Automatic Ticket Assignment

Shikha Sharma

Sahithi Oddiraju

Ashutosh Shukla

Viswanathan Mukundan

Sneharema Valiyachembazi

Table of Contents

1.Summary of problem statement, data and findings

Understanding the Business scenario:

In any IT industry, Incident Management plays a vital role in delivering quality support to its end users. If any interruption happening to the smooth business flow it will consider as an incident. When the end user intimates the issue to the concerned helpdesk/support team(L1) they will analyse the issue and categorise it based on the nature of incident to different categories and assigns to the corresponding L2/L3 teams inside the organisation.

In large organisations there will be different departments to handle various business functionalities. If we use the traditional manual assignment of incidents there might have below disadvantages:

- Huge human capital to be invested to analyse and categorise the issue.
- There is a possibility of assigning incident to wrong group
- Assigning the tickets may get delayed
- Resolution time may get longer
- Some complicated issue analysis and categorisation may hold the productive time of the service desk executives.

If we could automate the incident management, then the resolution rate can be faster and thus end user satisfaction will be more. The support team can work on more productive activities.

Project Objective

From the given problem description, we could see that the existing system is able to assign 57% 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 more accuracy.

Dataset Observations

Data provided in format: Excel

Total Records:8500

Data Fields:

Short description	A summary of the issue faced by the user
Description	Detailed description of the issue
Assignment group	GRP_0 ~ GRP_73 (total 74 classes of Assignment group)
Caller	User on behalf of customer who is raising the ticket

- ‘GRP_0’ is occupying the 46.7 percent of representation, while GRP_70 has the minimum number of values
- Many classes with very little representation.
- There 74 Assignment groups. i.e., - Target classes
- There are 25 groups which have under 10 samples
- There are 50 groups which have less than 50 samples
- Caller field may not be useful for training data as it is available in a random mode
- Description found in non-English language
- Email/chat format in description
- Special characters are present in the description
- Hyperlinks, URLS & few image data found in the description
- short description and description field has blank values
- in some records descriptions and short descriptions are same
- Few words were combined together
- Spelling mistakes and typo errors are found in description
- Sample data

Short description	**Description**	**Caller**	**Assignment group**
login issue	-verified user details.(employee# & manager na...	spxjnwir pjlcoqds	GRP_0
when undocking pc , screen will not come back	when undocking pc , screen will not come back	sigfdwcj reofwzlm	GRP_3
cant log in to vpn	\r\n\r\nreceived from: eylqgodm.ybqkwiam@gmail...	eylqgodm ybqkwiam	GRP_0
job Job_1424 failed in job_scheduler at: 10/31/2016 09:06:00	received from: monitoring_tool@company.com job Job_1424 failed in job_scheduler at: 10/31/2016 09:06:00	bpctwhsn kzqsbmtp	GRP_6
job Job_1390 failed in job_scheduler at: 10/29/2016 03:55:00	received from: monitoring_tool@company.com job Job_1390 failed in job_scheduler at: 10/29/2016 03:55:00	bpctwhsn kzqsbmtp	GRP_9

2. Overview of the final process

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 46.7% of the data
- 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

Cleaning processes applied with below steps:

- Removal of trailing spaces
- Removal of line breaks and tabs (\r\n\t)
- Removal of special characters
- Removal of extra spaces
- Removal of frequently used stop words

Handling the missing values:

- There are two columns Description and short description which have 1,8 null values respectively in the given data frame.
- Missing value imputation: Since short description is extracted mostly from Description, hence imputing the short description values into description.

Removing the unwanted columns:

- The objective of the problem is to categorise the issues based on description and assign them to various groups. Since the short description is extracted from the description only, we can remove the column.
- The objective is also to analyse the description and assign tickets, hence the caller column does not hold any value for the classification. There might be certain correlation However it will add bias. Hence removing the column also

Handling the columns with less representation:

- The classes with less than 75 values are grouped into a single column named ‘Others’ which has reduced the overall number of classes.

Translation of Non-English words:

- Since most of the tickets raised are in English language, translate the words from other languages

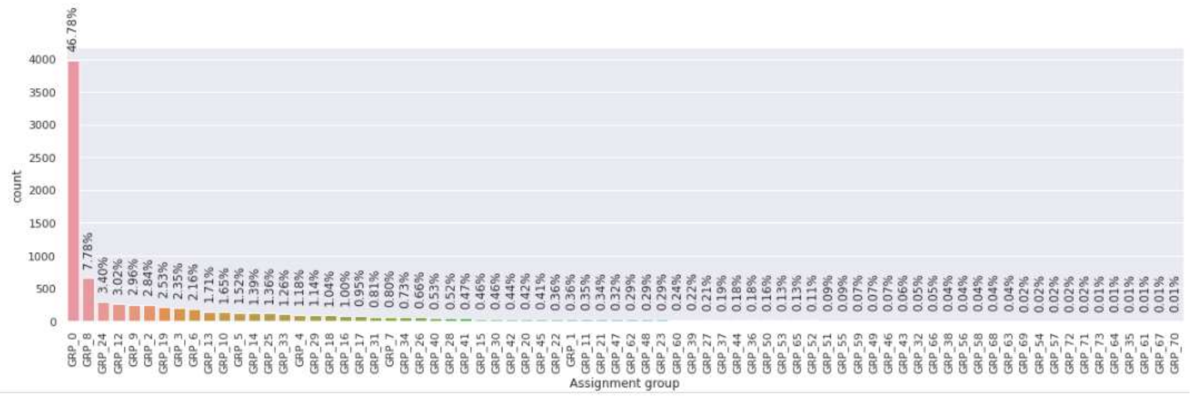
Algorithms used

We have tried below pre-modelling and modelling techniques

1 Word2Vector Embedding
2 Glove Embedding
3 Bidirectional LSTM Models using both embedding and compared
4 GRU model
5 RNN model
6 SVM Classifier model

3. Step-by-step walk through the solution

Data Pre-processing



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_text function is applied to the Description column and cleaned up data is generated for further and to remove the non English words.

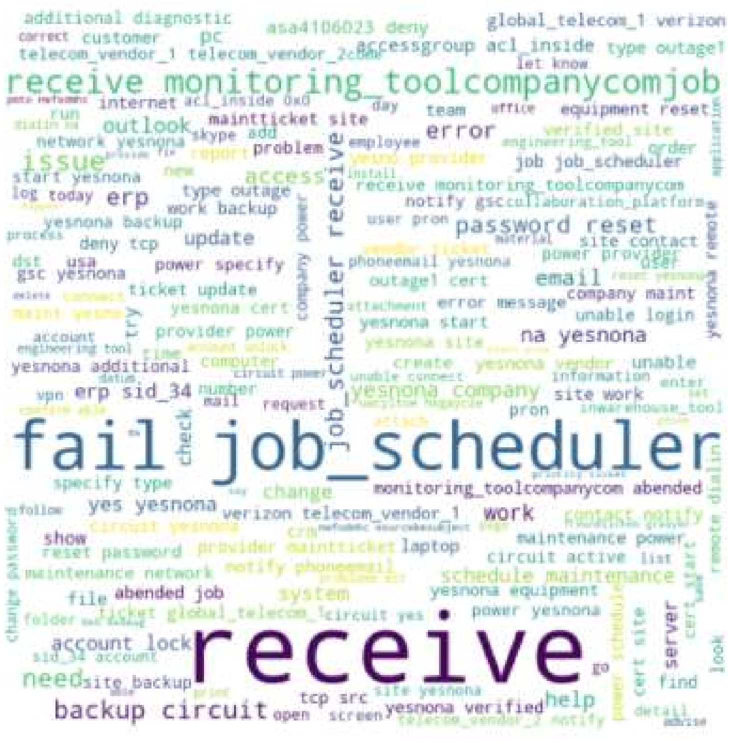
Lemmatization & Removal of stop words

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 analysed 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.

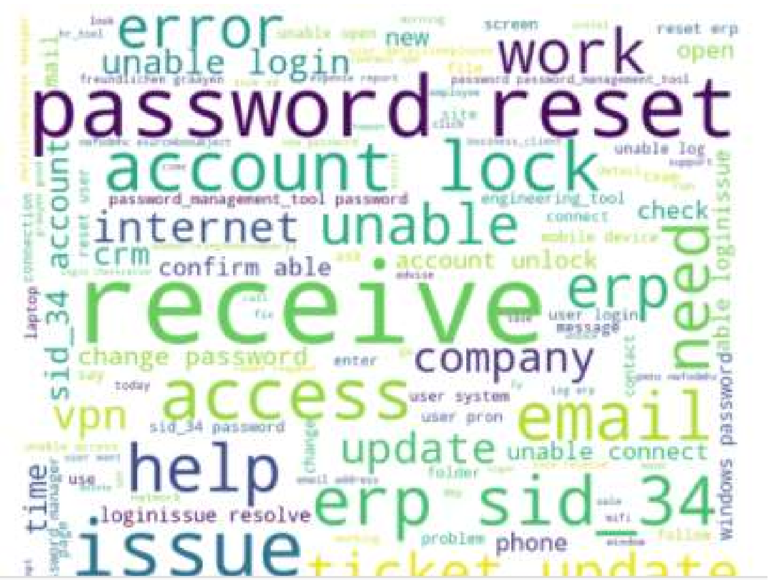
WordCloud Analysis

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. WordCloud have been generated with All available words & top 100 words. We have also inferred few observations over the target class – The word clouds of top 3 assignment groups was build to see the kind of ticket assigned.

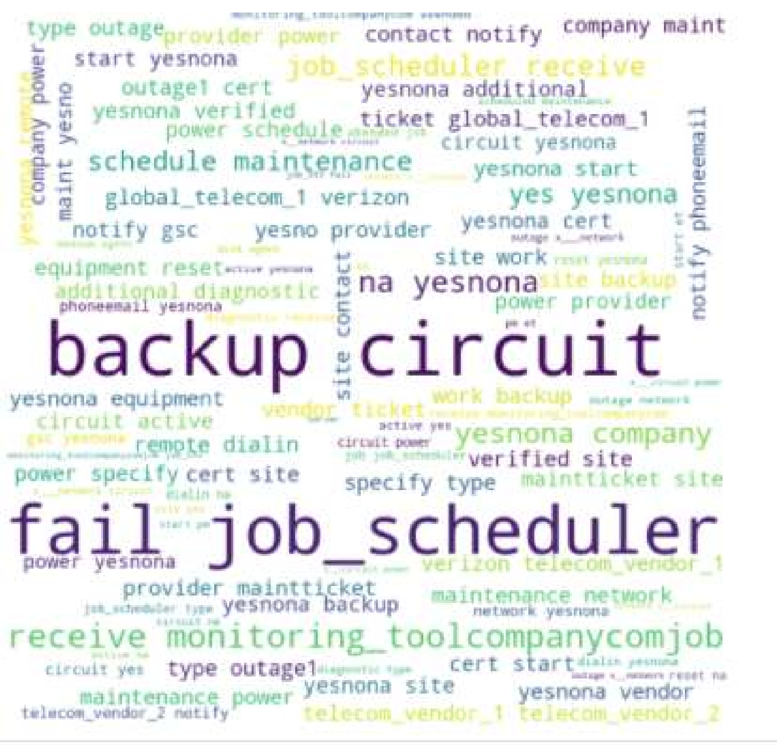
Overall wordC loud



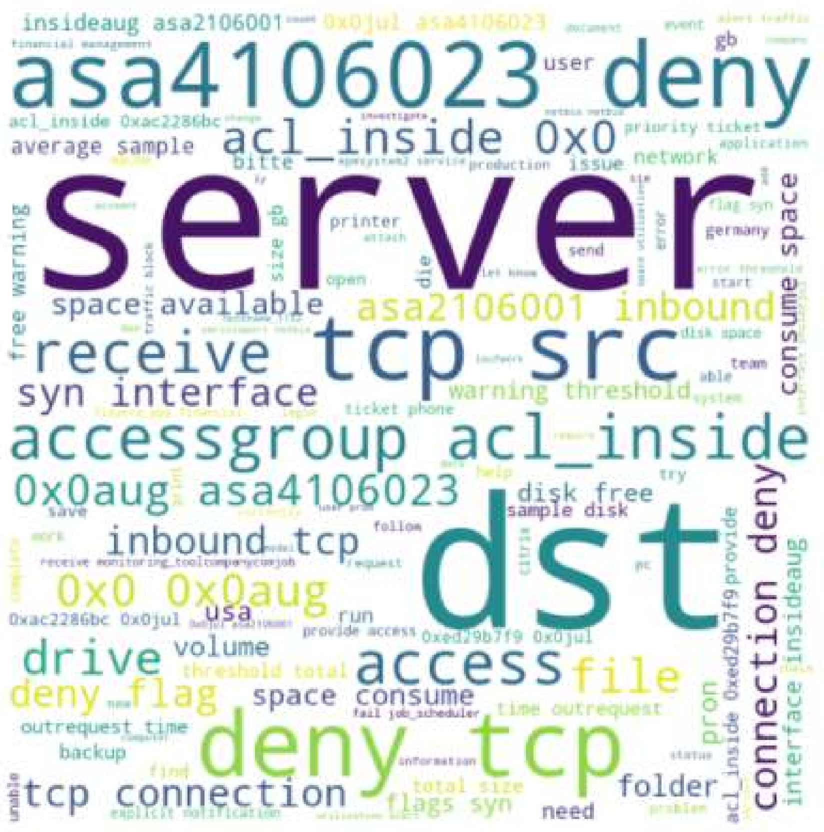
Word Cloud for tickets with Assignment group 'GRP_0'



Word Cloud for tickets with Assignment group 'GRP_8'



Word Cloud for tickets with Assignment group 'GRP_12'



Modelling

As the target class is completely skewed, we resampled the groups. The groups which have less than or equal to 75 number of tickets is combined in to one group as ‘others’. So, the total number of groups reduced from 74 to 21.

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.

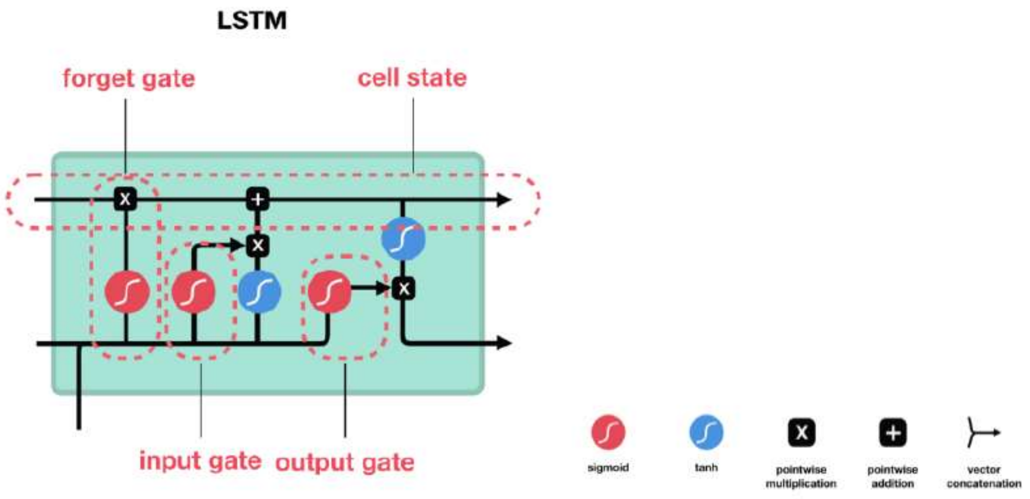
2. GloVe (Global Vectors) Embedding:

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

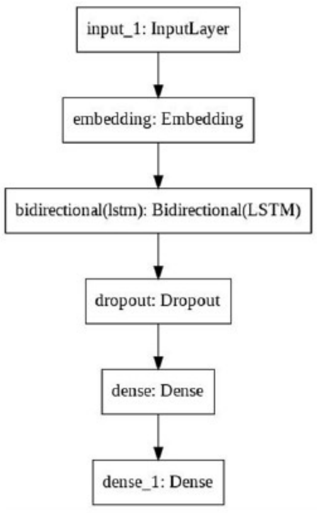
Bi-directional LSTM Model

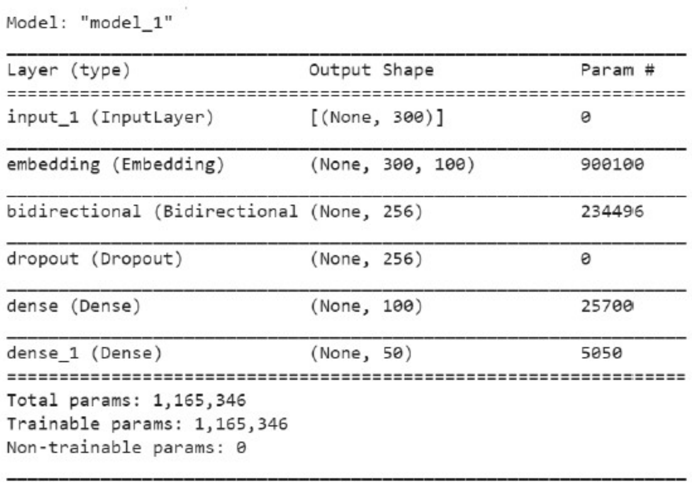
Bidirectional LSTMs are an extension of traditional LSTMs that can improve model performance on classification problems.

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.



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

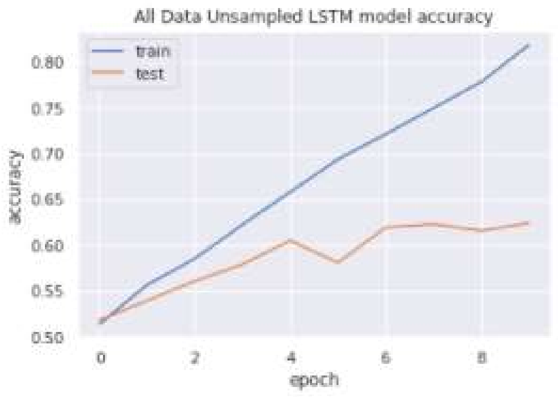


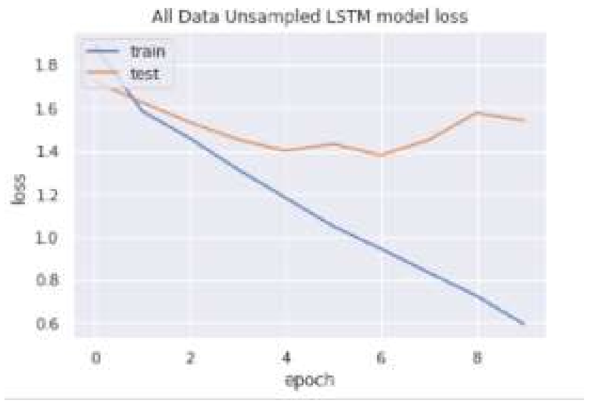


```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 300)]             0
_________________________________________________________________
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
_________________________________________________________________
```

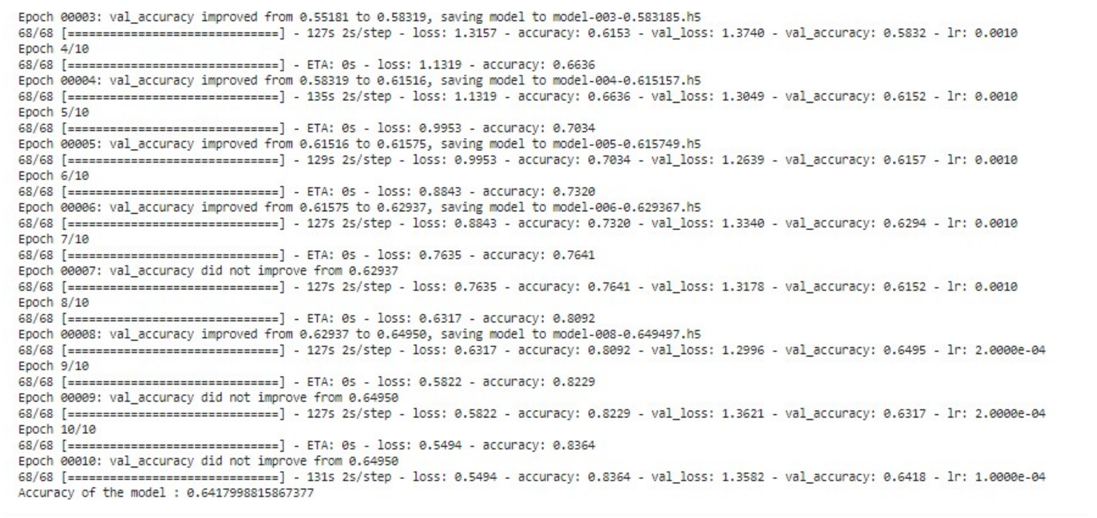
Observations from LSTM Model

Word2Vec



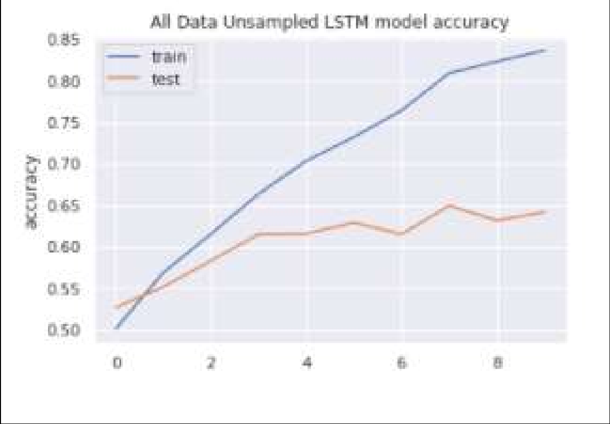


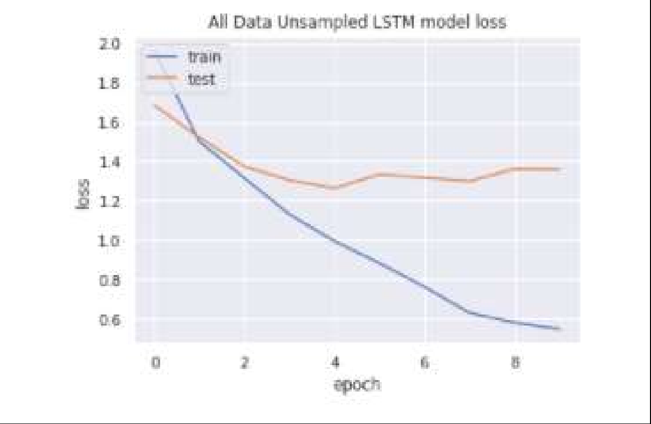
Training Accuracy for LSTM with Word2Vec



```
Epoch 00003: val_accuracy improved from 0.55181 to 0.58319, saving model to model-003-0.583185.h5
68/68 [==============================] - 127s 2s/step - loss: 1.3157 - accuracy: 0.6153 - val_loss: 1.3740 - val_accuracy: 0.5832 - lr: 0.0010
Epoch 4/10
68/68 [==============================] - ETA: 0s - loss: 1.1319 - accuracy: 0.6636
Epoch 00004: val_accuracy improved from 0.58319 to 0.61516, saving model to model-004-0.615157.h5
68/68 [==============================] - 135s 2s/step - loss: 1.1319 - accuracy: 0.6636 - val_loss: 1.3049 - val_accuracy: 0.6152 - lr: 0.0010
Epoch 5/10
68/68 [==============================] - ETA: 0s - loss: 0.9953 - accuracy: 0.7034
Epoch 00005: val_accuracy improved from 0.61516 to 0.61575, saving model to model-005-0.615749.h5
68/68 [==============================] - 129s 2s/step - loss: 0.9953 - accuracy: 0.7034 - val_loss: 1.2639 - val_accuracy: 0.6157 - lr: 0.0010
Epoch 6/10
68/68 [==============================] - ETA: 0s - loss: 0.8843 - accuracy: 0.7320
Epoch 00006: val_accuracy improved from 0.61575 to 0.62937, saving model to model-006-0.629367.h5
68/68 [==============================] - 127s 2s/step - loss: 0.8843 - accuracy: 0.7320 - val_loss: 1.3340 - val_accuracy: 0.6294 - lr: 0.0010
Epoch 7/10
68/68 [==============================] - ETA: 0s - loss: 0.7635 - accuracy: 0.7641
Epoch 00007: val_accuracy did not improve from 0.62937
68/68 [==============================] - 127s 2s/step - loss: 0.7635 - accuracy: 0.7641 - val_loss: 1.3178 - val_accuracy: 0.6152 - lr: 0.0010
Epoch 8/10
68/68 [==============================] - ETA: 0s - loss: 0.6317 - accuracy: 0.8092
Epoch 00008: val_accuracy improved from 0.62937 to 0.64950, saving model to model-008-0.649497.h5
68/68 [==============================] - 127s 2s/step - loss: 0.6317 - accuracy: 0.8092 - val_loss: 1.2996 - val_accuracy: 0.6495 - lr: 2.0000e-04
Epoch 9/10
68/68 [==============================] - ETA: 0s - loss: 0.5822 - accuracy: 0.8229
Epoch 00009: val_accuracy did not improve from 0.64950
68/68 [==============================] - 127s 2s/step - loss: 0.5822 - accuracy: 0.8229 - val_loss: 1.3621 - val_accuracy: 0.6317 - lr: 2.0000e-04
Epoch 10/10
68/68 [==============================] - ETA: 0s - loss: 0.5494 - accuracy: 0.8364
Epoch 00010: val_accuracy did not improve from 0.64950
68/68 [==============================] - 131s 2s/step - loss: 0.5494 - accuracy: 0.8364 - val_loss: 1.3582 - val_accuracy: 0.6418 - lr: 1.0000e-04
Accuracy of the model : 0.6417998815867377
```

Glove Embedding

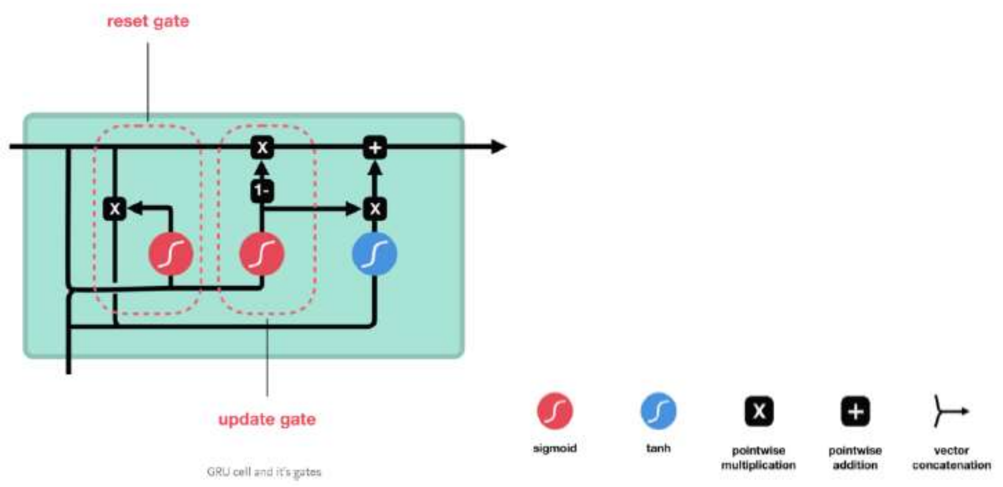




Note: *From Bi-LSTM with both Word2Vec and Glove embedding, we observed better performance with Glove Embedding. So we have proceeded with all other models using Glove Embedded data.*

GRU Model

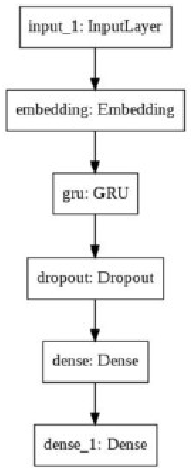
GRU (Gated Recurrent Unit) aims to solve the vanishing gradient problem which comes with a standard recurrent neural network. GRU can also be considered as a variation on the LSTM because both are designed similarly and, in some cases, produce equally excellent results.
To solve the vanishing gradient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

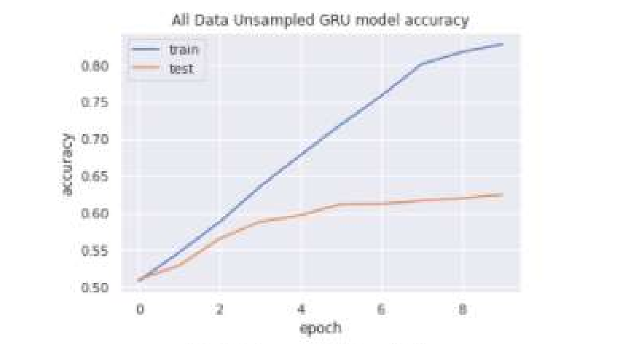
The flow of our GRU model is as shown.



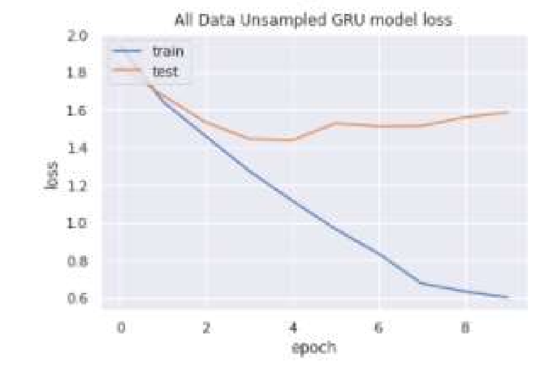
```
Model: "model_11"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_7 (InputLayer)         [(None, 300)]             0
_________________________________________________________________
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

```
Epoch 00004: val_accuracy improved from 0.56483 to 0.58792, saving model to model-004-0.587922.h5
68/68 [==============================] - 54s 795ms/step - loss: 1.2761 - accuracy: 0.6353 - val_loss: 1.4462 - val_accuracy: 0.5879 - lr: 0.0010
Epoch 5/10
68/68 [==============================] - ETA: 0s - loss: 1.1176 - accuracy: 0.6780
Epoch 00005: val_accuracy improved from 0.58792 to 0.59680, saving model to model-005-0.596803.h5
68/68 [==============================] - 54s 795ms/step - loss: 1.1176 - accuracy: 0.6780 - val_loss: 1.4386 - val_accuracy: 0.5968 - lr: 0.0010
Epoch 6/10
68/68 [==============================] - ETA: 0s - loss: 0.9676 - accuracy: 0.7194
Epoch 00006: val_accuracy improved from 0.59680 to 0.61160, saving model to model-006-0.611605.h5
68/68 [==============================] - 54s 795ms/step - loss: 0.9676 - accuracy: 0.7194 - val_loss: 1.5284 - val_accuracy: 0.6116 - lr: 0.0010
Epoch 7/10
68/68 [==============================] - ETA: 0s - loss: 0.8372 - accuracy: 0.7581
Epoch 00007: val_accuracy improved from 0.61160 to 0.61220, saving model to model-007-0.612197.h5
68/68 [==============================] - 54s 791ms/step - loss: 0.8372 - accuracy: 0.7581 - val_loss: 1.5122 - val_accuracy: 0.6122 - lr: 0.0010
Epoch 8/10
68/68 [==============================] - ETA: 0s - loss: 0.6781 - accuracy: 0.8013
Epoch 00008: val_accuracy improved from 0.61220 to 0.61634, saving model to model-008-0.616341.h5
68/68 [==============================] - 57s 837ms/step - loss: 0.6781 - accuracy: 0.8013 - val_loss: 1.5141 - val_accuracy: 0.6163 - lr: 2.0000e-04
Epoch 9/10
68/68 [==============================] - ETA: 0s - loss: 0.6359 - accuracy: 0.8177
Epoch 00009: val_accuracy improved from 0.61634 to 0.61989, saving model to model-009-0.619893.h5
68/68 [==============================] - 54s 793ms/step - loss: 0.6359 - accuracy: 0.8177 - val_loss: 1.5605 - val_accuracy: 0.6199 - lr: 2.0000e-04
Epoch 10/10
68/68 [==============================] - ETA: 0s - loss: 0.6036 - accuracy: 0.8284
Epoch 00010: val_accuracy improved from 0.61989 to 0.62463, saving model to model-010-0.624630.h5
68/68 [==============================] - 54s 795ms/step - loss: 0.6036 - accuracy: 0.8284 - val_loss: 1.5862 - val_accuracy: 0.6246 - lr: 1.0000e-04
Accuracy of the model : 0.6246299585553582
```



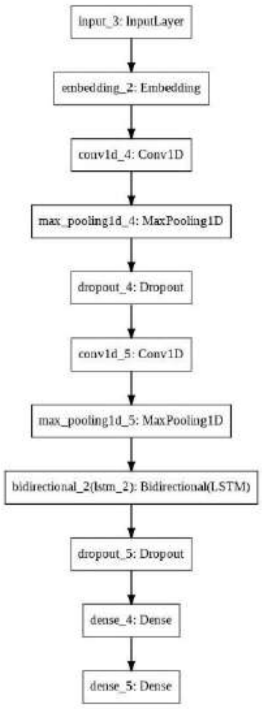
All Data Unsampled GRU model loss

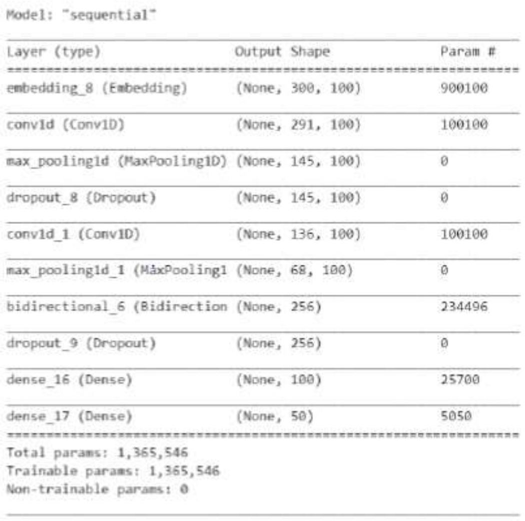


RNN Model

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.

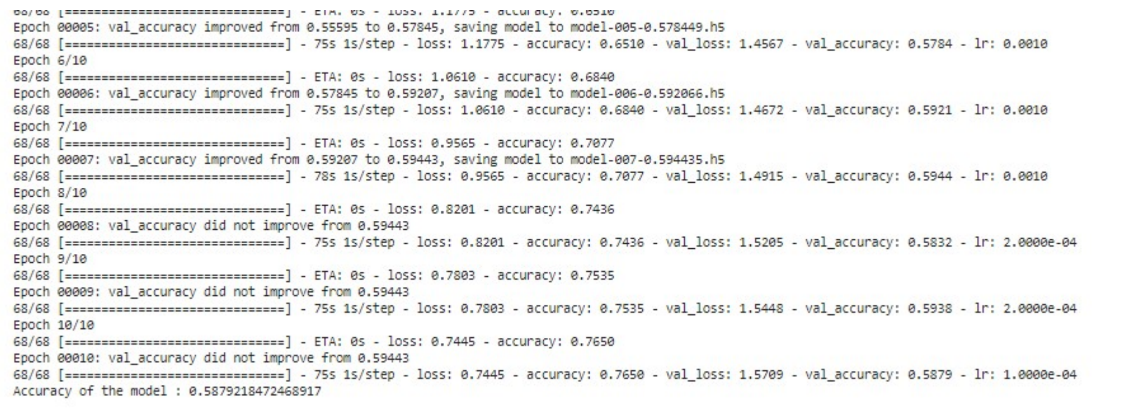
The flow of our RNN model is as shown.



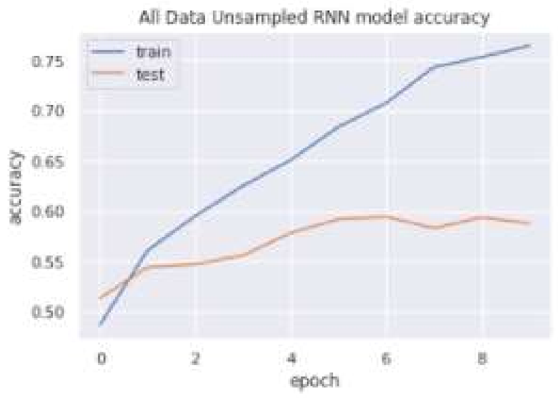


```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_8 (Embedding)      (None, 300, 100)          900100
_________________________________________________________________
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



```
68/68 [==============================] - ETA: 0s - loss: 1.1775 - accuracy: 0.6510
Epoch 00005: val_accuracy improved from 0.55595 to 0.57845, saving model to model-005-0.578449.h5
68/68 [==============================] - 75s 1s/step - loss: 1.1775 - accuracy: 0.6510 - val_loss: 1.4567 - val_accuracy: 0.5784 - lr: 0.0010
Epoch 6/10
68/68 [==============================] - ETA: 0s - loss: 1.0610 - accuracy: 0.6840
Epoch 00006: val_accuracy improved from 0.57845 to 0.59207, saving model to model-006-0.592066.h5
68/68 [==============================] - 75s 1s/step - loss: 1.0610 - accuracy: 0.6840 - val_loss: 1.4672 - val_accuracy: 0.5921 - lr: 0.0010
Epoch 7/10
68/68 [==============================] - ETA: 0s - loss: 0.9565 - accuracy: 0.7077
Epoch 00007: val_accuracy improved from 0.59207 to 0.59443, saving model to model-007-0.594435.h5
68/68 [==============================] - 78s 1s/step - loss: 0.9565 - accuracy: 0.7077 - val_loss: 1.4915 - val_accuracy: 0.5944 - lr: 0.0010
Epoch 8/10
68/68 [==============================] - ETA: 0s - loss: 0.8201 - accuracy: 0.7436
Epoch 00008: val_accuracy did not improve from 0.59443
68/68 [==============================] - 75s 1s/step - loss: 0.8201 - accuracy: 0.7436 - val_loss: 1.5205 - val_accuracy: 0.5832 - lr: 2.0000e-04
Epoch 9/10
68/68 [==============================] - ETA: 0s - loss: 0.7803 - accuracy: 0.7535
Epoch 00009: val_accuracy did not improve from 0.59443
68/68 [==============================] - 75s 1s/step - loss: 0.7803 - accuracy: 0.7535 - val_loss: 1.5448 - val_accuracy: 0.5938 - lr: 2.0000e-04
Epoch 10/10
68/68 [==============================] - ETA: 0s - loss: 0.7445 - accuracy: 0.7650
Epoch 00010: val_accuracy did not improve from 0.59443
68/68 [==============================] - 75s 1s/step - loss: 0.7445 - accuracy: 0.7650 - val_loss: 1.5709 - val_accuracy: 0.5879 - lr: 1.0000e-04
Accuracy of the model : 0.5879218472468917
```



Traditional ML Models

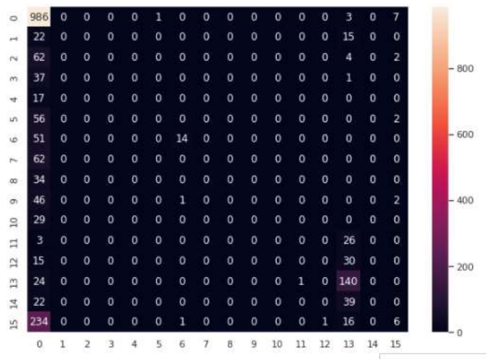
Logistic regression

Logistic regression is basically a supervised classification algorithm. In a classification problem, the target variable (or output), y, can take only discrete values for a given set of features(or inputs), X. Logistic regression IS a regression model. The model builds a regression model to predict the probability that a given data entry belongs to the category numbered as “1”.

Classification report:

```
              precision    recall  f1-score   support

           0       0.58      0.99      0.73       997
           1       0.00      0.00      0.00        37
           2       0.00      0.00      0.00        68
           3       0.00      0.00      0.00        38
           4       0.00      0.00      0.00        17
           5       0.00      0.00      0.00        58
           6       0.88      0.22      0.35        65
           7       0.00     .0.00      0.00        62
           8       0.00      0.00      0.00        34
           9       0.00      0.00      0.00        49
          10       0.00      0.00      0.00        29
          11       0.00      0.00      0.00        29
          12       0.00      0.00      0.00        45
          13       0.51      0.85      0.64       165
          14       0.00      0.00      0.00        61
          15       0.32      0.02      0.04       258

    accuracy                           0.57      2012
   macro avg       0.14      0.13      0.11      2012
weighted avg       0.40      0.57      0.43      2012
```

Heatmap



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	986	0	0	0	0	1	0	0	0	0	0	0	0	3	0	7
1	22	0	0	0	0	0	0	0	0	0	0	0	0	15	0	0
2	62	0	0	0	0	0	0	0	0	0	0	0	0	4	0	2
3	37	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
4	17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	56	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
6	51	0	0	0	0	0	14	0	0	0	0	0	0	0	0	0
7	62	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	46	0	0	0	0	0	1	0	0	0	0	0	0	0	0	2
10	29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	3	0	0	0	0	0	0	0	0	0	0	0	0	26	0	0
12	15	0	0	0	0	0	0	0	0	0	0	0	0	30	0	0
13	24	0	0	0	0	0	0	0	0	0	0	1	0	140	0	0
14	22	0	0	0	0	0	0	0	0	0	0	0	0	39	0	0
15	234	0	0	0	0	0	1	0	0	0	0	0	1	16	0	6

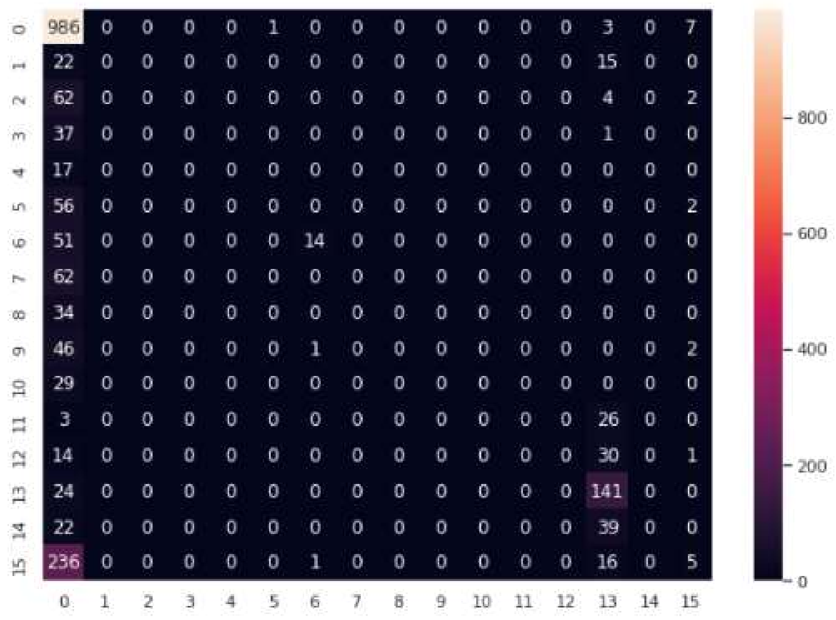
Linear SVC

The objective of a Linear SVC (Support Vector Classifier) is to fit to the data you provide, returning a "best fit" hyperplane that divides, or categorizes, your data. From there, after getting the hyperplane, you can then feed some features to your classifier to see what the "predicted" class is.

Classification report:

```
              precision    recall  f1-score   support

           0       0.58      0.99      0.73       997
           1       0.00      0.00      0.00        37
           2       0.00      0.00      0.00        68
           3       0.00      0.00      0.00        38
           4       0.00      0.00      0.00        17
           5       0.00      0.00      0.00        58
           6       0.88      0.22      0.35        65
           7       0.00      0.00      0.00        62
           8       0.00      0.00      0.00        34
           9       0.00      0.00      0.00        49
          10       0.00      0.00      0.00        29
          11       0.00      0.00      0.00        29
          12       0.00      0.00      0.00        45
          13       0.51      0.85      0.64       165
          14       0.00      0.00      0.00        61
          15       0.26      0.02      0.04       258

    accuracy                           0.57      2012
   macro avg       0.14      0.13      0.11      2012
weighted avg       0.39      0.57      0.43      2012
```

Heatmap



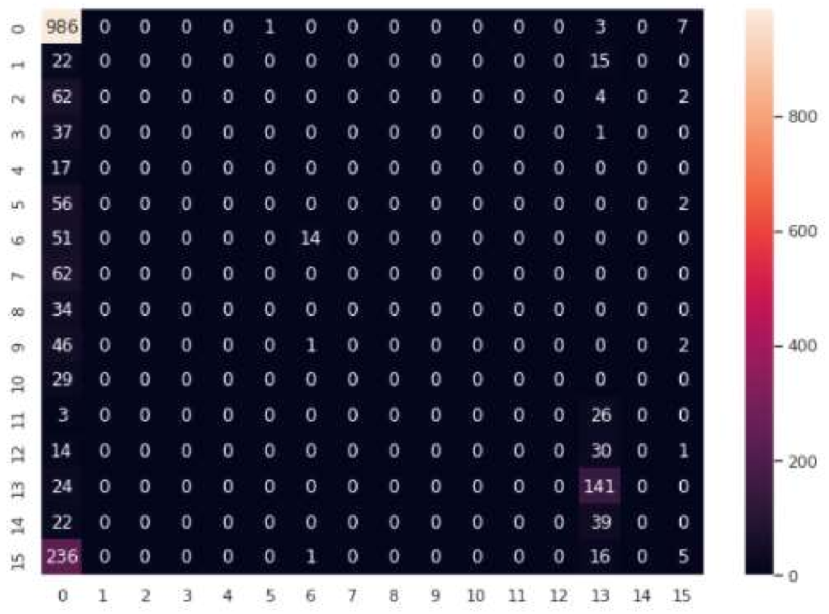
Bernoulli NB Classifier

Naive Bayes is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set. There is not a single algorithm for training such classifiers, but a family of algorithms based on a common principle: all naive Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable

Classification report:

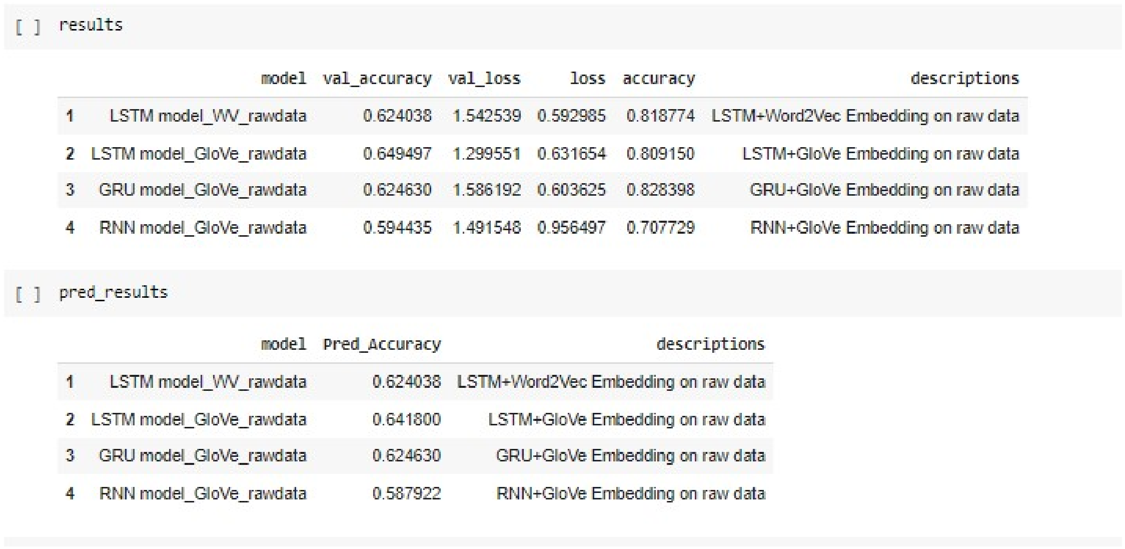
```
              precision    recall  f1-score   support

           0       0.57      0.98      0.72       997
           1       0.00      0.00      0.00        37
           2       0.00      0.00      0.00        68
           3       0.00      0.00      0.00        38
           4       0.00      0.00      0.00        17
           5       0.00      0.00      0.00        58
           6       0.46      0.25      0.32        65
           7       0.00      0.00      0.00        62
           8       0.00      0.00      0.00        34
           9       0.00      0.00      0.00        49
          10       0.00      0.00      0.00        29
          11       0.00      0.00      0.00        29
          12       0.00      0.00      0.00        45
          13       0.32      0.07      0.12       165
          14       0.18      0.56      0.27        61
          15       0.28      0.02      0.04       258

    accuracy                           0.52      2012
   macro avg       0.11      0.12      0.09      2012
weighted avg       0.36      0.52      0.39      2012
```

Heatmap



4. Model evaluation

Accuracy for each model



```
[ ] results
```

	model	val_accuracy	val_loss	loss	accuracy	descriptions
1	LSTM model_WV_rawdata	0.624038	1.542539	0.592985	0.818774	LSTM+Word2Vec Embedding on raw data
2	LSTM model_GloVe_rawdata	0.649497	1.299551	0.631654	0.809150	LSTM+GloVe Embedding on raw data
3	GRU model_GloVe_rawdata	0.624630	1.586192	0.603625	0.828398	GRU+GloVe Embedding on raw data
4	RNN model_GloVe_rawdata	0.594435	1.491548	0.956497	0.707729	RNN+GloVe Embedding on raw data

```
[ ] pred_results
```

	model	Pred_Accuracy	descriptions
1	LSTM model_WV_rawdata	0.624038	LSTM+Word2Vec Embedding on raw data
2	LSTM model_GloVe_rawdata	0.641800	LSTM+GloVe Embedding on raw data
3	GRU model_GloVe_rawdata	0.624630	GRU+GloVe Embedding on raw data
4	RNN model_GloVe_rawdata	0.587922	RNN+GloVe Embedding on raw data

Traditional Models Accuracy

1	**Logistic Regression**	**0.5695825049701789**
2	**Linear SVC**	**0.5695825049701789**
3	**Bernoulli Classifier**	**0.5208747514910537**

From the predicted accuracies, we could see that the LSTM and GRU models with the resampled Augmented dataset is performing well with 82% and 83% 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.

5. Comparison to benchmark

From the given problem description, we could see that the existing system is able to assign 57% 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. From the prediction results we see that the Bi-directional GRU model based on the resampled data is able to achieve an accuracy of 83% which is above our benchmark.

6. Implications

Although this model can classify the IT tickets with 83% accuracy, to achieve better accuracy in the real world it would be good if the business can collect additional data around 500 records for each group.

7. Limitations

As part of Data pre-processing, we had grouped all assignment groups with less than 75 entries as one group (other) which had reduced the Target class from 74 to 21groups. While applying this model in the real world there could be additional intervention required to classify the tickets if it has been classified as other by our model. Since the number of elements reported under other are less, we expect this intervention to be done less often.

8. 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 analysed. 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.