AUTOMATIC TICKET ASSIGNMENT – FINAL REPORT

Contents

INTRODUCTION

ABSTRACT

One of the key activities of any IT function is 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.

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.

BUSINESS DOMAIN VALUE:

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. Incase 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.

PROBLEM STATEMENT

Incident Management process is expected to **quickly** restore services.

Majority of IT Organizations are yet to Automate Ticket assignment leading to:-

1. Increase in the response and resolution time.
2. Decrease in Customer Satisfaction.
3. Possibility of human error in Ticket Assignment. (~25%)
4. Poor Customer Service.
5. Additional Staffing (~1 FTE) requirement
6. Additional Effort (15 mins for SOP review) requirement.
7. Decrease in morale of Support Resource.
8. Increase in Expenses.
9. Call quality could impact pertinent data capture

OBJECTIVE

Build Multi-Class classifier that can classify the tickets by analyzing text.

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.

MILESTONES

Milestone 1: Pre-Processing, Data Visualization and EDA

A. Exploring the given Data files
B. Understanding the structure of data
C. Missing points in data
D. Finding inconsistencies in the data
E. Visualizing different patterns
F. Visualizing different text features
G. Dealing with missing values
H. Text preprocessing
I. Creating word vocabulary from the corpus of report text data

Creating tokens as required.

Milestone 2: Model Building

A. Building a model architecture which can classify.
B. Trying different model architectures by researching state of the art for similar tasks.
C. Train the model
D. To deal with large training time, save the weights so that you can use them when

training the model for the second time without starting from scratch.

Milestone 3: Test the Model, Fine-tuning and Repeat.

A. Test the model and report as per evaluation metrics
B. Try different models
C. Try different evaluation metrics
D. Set different hyper parameters, by trying different optimizers, loss functions, epochs, learning rate, batch size, checkpointing, early stopping etc. For these models to fine-tune them
E. Report evaluation metrics for these models along with your observation on how changing different hyper parameters leads to change in the final evaluation metric.

DATA SET

Ticket classification will be performed based on the analysis of text within the data available at following location: https://drive.google.com/file/d/1OZNJm81JXucV3HmZroMq6qCT2m7ez7IJ

File Name: Input Data Synthetic (created but not used in our project)

Number of Columns: 4

Number of Rows: 8500

Number of Rows with garbled Text: 828

Number of Rows with non-English language: 824

Following details / Columns of Ticket is available:

1. Short description
   Summary of the Incident
   Blank rows: 2
2. Description Incident
   details Blank rows: 1
3. Caller
   Name of the person who called to inform the incident
   Unique Values: 2950
4. Assignment group
   The ticket queue details, which is our target value.
   Unique Values:74

APPROACH/DESIGN

PACKAGE INSTALLATIONS & CONFIGURATIONS

Import all necessary packages. Install the libraries which are not included in Anaconda distribution by default.

- Extract Glove 6Billion word embeddings. We're going to use the 200d file which has 200 embedding dimensions for each word in the corpus.
- Load the bodies and headlines CSVs for both training and test data and join them together individually to form datasets.

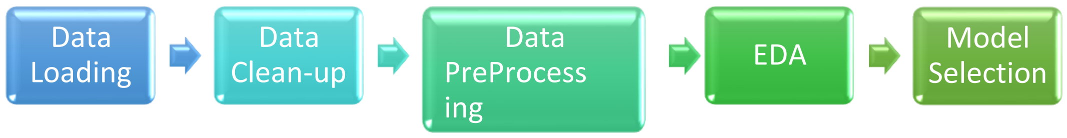
LIBRARIES

Library	Purpose	Remarks
Numpy	Numerical Calculation	
Pandas	Data Handling	
Plotly	Data Visualization	
Seaborn	Data Visualization	
Keras	Sequential Modelling	
Sklearn	Tools & Evaluation metrics	
NLTK	NLP Toolkit	
Mojibake	Text Pre-Processing	Mojibake is the garbled text that is the result of text being decoded using an unintended character encoding. The result is a systematic replacement of symbols with completely unrelated ones, often from a different writing system. This display may include the generic replacement character ("�") in places where the binary representation is considered invalid. A replacement can also involve multiple consecutive symbols, as viewed in one encoding, when the same binary code constitutes one symbol in the other encoding. This is either because of differing constant length encoding (as in Asian 16-bit encodings vs European 8-bit encodings), or the use of variable length encodings (notably UTF-8 and UTF-16). Few such Mojibakes are ¶, ç, å, €, æ, œ, º, ‡, ¼, ¥ etc. As we're dealing with Natural Language and the source of the data is unknown to us, let's run the encoding check to figure out if the dataset is Mojibake impacted. The library ftfy (Fixes Text For You) has a greater ability to

		detect, fix and deal with such Mojibakes. It fixes Unicode that's broken in various ways. The goal of ftfy is to take in bad Unicode and output good Unicode. Installation: using pypi: !pip install ftfy using conda: conda install -c conda-forge ftfy
Language Translation (Goslate: Free Google Translate API)	Text Pre-Processing	Goslate is an open source python library that implemented Google Translate API. This uses the Google Translate Ajax API to make calls to such methods as detect and translate. It is selected over another library Googletrans from Google as Goslate is developed to bypass the ticketing mechanism to prevent simple crawler program to access the Ajax API. Hence Goslate with multiple service URLs is able to translate the entire dataset in very few iterations without blocking the user's IP address. Installation: using pypi: !pip install goslate using conda: conda install -c conda-forge goslate Service URLs used: translate.google.com, translate.google.com.au, translate.google.com.ar, translate.google.co.kr, translate.google.co.in, translate.google.co.jp, translate.google.at, translate.google.de, translate.google.ru, translate.google.ch, translate.google.fr, translate.google.es, translate.google.ae

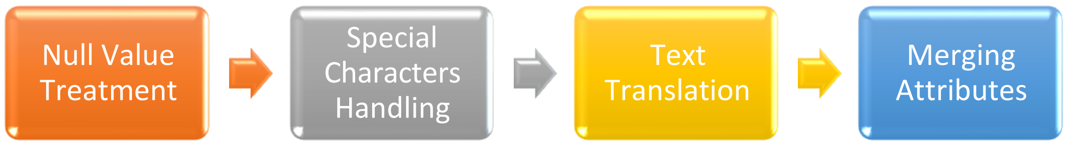
DESIGN

High Level Process flow for the solution is given below



- Data Loading – Loading data from the dataset
- Data Clean-up
- Data Preprocessing
- Exploratory Data Analysis (EDA)
- Model

Data Clean-up Sub flow:



- Null Value Treatment
- Special Character Handling
- Text Translation
- Merging Attributes

OVERVIEW OF THE FINAL PROCESS

DATA VISUALIZATION AND EDA

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.

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.
- Merged Short Description & Description fields for analysis
- Contraction words found in the merged Description are removed for ease of word modelling
- 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

Algorithms used

We have tried below pre-modelling and modelling techniques

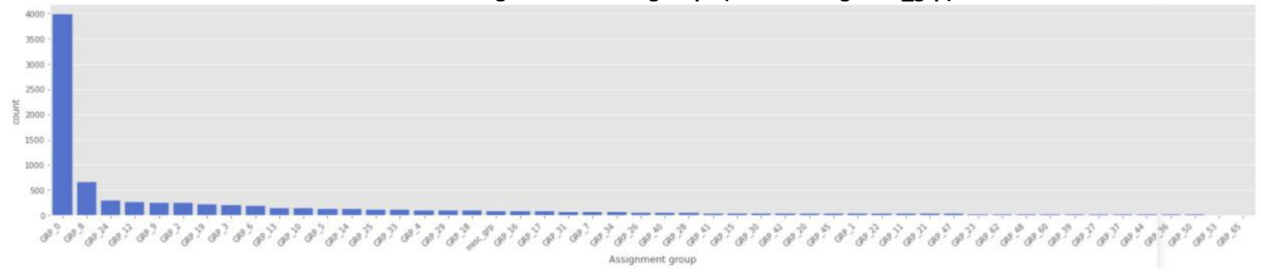
1. Built Bi-gram and Tri-gram models from the bag of words
2. Built Gensim LDA model to understand the data by creating Topic-based clustering
3. Word2Vector Embedding
4. Glove Embedding
5. Bidirectional LSTM Models using both embedding and compared
6. GRU model
7. RNN model
8. Random Forest classifier
9. SVM Classifier model

STEP-BY-STEP WALK THROUGH THE SOLUTION

Data Pre-processing

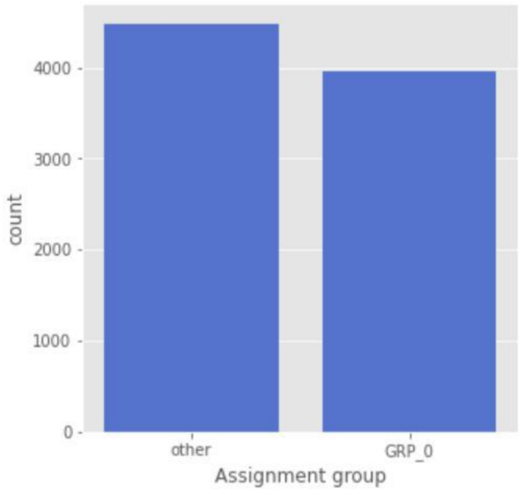
The target class is extremely skewed data. The target class were filtered for less than 10 entries and grouped together as misc_grp as there is no much information with the groups individually.

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

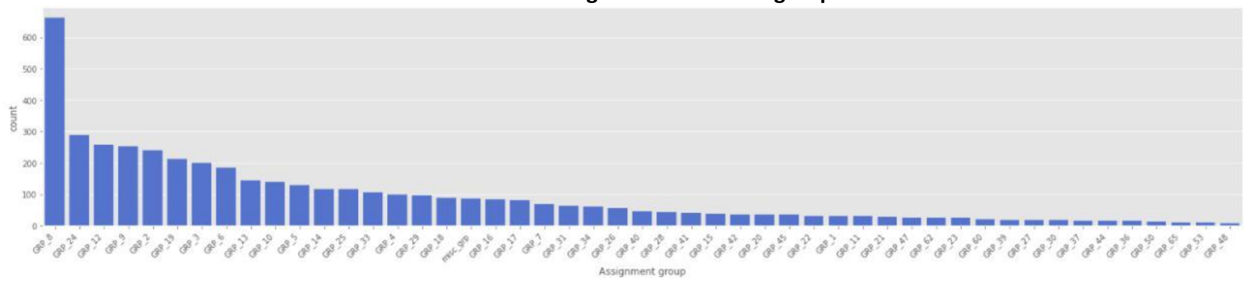


To further address the imbalance in the target class, we have split the dataset as 2 groups

- A dataset where we resample all the groups to size 660. Here note GRP0 would be downsampled & all other groups would be upsampled.
- We could use 2 separate models. Here one model would be used to classify the GRP0 & a second model would be used to classify the other groups. The dataset from the 1st model contains GRP0 data & all the remaining data combined to a single group, say, 'Others'. The dataset for the 2nd Model would contain all groups other than GRP0. The dataset here would be resampled again to address the target imbalance if any.



Distribution of Target class for Other groups



Data cleaning

A function clean_data() 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.

Translation

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.

Initial thought was to combine the 'Short Description' & "Description' field with the assumption that the vocabulary from the 'Short Description' could help in model accuracy. But found that combining these two fields could lead to combining non-english "short Description' to 'Description' in english or vice versa. This posed a problem of many combined entries to be not translated. To further improve the translation process, we attempted to measure the impact of model accuracy on dropping the 'Short Description. By dropping the 'Short Description' field we only observed a minor drop in model accuracy, ~1% drop & hence concluded to proceed with dropping 'Short Description'.

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.

Spell Check

We have used pySpellchecker to perform spell check on the data. But there were few technical words which were also corrected with this function. Eg. Hostage for hostname, sky for skype, wife for wifi, etc. So a set of exceptional words have been loaded with such IT related technical words.

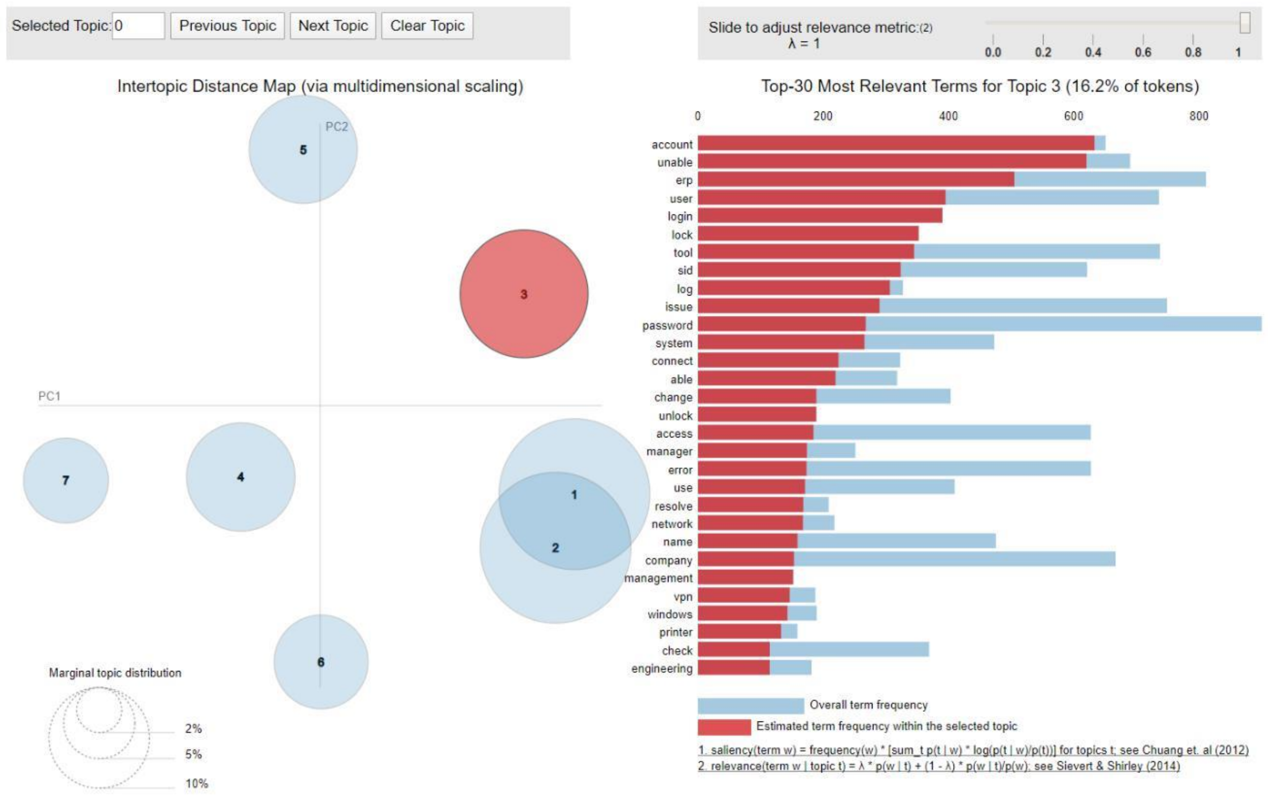
Found that performing spell check was a time consuming process. Although spell check helped in reduction of vocabulary size, it did not help in model accuracy improvement. Hence decided against applying spell check as it did not provide any substantial improvement to the whole process.

Topic Modelling

Spacy module is used for lemmatization. PyLDAvis module is installed and applied for the plot to provide reasonable information on the data based on the topics. Bigram models are used to cluster relevant data together using GenSIM.

7 key topics have been extracted from the model. Relevancy of each topic from the provided data set is ~56% .

Copora dictionary has been created from the bigram words.



Observations from Topic Model Clustering

tool_issue

issue, tool, unable, account, email,

order_issue

order, number, customer, find,

job_failure

receive, job, fail, scheduler, alert,

phone_reset

work, phone, reset, ticket, site,

application_alert

computer,application,message,ser

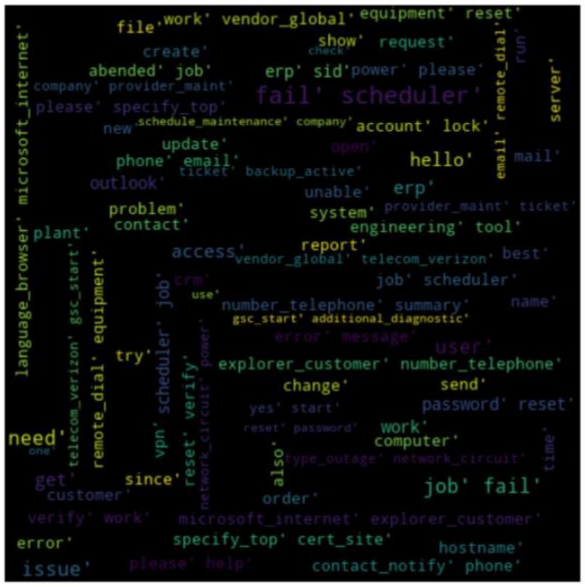
file_drive_issue

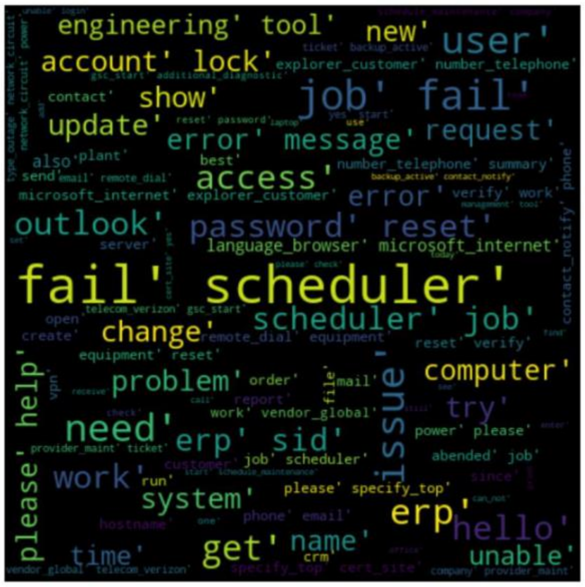
file, drive, folder, server, available

Analysis using WordCloud

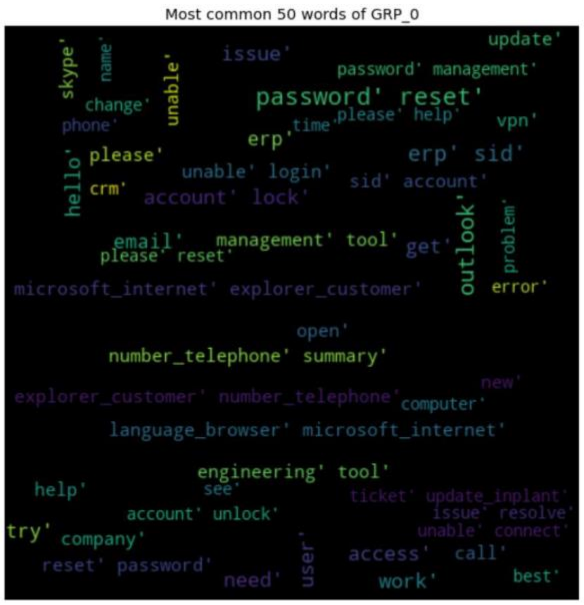
WordCloud is a data visualization technique used for representing text data in which the size of each word indicates its frequency or importance. Significant textual data points can be highlighted using a word cloud. WordClouds have been generated with All available words & top 100 words. We have also inferred few observations over the target class – Assignment groups with word clouds for top 50 words from each group.

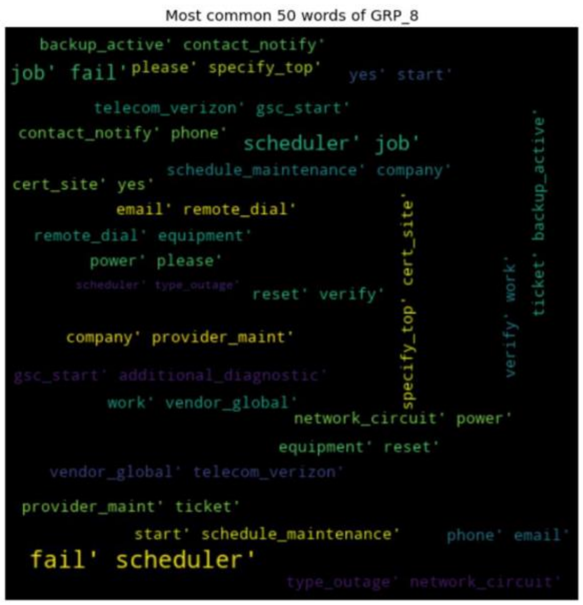
Word clouds for all words and top 100 words are as shown below.





Samples for word clouds based on Assignment Groups:





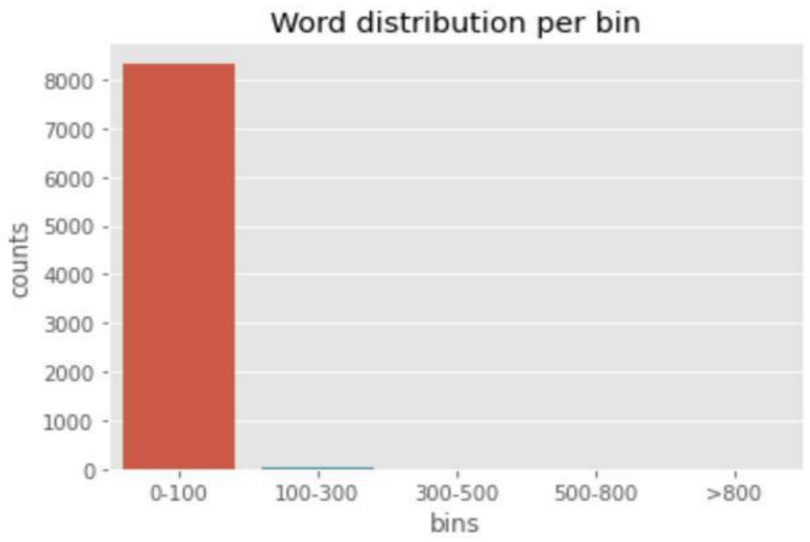
Observations from WordClouds for Top 10 Assignment Groups

Assignment Group	Most Common Words	Observations
Grp_0	password, reset, unable, account, email, explorer_customer, engineering, number_telephone, microsoft_internet, management, collaboration_platform, outlook, crm, erp, lock, phone, skype, computer, error, pron, change, vpn	User account, browser related issues
Grp_8	vendor, equipment, power_provider, outage, job, scheduler, gsc_start, top_cert, contact_notify, site, specify_outage, schedule_maintenance, additional_diagnostic, telecom_Verizon, remote_dial	Network communications related
Grp_24	Problem, defective, tool, printer, computer, setup, install, new_ws, work	System related issues (most words were in German before translations)
Grp_12	server, hostname, drive, folder, access, file, inside_outside	Server related issues
Grp_9	job, fail, scheduler, abended, report	Job Scheduler related issues
Grp_2	sid, event, transaction_code, http, com, pron, message_recreate, authorize, condition_nsu, sle_inspector, access, attached, result, content, hrp, erp	Message/ web related
Grp_19	computer, laptop, printer, monitor, network, connect, software, office, system, pron, error, inside_outside, install, contact, detail, dell	System & Hardware related issues
Grp_3	boot, file, window, location, drive, run, document, computer,	System/OS related

	printer, update, dell, client, tool, replace, outlook, pc, erp, email, inside_outside, user, new, phone, print	
Grp_6	job, fail, scheduler, abended	Job Scheduler related issues (similar to grp_48)
Grp_13	billing, sale, price, inwarehouse, delivery, material, customer, tool, block, can_not, quote, item, fix, time, unable, receive, erp, correct, note, print, send, system, check	Sales related issues

Word Distribution

A plot has been created to analyze the distribution of words in each ticket. It has been found that most of the descriptions of the problems raised by callers are short within 0-100 words. Few entries are a bit descriptive.



Bins	Counts
0-100	8340
100-300	33
300-500	7
500-800	0
>800	0

MODELLING

As the target class is completely skewed, various models have been tried with the below set of datasets to compare each performance. Datasets used for each model are:

Raw data with the target class without any sampling

Resampled data where all the target classes are sampled with a count of 660. (Eg. Grp_0 is down sampled and other groups are up sampled)

Model with Two datasets: Model 1 with Grp_0 & Model 2 with all other groups except Grp_0 and Model 2 is resampled

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.

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.

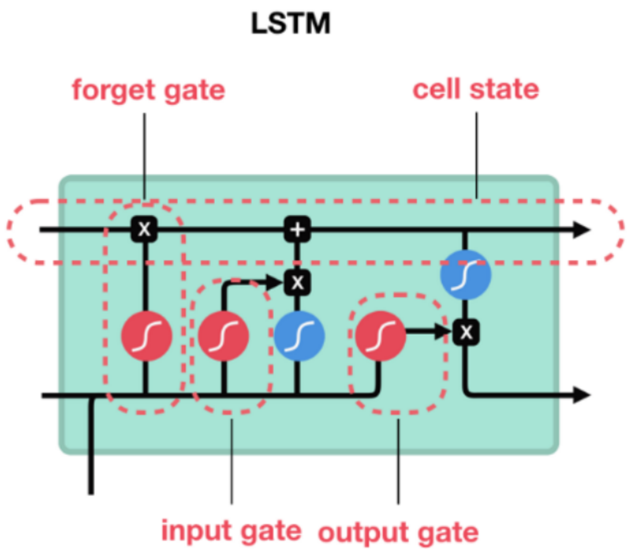
2. **GloVe (Global Vectors) Embedding:**

   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.

Bi-directional LSTM Model

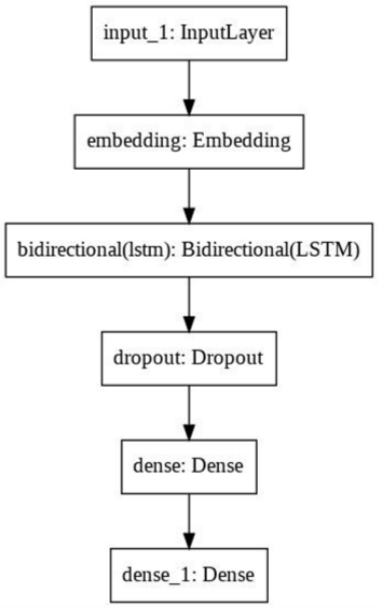
Bidirectional LSTMs are an extension of traditional LSTMs that can improve model performance on classification problems.

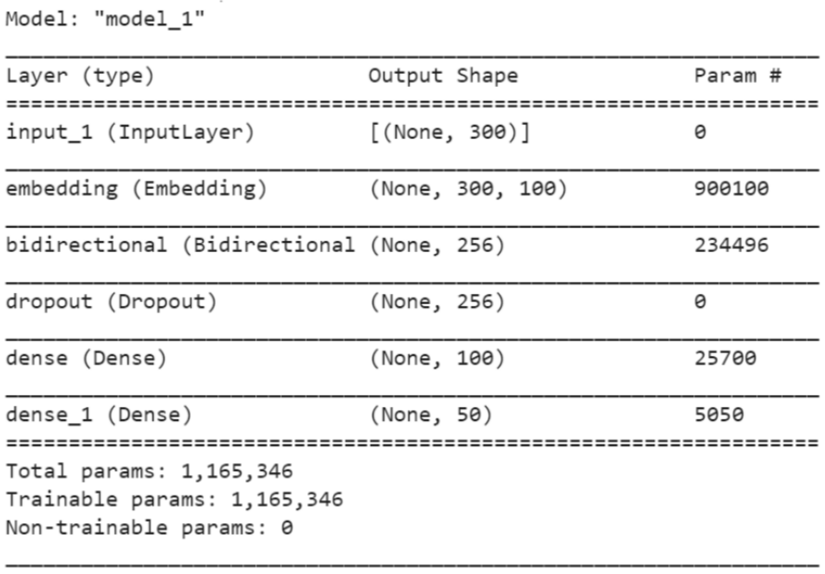
In problems where all timesteps of the input sequence are available, Bidirectional LSTMs train two instead of one LSTMs on the input sequence. The first on the input sequence as-is and the second on a reversed copy of the input sequence. This can provide additional context to the network and result in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



sigmoid
tanh
pointwise multiplication
pointwise addition
vector concatenation

The flow of our Bi-directional LSTM model is as shown.

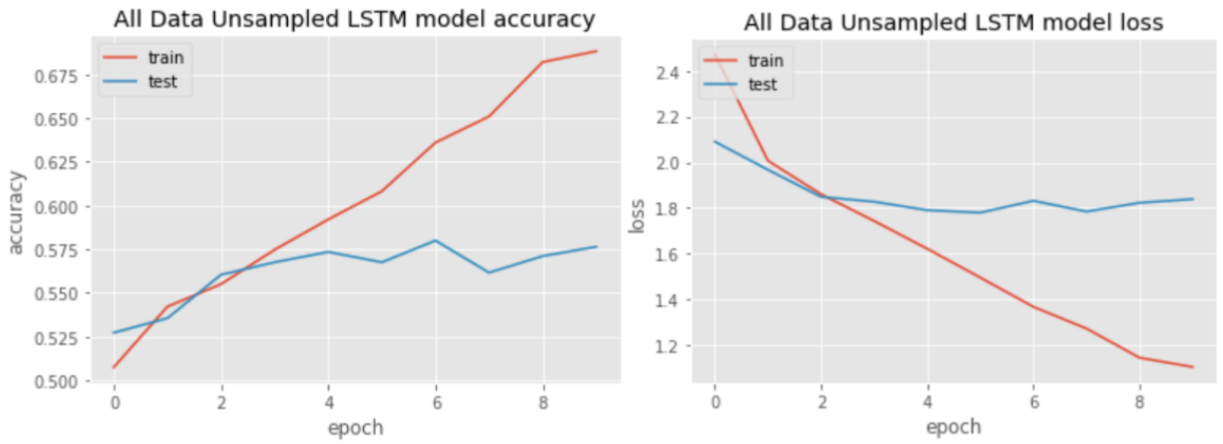


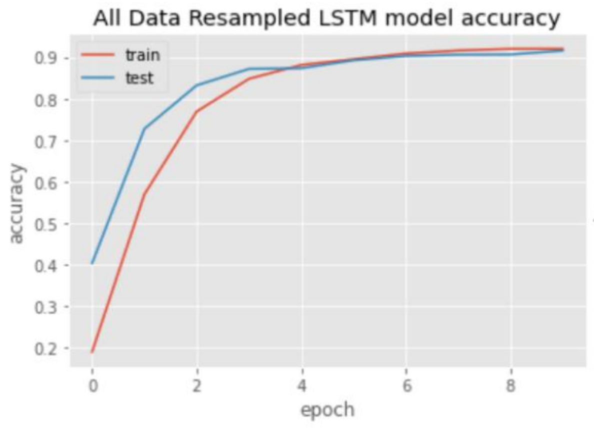


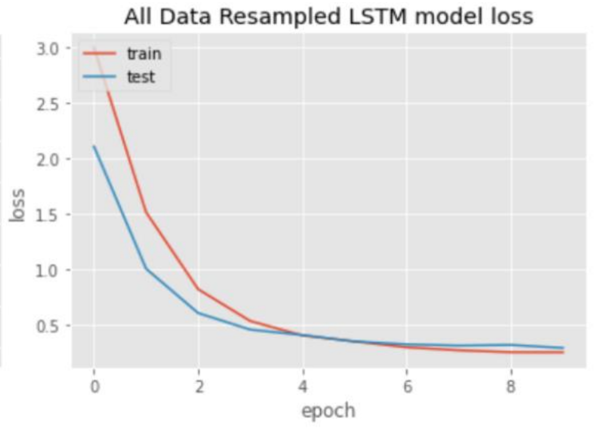
```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 300)]             0
_________________________________________________________________
embedding (Embedding)        (None, 300, 100)          900100
_________________________________________________________________
bidirectional (Bidirectional (None, 256)               234496
_________________________________________________________________
dropout (Dropout)            (None, 256)               0
_________________________________________________________________
dense (Dense)                (None, 100)               25700
_________________________________________________________________
dense_1 (Dense)              (None, 50)                5050
=================================================================
Total params: 1,165,346
Trainable params: 1,165,346
Non-trainable params: 0
_________________________________________________________________
```

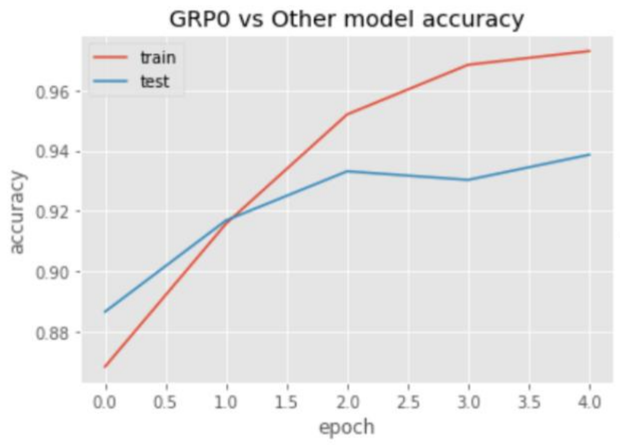
Observations from LSTM Model

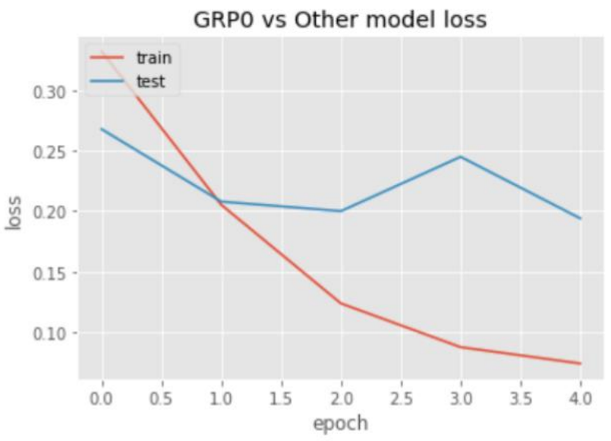
Word2Vec

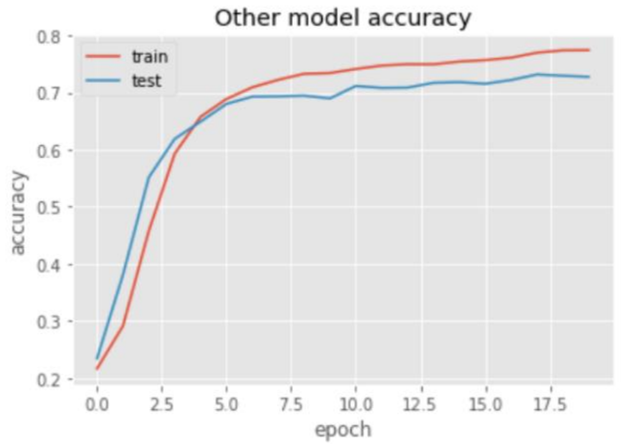


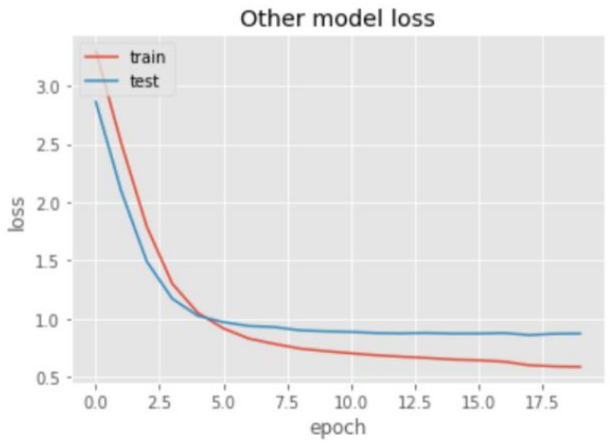








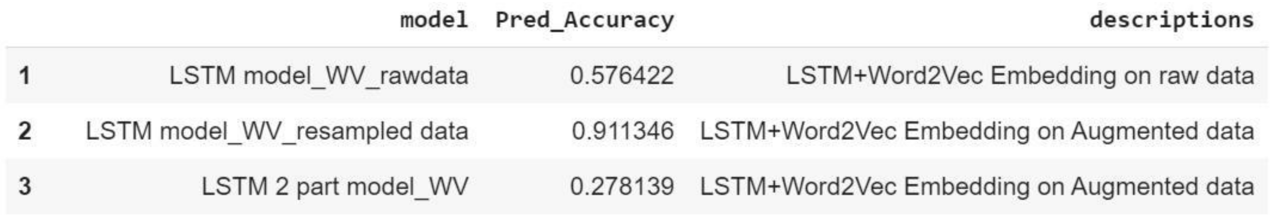




Training Accuracy for LSTM with Word2Vec

	model	val_accuracy	val_loss	loss	accuracy	descriptions
1	LSTM model_WV_rawdata	0.579976	1.831799	1.368159	0.636054	LSTM+Word2Vec Embedding on raw data
2	LSTM model_WV_resampled data	0.911346	0.303620	0.237115	0.922504	LSTM+Word2Vec Embedding on Augmented data
3	LSTM 2 part model_WV_grp0	0.938643	0.193967	0.073982	0.972977	LSTM+Word2Vec Embedding on grp0_data
4	LSTM 2 part model_WV_Others	0.724455	0.893221	0.597537	0.771313	LSTM+Word2Vec Embedding on Rest of groups

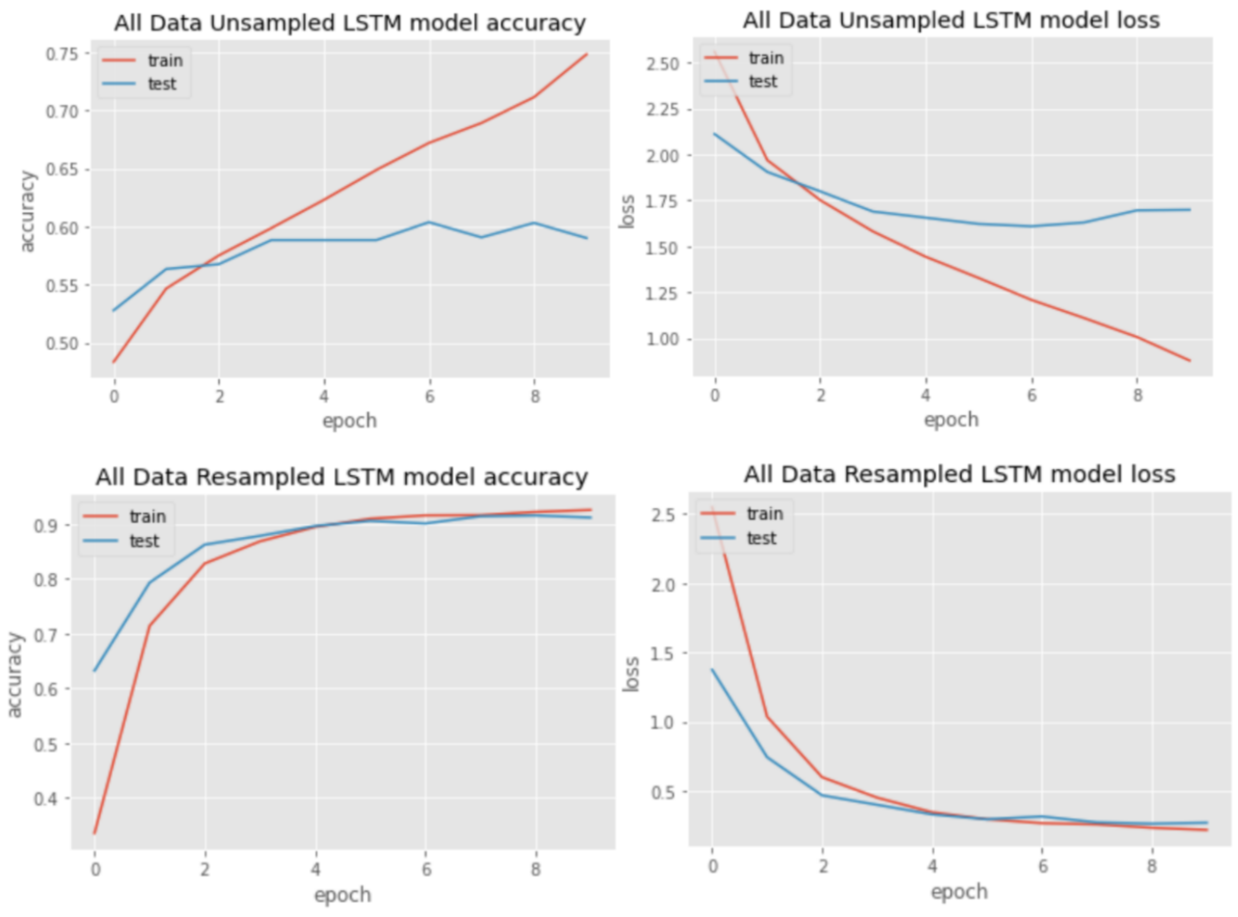
Test Accuracy for LSTM with Word2Vec



	model	Pred_Accuracy	descriptions
1	LSTM model_WV_rawdata	0.576422	LSTM+Word2Vec Embedding on raw data
2	LSTM model_WV_resampled data	0.911346	LSTM+Word2Vec Embedding on Augmented data
3	LSTM 2 part model_WV	0.278139	LSTM+Word2Vec Embedding on Augmented data

Observation: *The Two part models prediction accuracy is very low compared to other 2 models. Hence we are dropping the 2 part model for further experiments and we will generate models based on 2 datasets namely Unsampled data and Resampled data.*

Glove Embedding

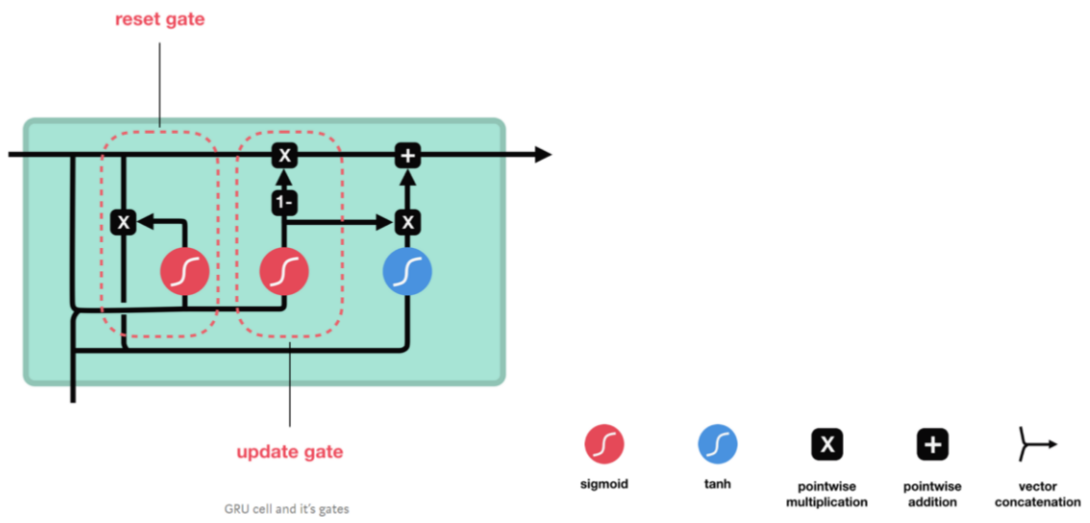


Note: *From Bi-LSTM with both Word2Vec and Glove embedding, we observed better performance with Glove Embedding. So we have proceeded with all other models using Glove Embedded data.*

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, update gate and 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.



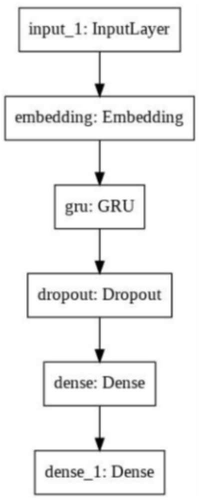
GRU cell and it's gates

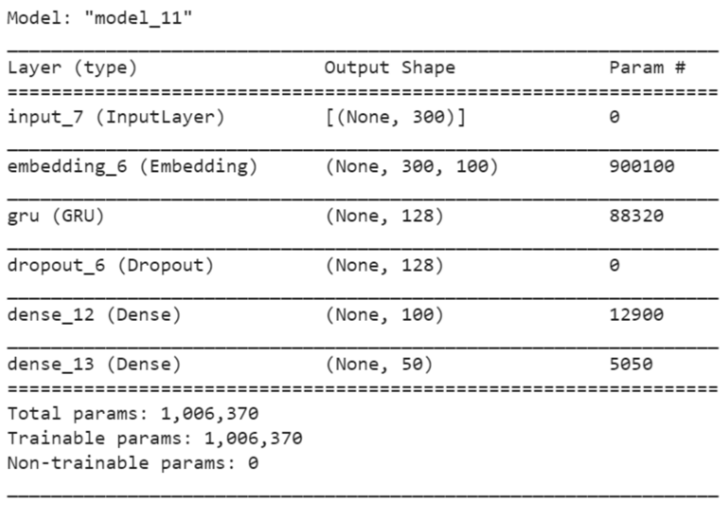
Update Gate: The update gate acts similar to the forget and input gate of an LSTM. It decides what information to throw away and what new information to add.

Reset Gate: 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.

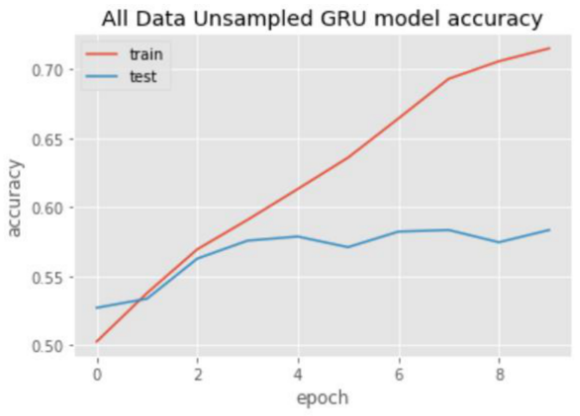
The flow of our GRU model is as shown.

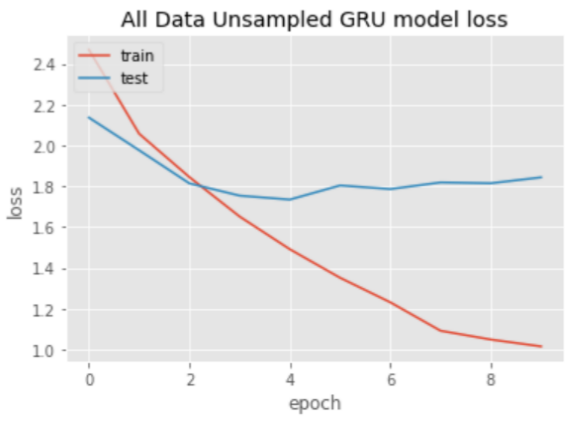


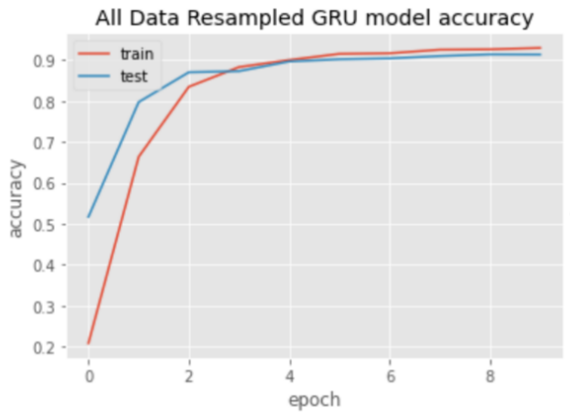


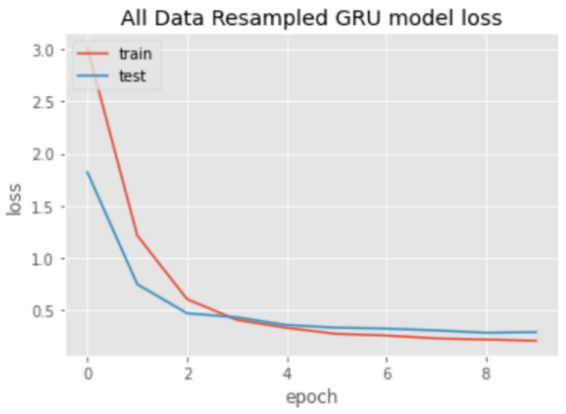
```
Model: "model_11"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_7 (InputLayer)         [(None, 300)]             0
_________________________________________________________________
embedding_6 (Embedding)      (None, 300, 100)          900100
_________________________________________________________________
gru (GRU)                    (None, 128)               88320
_________________________________________________________________
dropout_6 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_12 (Dense)             (None, 100)               12900
_________________________________________________________________
dense_13 (Dense)             (None, 50)                5050
=================================================================
Total params: 1,006,370
Trainable params: 1,006,370
Non-trainable params: 0
_________________________________________________________________
```

Observations from GRU Model







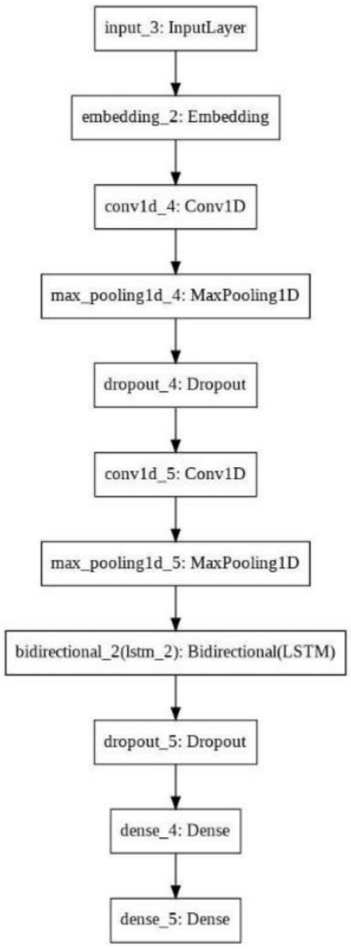


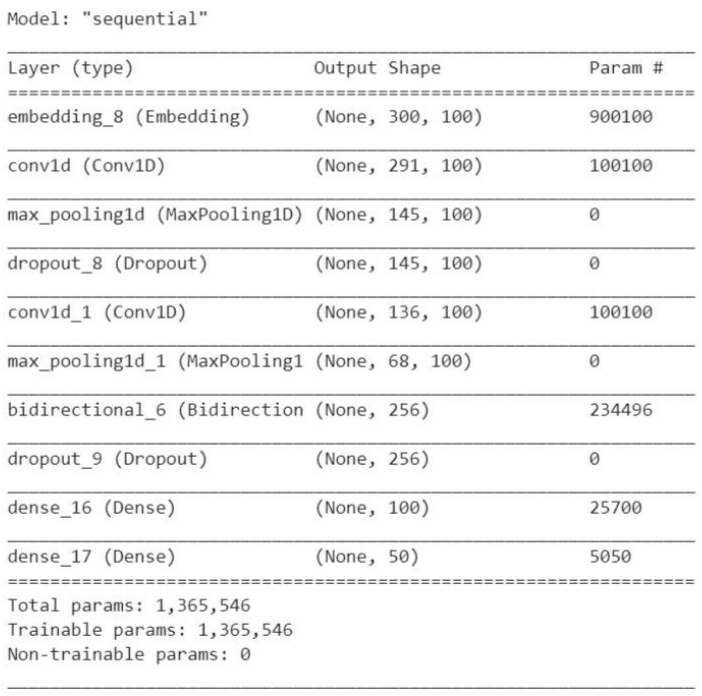
RNN Model

Recurrent Neural Networks (RNNs) are a family of neural networks designed specifically for sequential data processing. The RNN model will do prediction of the next word in a sequence based on the previous ones. The same operation is performed recurrently which is why it is called as Recurrent Neural Networks.

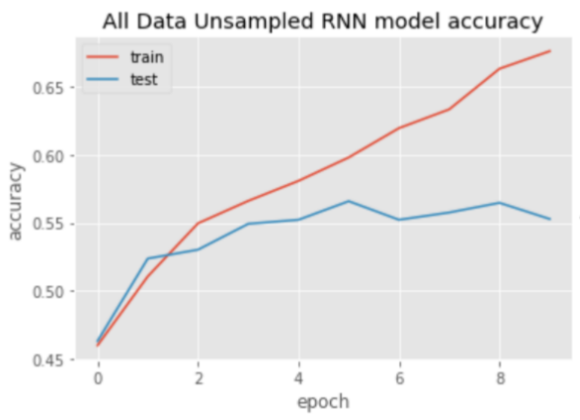
RNNs perform the same task for every element of a sequence, with the output being dependent on the previous computations. Another way to think about RNNs is that they have a “memory” which captures information about what has been calculated so far.

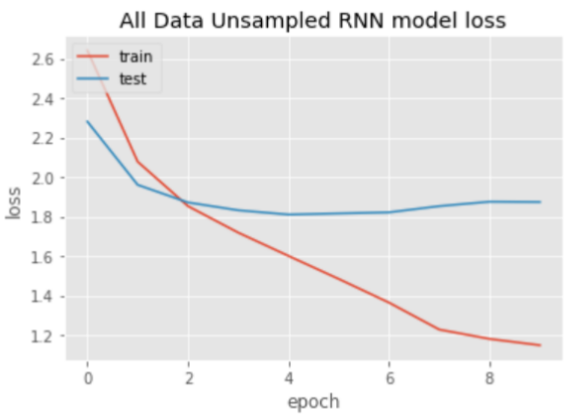
The flow of our RNN model is as shown.

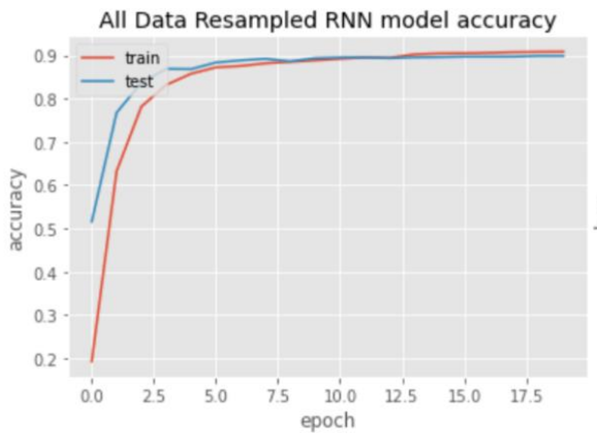


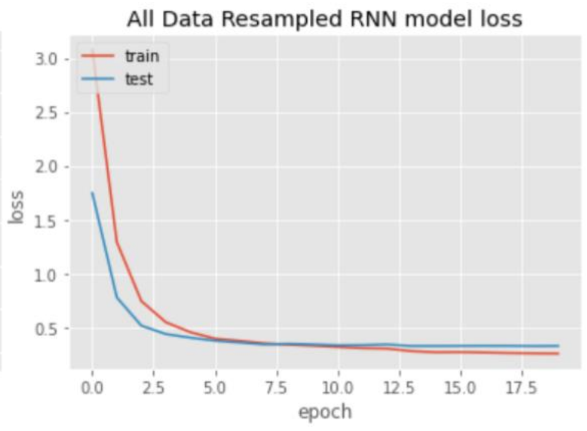

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
embedding_8 (Embedding)      (None, 300, 100)          900100    
_________________________________________________________________
conv1d (Conv1D)              (None, 291, 100)          100100    
_________________________________________________________________
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, 50)                5050      
=================================================================
Total params: 1,365,546
Trainable params: 1,365,546
Non-trainable params: 0
_________________________________________________________________
```

Observations from RNN Model





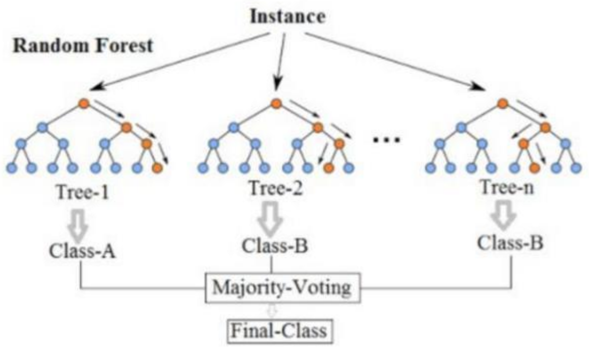




Traditional ML Models

Random Forest Classifier Model

Random forests is a supervised learning algorithm. It can be used both for classification and 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.



Structure of Random Forest Classification

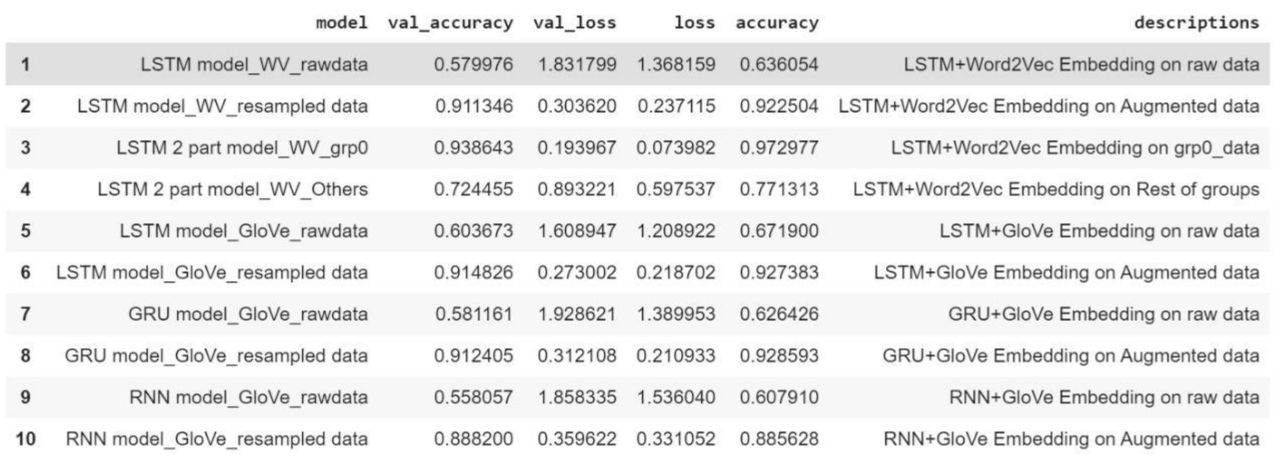
Observation from our experiments: Random Forest Model using raw data seems to perform better as compared to Random Forest Model with PCA done. It also affirms the understanding that PCA uses linearity in the data to reduce dimensionality, whereas the problems such NLP uses the non-linearity of feature.

Support Vector Machines (SVM) Model

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 new text. This is a fast and dependable classification algorithm that performs very well with a limited amount of data.

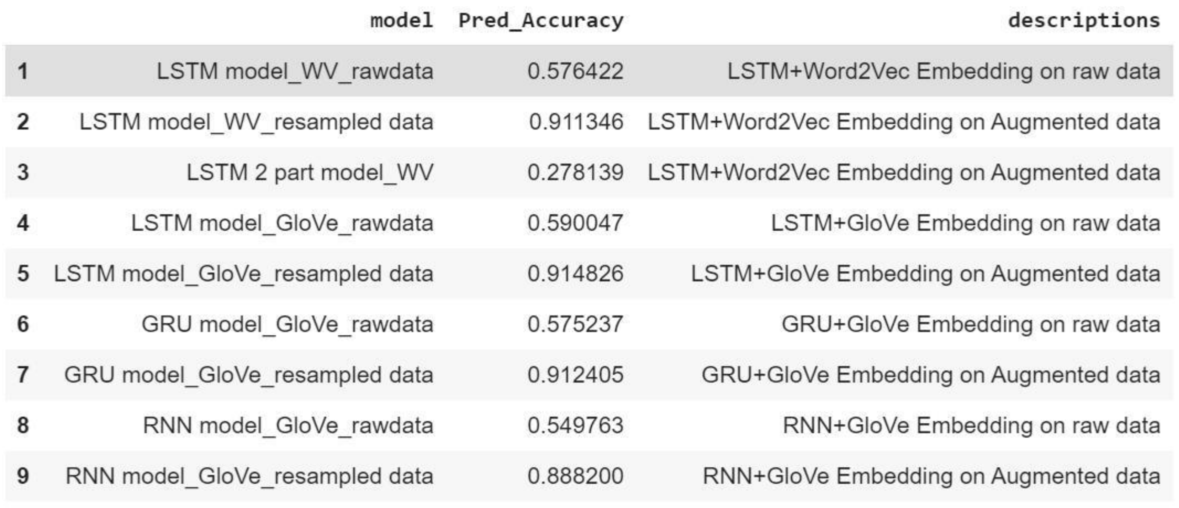
MODEL EVALUATION

Training Accuracy for each model



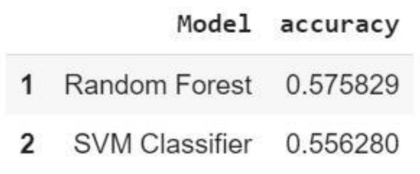
	model	val_accuracy	val_loss	loss	accuracy	descriptions
1	LSTM model_WV_rawdata	0.579976	1.831799	1.368159	0.636054	LSTM+Word2Vec Embedding on raw data
2	LSTM model_WV_resampled data	0.911346	0.303620	0.237115	0.922504	LSTM+Word2Vec Embedding on Augmented data
3	LSTM 2 part model_WV_grp0	0.938643	0.193967	0.073982	0.972977	LSTM+Word2Vec Embedding on grp0_data
4	LSTM 2 part model_WV_Others	0.724455	0.893221	0.597537	0.771313	LSTM+Word2Vec Embedding on Rest of groups
5	LSTM model_GloVe_rawdata	0.603673	1.608947	1.208922	0.671900	LSTM+GloVe Embedding on raw data
6	LSTM model_GloVe_resampled data	0.914826	0.273002	0.218702	0.927383	LSTM+GloVe Embedding on Augmented data
7	GRU model_GloVe_rawdata	0.581161	1.928621	1.389953	0.626426	GRU+GloVe Embedding on raw data
8	GRU model_GloVe_resampled data	0.912405	0.312108	0.210933	0.928593	GRU+GloVe Embedding on Augmented data
9	RNN model_GloVe_rawdata	0.558057	1.858335	1.536040	0.607910	RNN+GloVe Embedding on raw data
10	RNN model_GloVe_resampled data	0.888200	0.359622	0.331052	0.885628	RNN+GloVe Embedding on Augmented data

Test Accuracy for each model



	model	Pred_Accuracy	descriptions
1	LSTM model_WV_rawdata	0.576422	LSTM+Word2Vec Embedding on raw data
2	LSTM model_WV_resampled data	0.911346	LSTM+Word2Vec Embedding on Augmented data
3	LSTM 2 part model_WV	0.278139	LSTM+Word2Vec Embedding on Augmented data
4	LSTM model_GloVe_rawdata	0.590047	LSTM+GloVe Embedding on raw data
5	LSTM model_GloVe_resampled data	0.914826	LSTM+GloVe Embedding on Augmented data
6	GRU model_GloVe_rawdata	0.575237	GRU+GloVe Embedding on raw data
7	GRU model_GloVe_resampled data	0.912405	GRU+GloVe Embedding on Augmented data
8	RNN model_GloVe_rawdata	0.549763	RNN+GloVe Embedding on raw data
9	RNN model_GloVe_resampled data	0.888200	RNN+GloVe Embedding on Augmented data

Traditional Models Accuracy



	Model	accuracy
1	Random Forest	0.575829
2	SVM Classifier	0.556280

From the predicted accuracies, we could see that the LSTM and GRU models with the resampled Augmented dataset is performing well with 91.48% and 91.24% respectively. Although the differences in the accuracy are marginal, we have decided to go with the GRU model as it is faster than LSTM and it takes care of the vanishing gradient problem.

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 analysing 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 91.24% which is above our benchmark.

IMPLICATIONS

Although this model can classify the IT tickets with 91.24% 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.

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.

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.