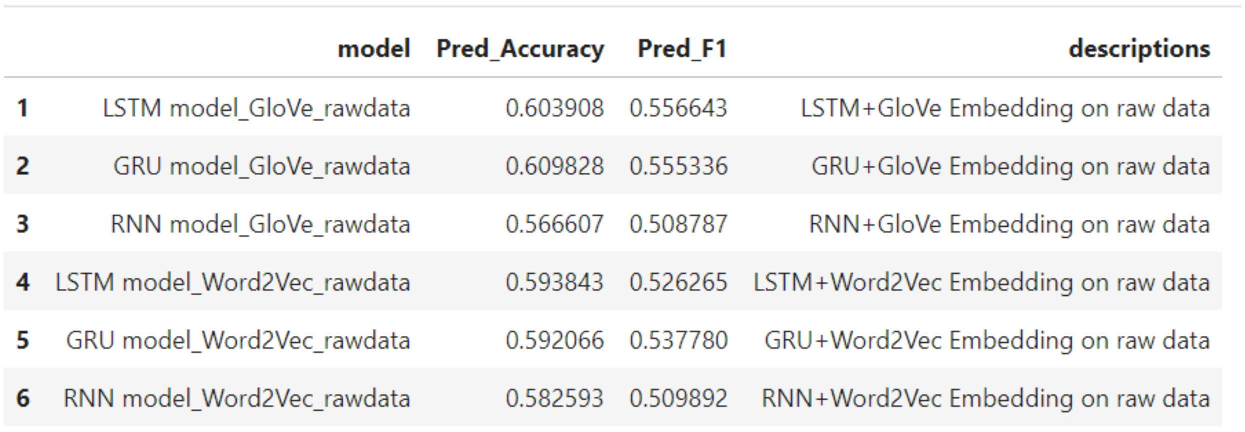


model	val_accuracy	val_loss	loss	accuracy	descriptions
LSTM model_GloVe_rawdata	0.606868	1.643158	0.824131	0.758810	LSTM+GloVe Embedding on raw data
GRU model_GloVe_rawdata	0.609828	1.658958	0.907107	0.742819	GRU+GloVe Embedding on raw data
RNN model_GloVe_rawdata	0.582593	1.738532	1.482026	0.615487	RNN+GloVe Embedding on raw data

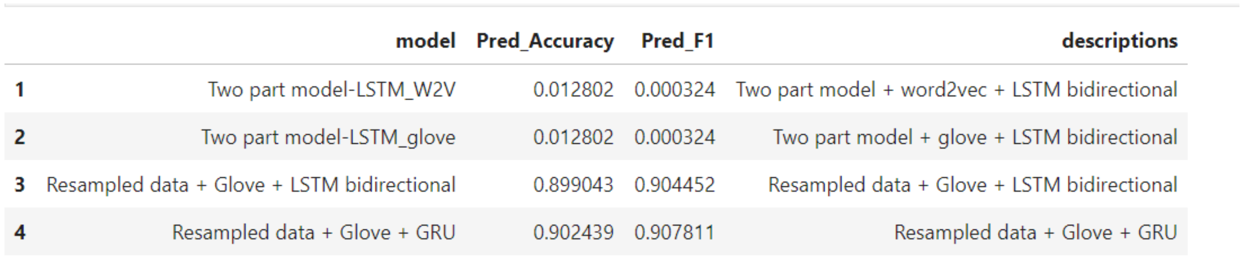


model	val_accuracy	val_loss	loss	accuracy	descriptions
Two part model-LSTM_W2V_grp0	0.954499	0.238910	0.035702	0.987079	LSTM+Word2Vec Embedding on grp0_data
Two part model-LSTM_W2V_Others	0.722550	0.907475	0.722187	0.740736	LSTM+Word2Vec Embedding on Rest of groups
Two part model-LSTM_glove_grp0	0.956969	0.199687	0.031041	0.988778	LSTM+glove Embedding on grp0_data
Two part model-LSTM_glove_Others	0.722797	0.884250	0.671425	0.757725	LSTM+glove Embedding on Rest of groups
Resampled data + Glove + LSTM bidirectional	0.899043	0.307702	0.286364	0.902590	Resampled data + Glove + LSTM bidirectional
Resampled data + Glove + GRU	0.903520	0.305960	0.269315	0.906488	Resampled data + Glove +GRU

Model prediction summary:



	model	Pred_Accuracy	Pred_F1	descriptions
1	LSTM model_GloVe_rawdata	0.603908	0.556643	LSTM+GloVe Embedding on raw data
2	GRU model_GloVe_rawdata	0.609828	0.555336	GRU+GloVe Embedding on raw data
3	RNN model_GloVe_rawdata	0.566607	0.508787	RNN+GloVe Embedding on raw data
4	LSTM model_Word2Vec_rawdata	0.593843	0.526265	LSTM+Word2Vec Embedding on raw data
5	GRU model_Word2Vec_rawdata	0.592066	0.537780	GRU+Word2Vec Embedding on raw data
6	RNN model_Word2Vec_rawdata	0.582593	0.509892	RNN+Word2Vec Embedding on raw data



	model	Pred_Accuracy	Pred_F1	descriptions
1	Two part model-LSTM_W2V	0.012802	0.000324	Two part model + word2vec + LSTM bidirectional
2	Two part model-LSTM_glove	0.012802	0.000324	Two part model + glove + LSTM bidirectional
3	Resampled data + Glove + LSTM bidirectional	0.899043	0.904452	Resampled data + Glove + LSTM bidirectional
4	Resampled data + Glove + GRU	0.902439	0.907811	Resampled data + Glove + GRU

5.3. Final summary from built models

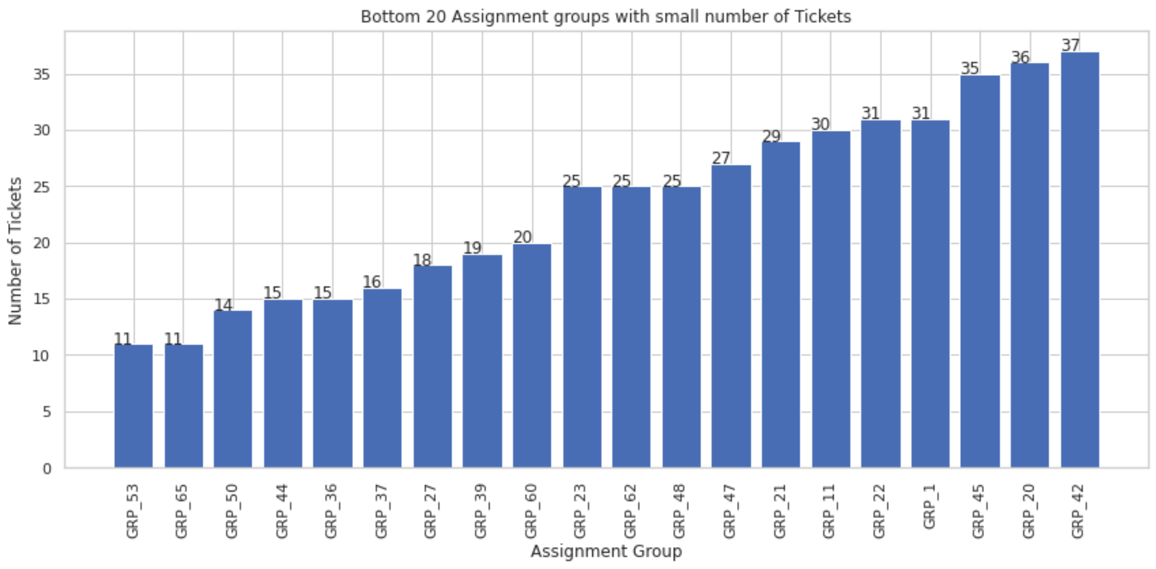
- Resampling of data performs better than Raw data
- Two model approach is performing worse than Raw data models
- Accuracy and F1 score for Resampled data+Glove+GRU combination is more than 90% hence it is the best model according to the analysis

6. Comparison to benchmark

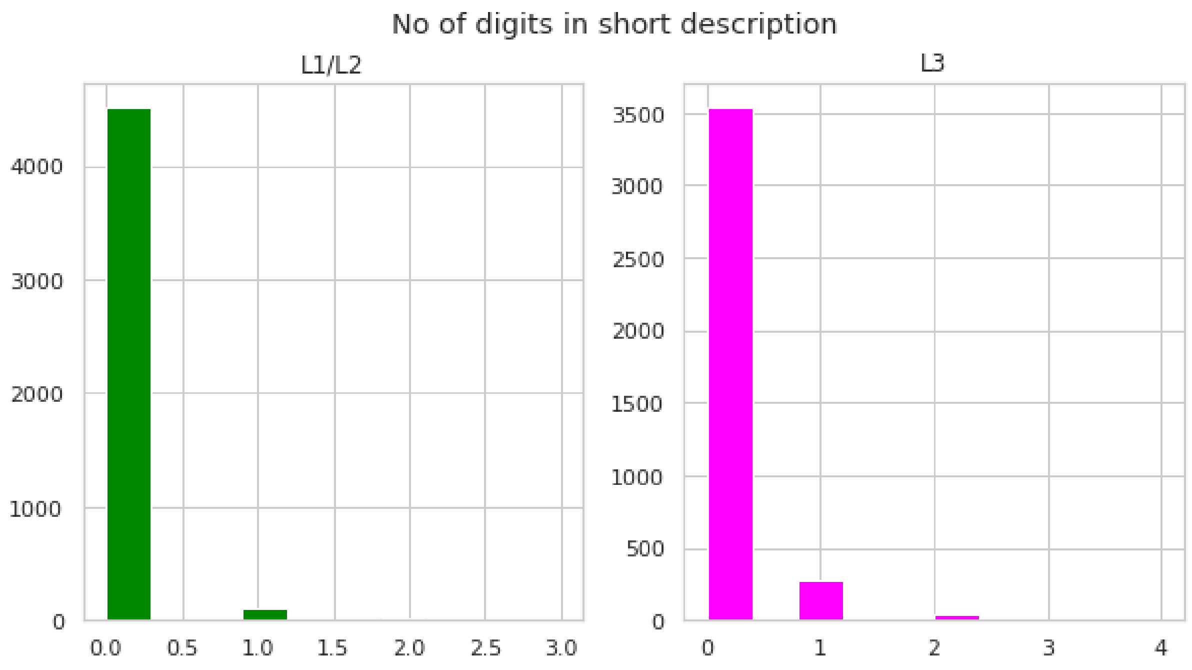
- From the given problem description, we could see that the existing system is able to assign 75% of the tickets correctly.
- So our objective here is to build an AI-based classifier model to assign the tickets to right functional groups by analyzing the given description with an accuracy of at least 85%.
- From the prediction results we see that the GRU model based on the resampled data is able to achieve an accuracy of 90% which is above our benchmark.

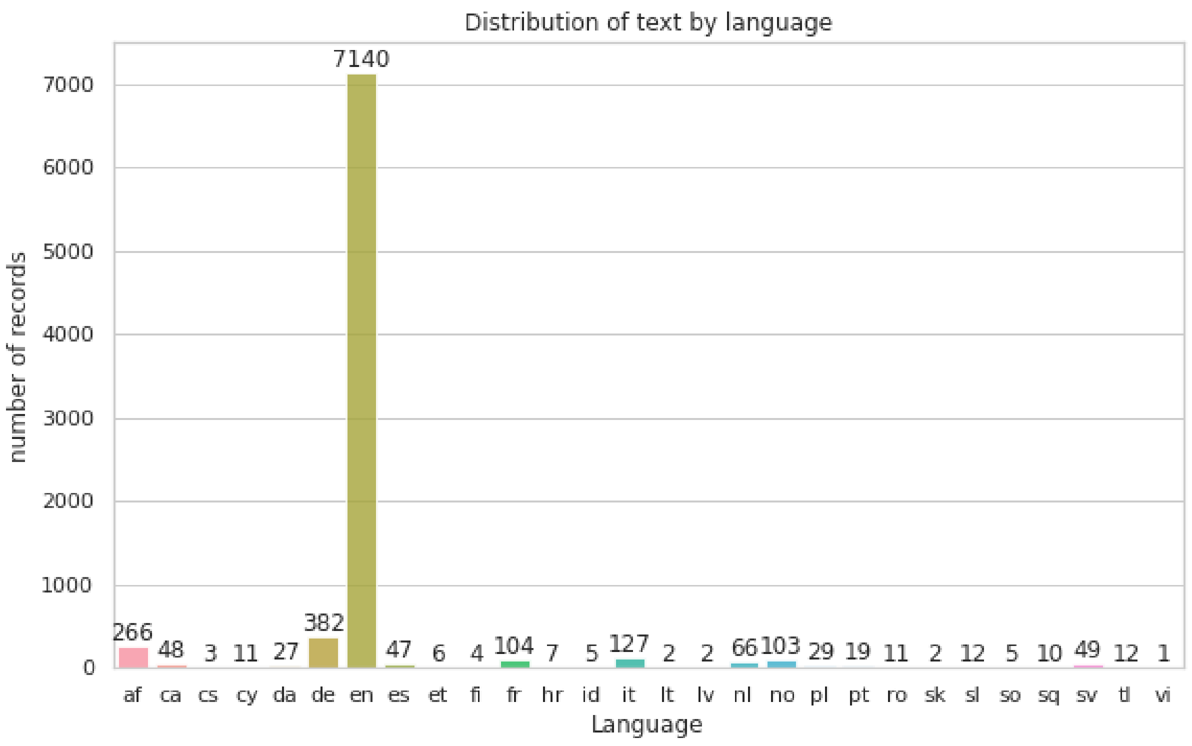
7. Visualizations

- Exploratory Data Analysis

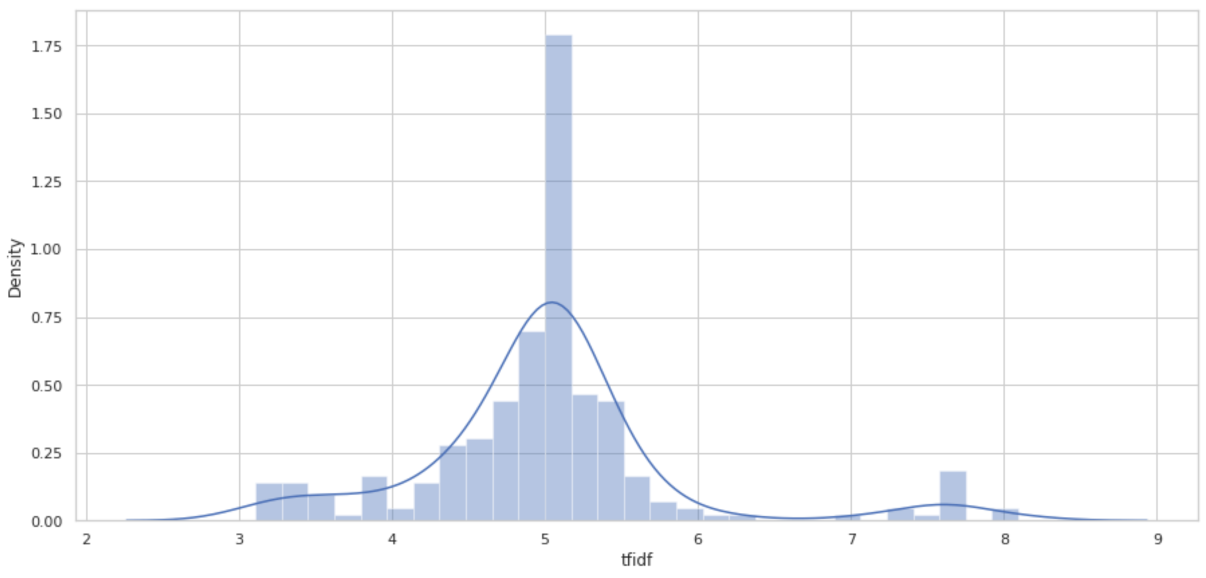


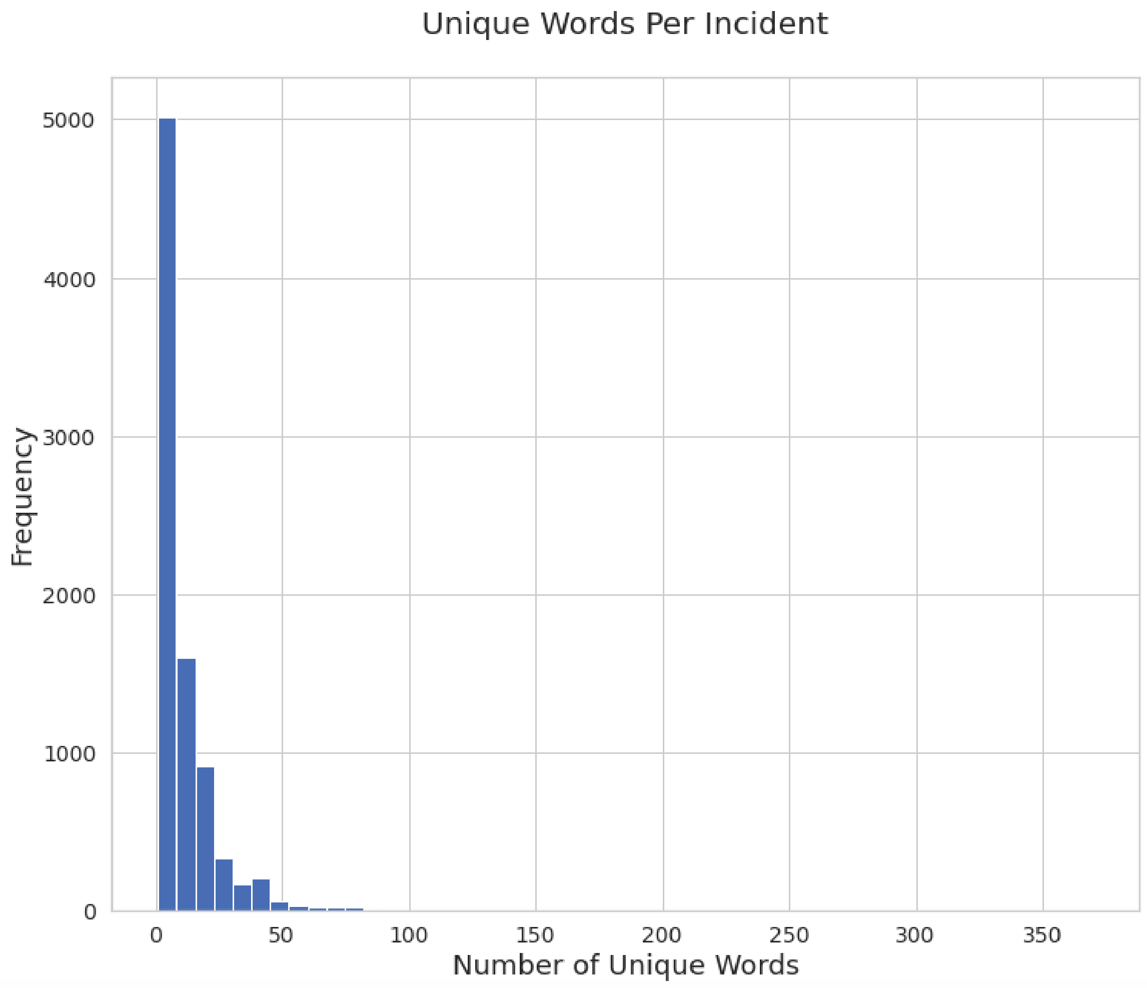
-- The Assigment group consists of 74 different classes



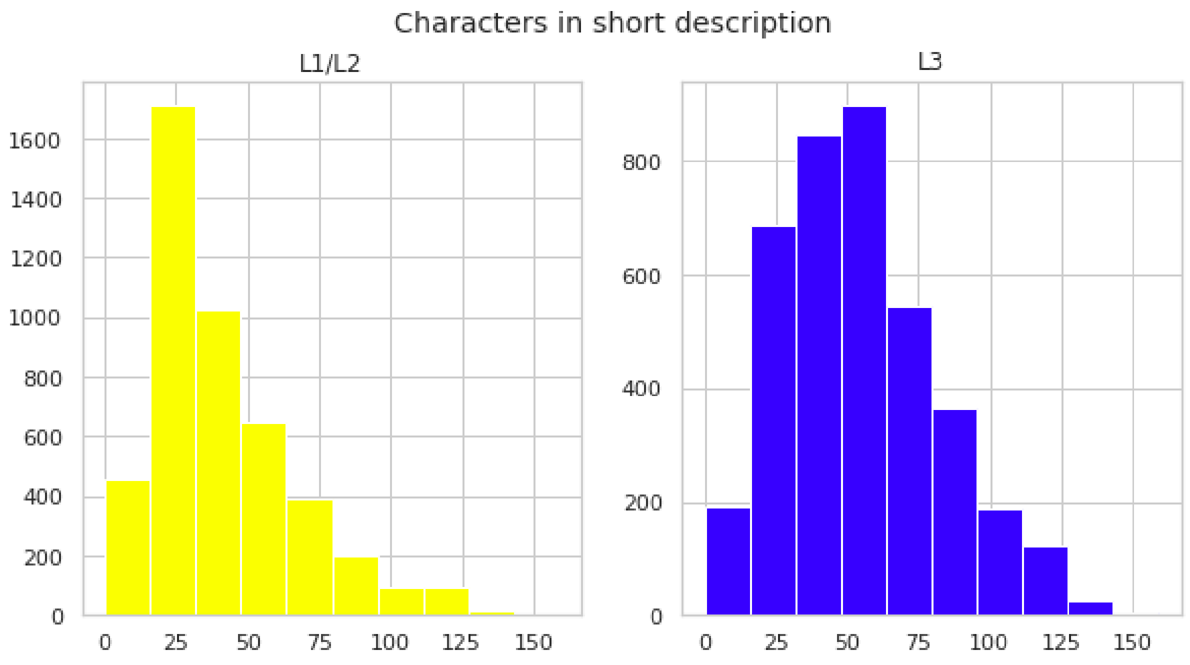


TF-IDF Distribution





Unique Words Per Incident
Frequency
Number of Unique Words
0
1000
2000
3000
4000
5000
0
50
100
150
200
250
300
350



Characters in short description
L1/L2
L3
0
200
400
600
800
1000
1200
1400
1600
0
25
50
75
100
125
150

8. Implications

- This solution can offer ticket assignment with much higher speed than the current system and also with much better accuracy.
- With time, the models can be further improved as and when we get more data points to understand the pattern of ticket assignment to the respective groups.
- This solution is quite scalable as well, it can be applied to multiple domains at once.
- This exercise would definitely help in reducing the workload in terms of FTEs for assignment and evaluation of tickets and L1/L2 levels
- Although this model can classify the IT tickets with 90% accuracy, to achieve better accuracy in the real world it would be good if the business can collect additional data around 300 records for each group.

9. Limitations

- As part of Data pre-processing, we had grouped all assignment groups with less than 10 entries as one group (misc_grp) which had reduced the Target class from 74 to 50 groups.
- While applying this model in the real world there could be additional intervention required to classify the tickets if it has been classified as misc_grp by our model. Since the number of elements reported under misc_grp are less, we expect this intervention to be done less often.

10. Closing Reflections

- We found the data was present in multiple languages and in various formats such as emails, chat, etc bringing in a lot of variability in the data to be analyzed.
- The Business can improve the process of raising tickets via a common unified IT Ticket Service Portal which reduces the above mentioned variability. By doing this, the model can perform better which can help businesses to identify the problem area for relevant clusters of topics.

- **What could have gone better:**
- There were some steps that we as a team also could have performed and/or improved upon.
- At times, CNN architecture also performs well in case of text classifications, we could have explored that route.
- Hyperparameter tuning of some of the better performing models could have given us faster and/or better results.
- This application does not consider samples with lower ticket counts, as a future scope of work,
- We could collect more data around groups with lower ticket samples.
- We could also use any pre-trained word embeddings specific to ITSM tickets.