Capstone Project - AIML
(2019-20)

Automatic Ticket Assignment

Mentored By: Naga Pavan Kumar Kalepu

Prepared By:

- Shivalik Chhabra
- Nishant Joshi
- Srikanth Rammnath
- Mansi Garg

Summary of problem statement, data and findings

PROBLEM STATEMENT

In any IT industry, Incident (an unplanned interruption to an IT service or reduction in the quality of an IT service that affects the Users and the Business) Management plays an important role in delivering quality support to customers. The main goal of this 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. 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 manual assignment of these incidents might have below disadvantages:

- Time consuming and requires human efforts
- Increases human errors and resource consumption, as it is carried out ineffectively because of the misaddressing.
- Increases the response and resolution times which result in user satisfaction deterioration / poor customer service

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.

OBJECTIVE

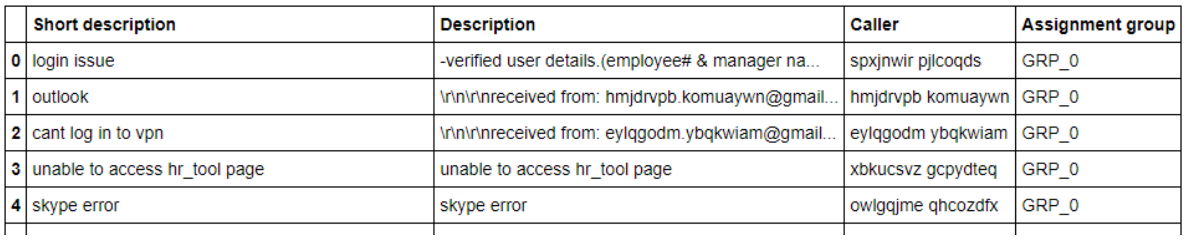
The goal is to build a classifier that can classify the tickets by analysing text.

DATA DESCRIPTION

The given dataset consists of the following four attributes:

1. Short Description (a summary of the issue faced by the user)
2.Description (detailed description of the issue)
3.Caller (ID of the caller)
4.Assignment group (GRP_0 ~ GRP_73 i.e., total 74 classes of Assignment group) – Target class

SAMPLE DATA



	Short description	Description	Caller	Assignment group
0	login issue	-verified user details.(employee# & manager na...	spxjnwir pjlcoqds	GRP_0
1	outlook	\r\n\r\nreceived from: hmjdrvpb.komuaywn@gmail...	hmjdrvpb komuaywn	GRP_0
2	cant log in to vpn	\r\n\r\nreceived from: eylqgodm.ybqkwiam@gmail...	eylqgodm ybqkwiam	GRP_0
3	unable to access hr_tool page	unable to access hr_tool page	xbkucsvz gcpydteq	GRP_0
4	skype error	skype error	owlgqjme qhcozdfx	GRP_0

FINDINGS

Followings are the general observation from the given dataset:

- Caller ID are present in a random manner (may not be useful for training data)
- Languages other than English for example- German, etc. are also present in the dataset
- Non-English languages are also found in the data
- Email/chat format with symbols in description
- Hyperlinks and URLS are found in the description
- Blank records are present in either short description or description
- Few descriptions are replica of the short description
- Few words were combined together
- Spelling mistakes and typo errors are found

CURRENT PROCESS

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.

GOAL

The goal here is to create NLP based classifier that could automatically classify any ticket raised by analysing ticket description to the suitable Assignment group, this could be integrated with any ticket management service like Service Now

Based on the ticket description our model would assign a probability of it to being assigned to one of the 74 Groups.

This project intends to reduce the manual effort of IT support teams by automating the process of ticket assignment.

Summary of Data

Data Source

- Details about the data and dataset files are given in below link,
    - https://drive.google.com/open?id=1OZNJm81JXucV3HmZroMq6qCT2m7ez7IJ
- The data set contains 4 string columns

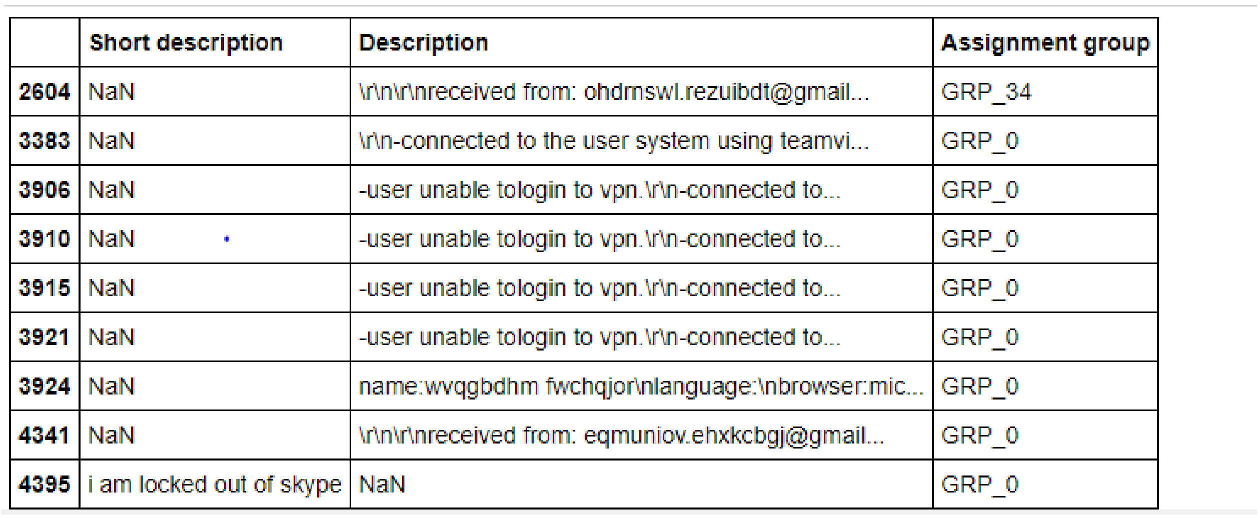
Column	Description	Data type
Short description	Short description for problem for which the incident is being raised	8492 non-null object
Description	Detailed description of the problem for which the incident is being raised	8499 non-null object
Caller	Masked user name	8500 non-null object
Assignment Group	IT Support Group to whichthe incident has to be assigned	8500 non-null object

- The dataset is divided into two parts, features and the target classes.
- For given dataset, features are Short description, Description and Caller .
- For given dataset, the class variable name is Assignment group.
- There are totally 8500 rows
- There seems to be missing values in Short description and Description columns, which needs to be looked into and handled.
    - There are 8 null/missing values present in the Short description and 1 null/missing values present in the description column
- Caller columns mainly contain the details of the user who raised the incident and is of not much use in our analysis and can be dropped.
- "Short Description" and "Description" could be concatenated into a single column, so that we don’t miss information about the tickets.

Summary of Approach to EDA and Pre-processing

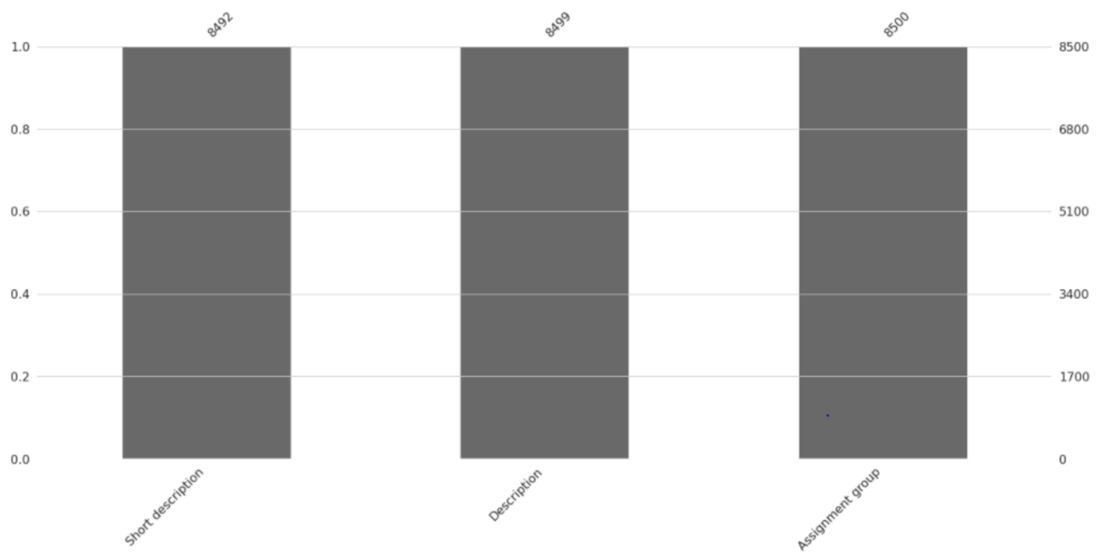
Step-by-step walk through the solution with the observations

- Loaded the input csv file into pandas' data frame.
- EDA has been performed on the dataset and following are the observations from that:
    - All columns are of type object containing textual information.
    - Assignment group is our predictor / target column with multiple classes. So, this is a Multiclass Classification problem.
    - There are **8 null/missing values** present in the Short description and **1 null/missing values** present in the description column.



	Short description	Description	Assignment group
2604	NaN	\r\n\r\nreceived from: ohdmswl.rezuibdt@gmail...	GRP_34
3383	NaN	\r\n-connected to the user system using teamvi...	GRP_0
3906	NaN	-user unable tologin to vpn.\r\n-connected to...	GRP_0
3910	NaN	-user unable tologin to vpn.\r\n-connected to...	GRP_0
3915	NaN	-user unable tologin to vpn.\r\n-connected to...	GRP_0
3921	NaN	-user unable tologin to vpn.\r\n-connected to...	GRP_0
3924	NaN	name:wvqgbdhm fwchqjor\nlanguage:\nbrowser:mic...	GRP_0
4341	NaN	\r\n\r\nreceived from: eqmuniov.ehxkcbgj@gmail...	GRP_0
4395	i am locked out of skype	NaN	GRP_0

Visually the number of missing values are as shown below:

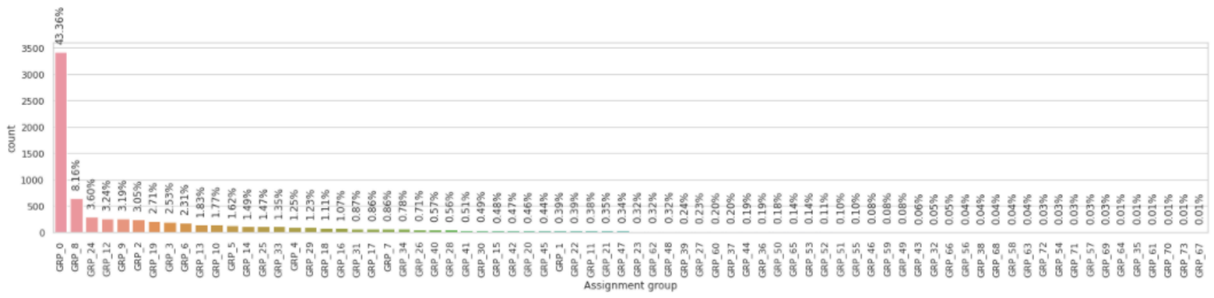


- Dropped the duplicates entries of the incidents. Thus, the size of the dataset gets reduced to (7909,4)
- Dropped the caller attribute as the data was not found to be useful for analysis
- Replaced Null values in Short description & description with space.
- Merged Short Description & Description fields for analysis

OBSERVATIONS REGARDING TARGET CLASS

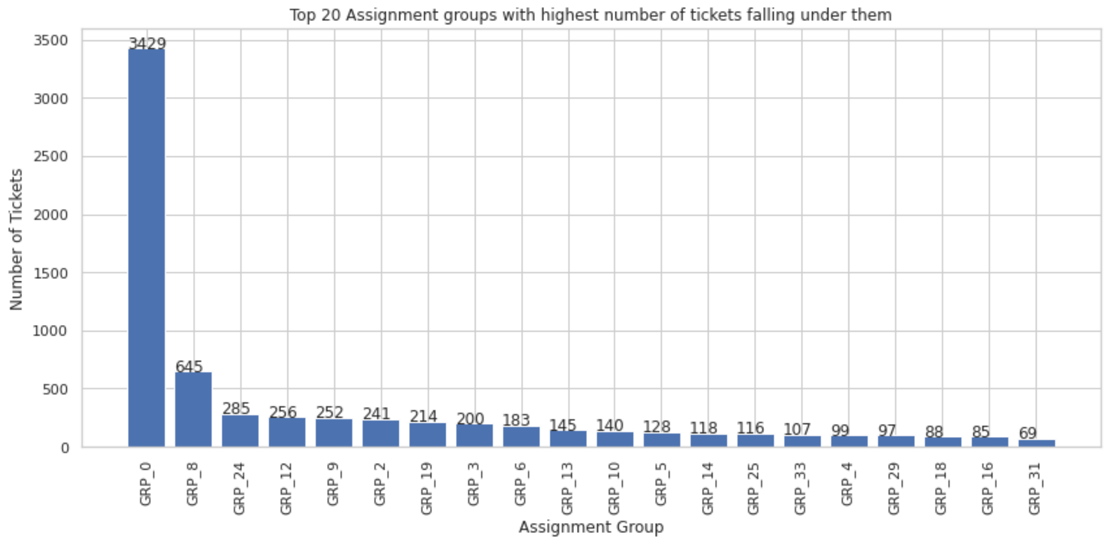
- A large number of entries belonged to GRP_0 (mounting to 3429 which account for ~ more than 40% of the data)

Distribution of Target class for all groups

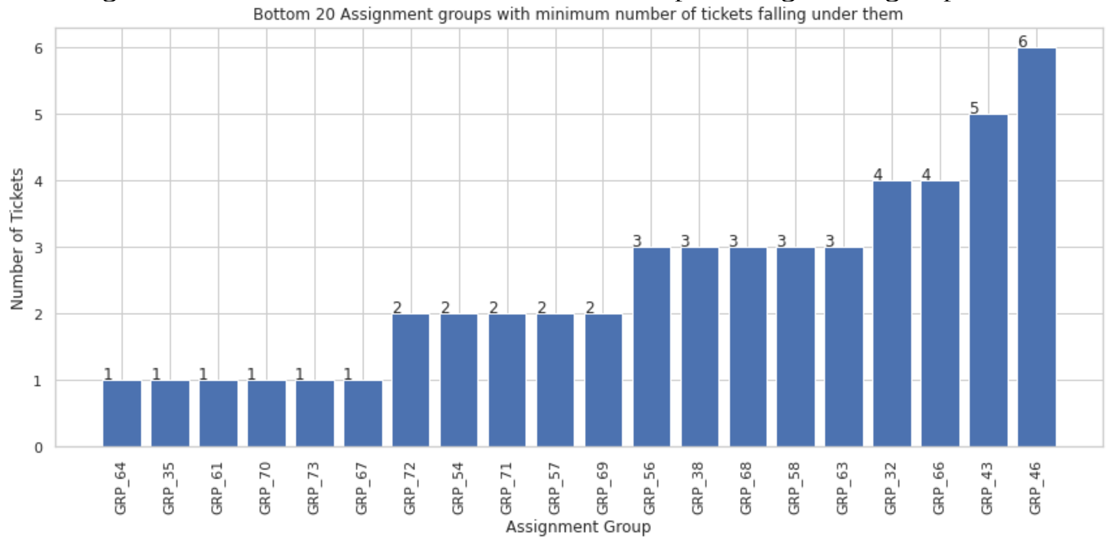


- The Target class distribution is extremely skewed
- The data is too much biased towards a single group and seems to be highly imbalanced, with majority of incidents are from Group 0 followed by Group 8 , 24 , 12 , 9 , 2 and so on
- There are few classes which just have less than 10 incidents per class and even classes with just 1 or 2 incidents (samples), need to see if we can drop those rows due to the lack of samples representing those classes. They might not be of much help as a predictor

- Top 20 Assignment groups having the highest number of tickets for training the data.



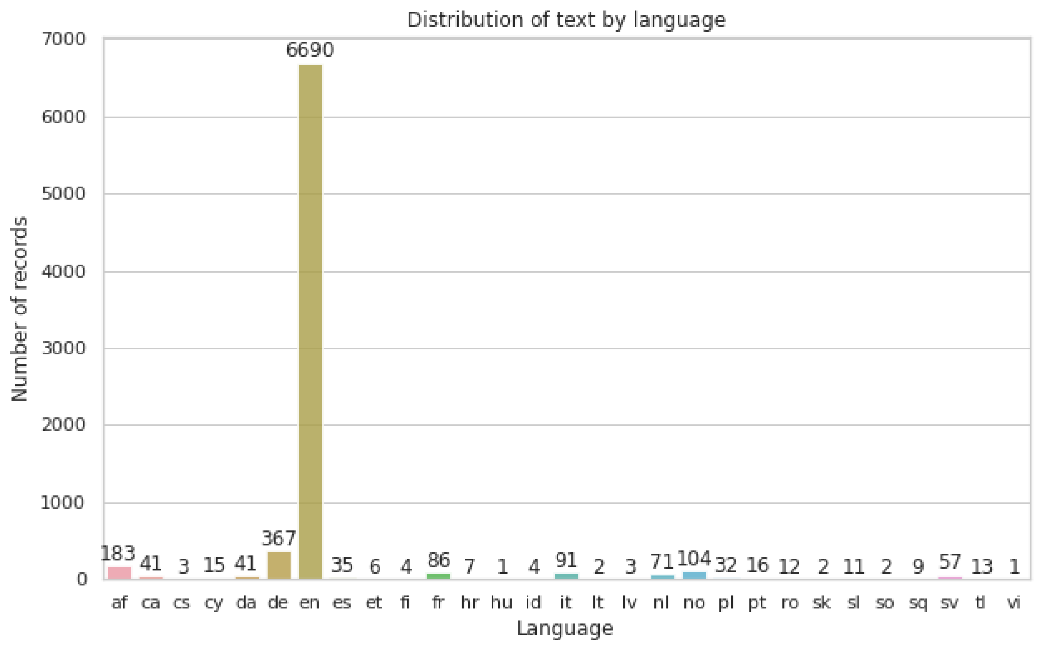
- Following are the Tickets with less number of tickets per Assignment groups.



DATA PRE-PROCESSING

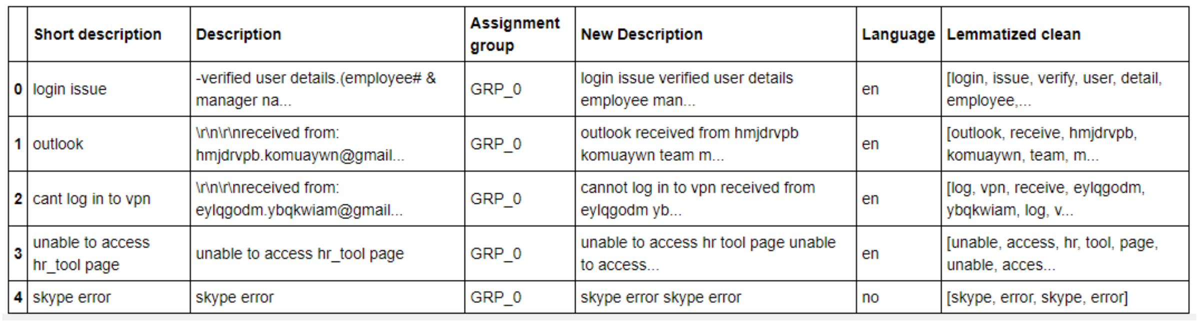
Below steps have been performed for initial pre-processing and clean-up of data:

- Replaced the gibberish text using **FTFY**
- 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.
- There were quite few entries with languages different from English.



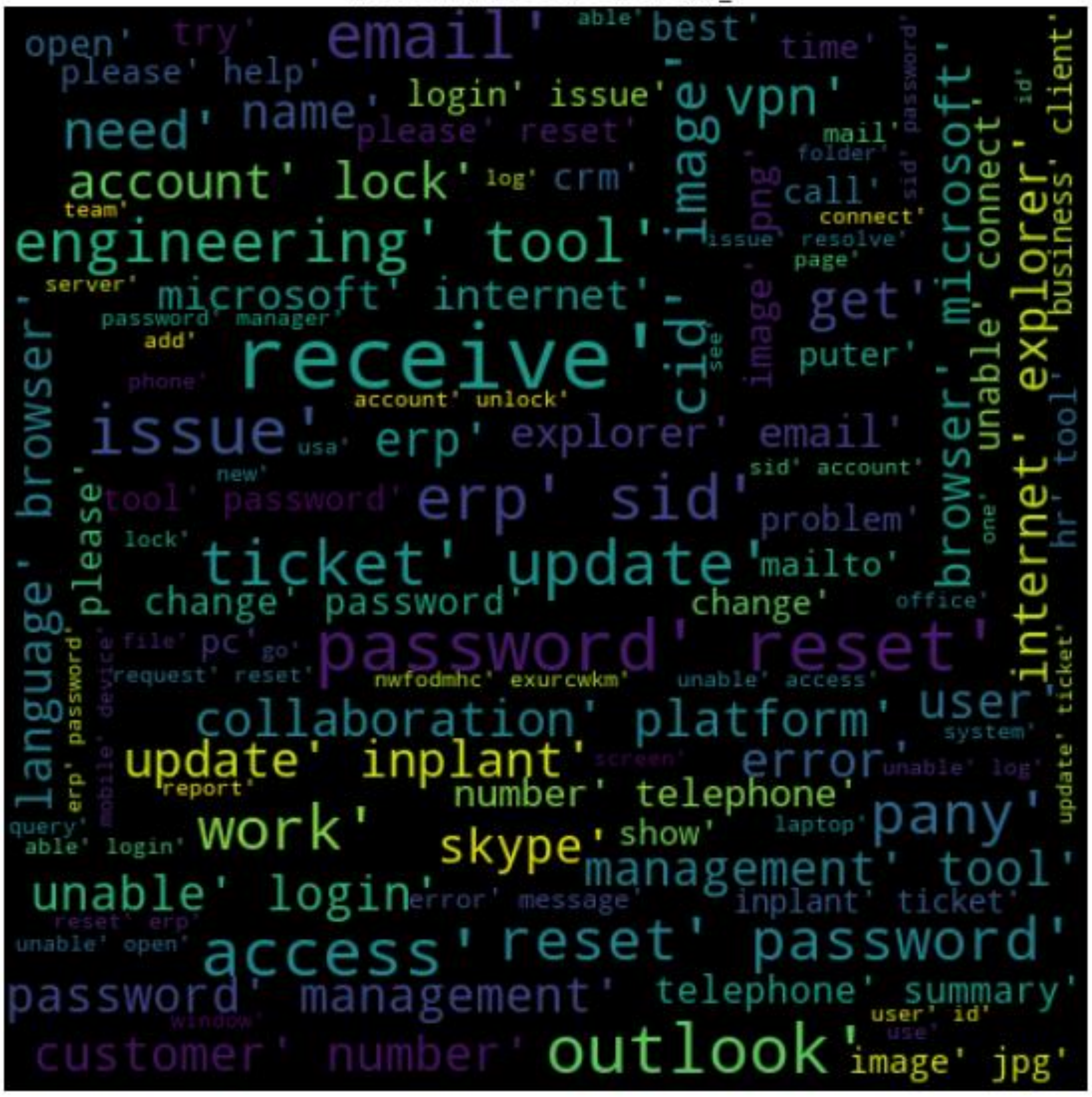
We can see that most of the tickets are in English, followed by tickets in German language.

- **Tokenization** of merged data
- **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.



	Short description	Description	Assignment group	New Description	Language	Lemmatized clean
0	login issue	-verified user details.(employee# & manager na...	GRP_0	login issue verified user details employee man...	en	[login, issue, verify, user, detail, employee,...
1	outlook	\r\n\r\nreceived from: hmjdrvpb.komuaywn@gmail...	GRP_0	outlook received from hmjdrvpb komuaywn team m...	en	[outlook, receive, hmjdrvpb, komuaywn, team, m...
2	cant log in to vpn	\r\n\r\nreceived from: eylqgodm.ybqkwiam@gmail...	GRP_0	cannot log in to vpn received from eylqgodm yb...	en	[log, vpn, receive, eylqgodm, ybqkwiam, log, v...
3	unable to access hr_tool page	unable to access hr_tool page	GRP_0	unable to access hr tool page unable to access...	en	[unable, access, hr, tool, page, unable, acces...
4	skype error	skype error	GRP_0	skype error skype error	no	[skype, error, skype, error]

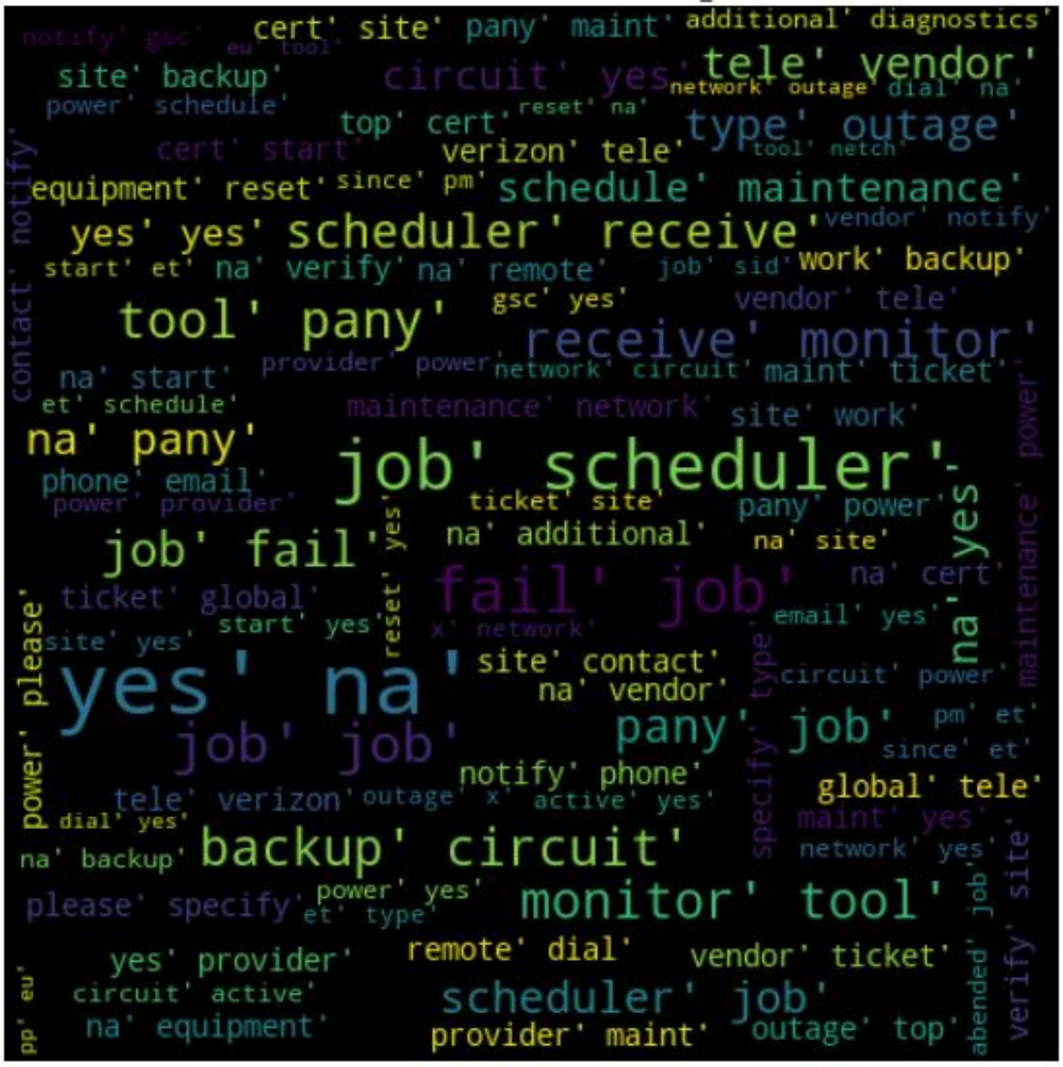
- **WordCloud** created for all available 50 groups to have more information specific to Assignment groups



Most common 50 words of GRP_0

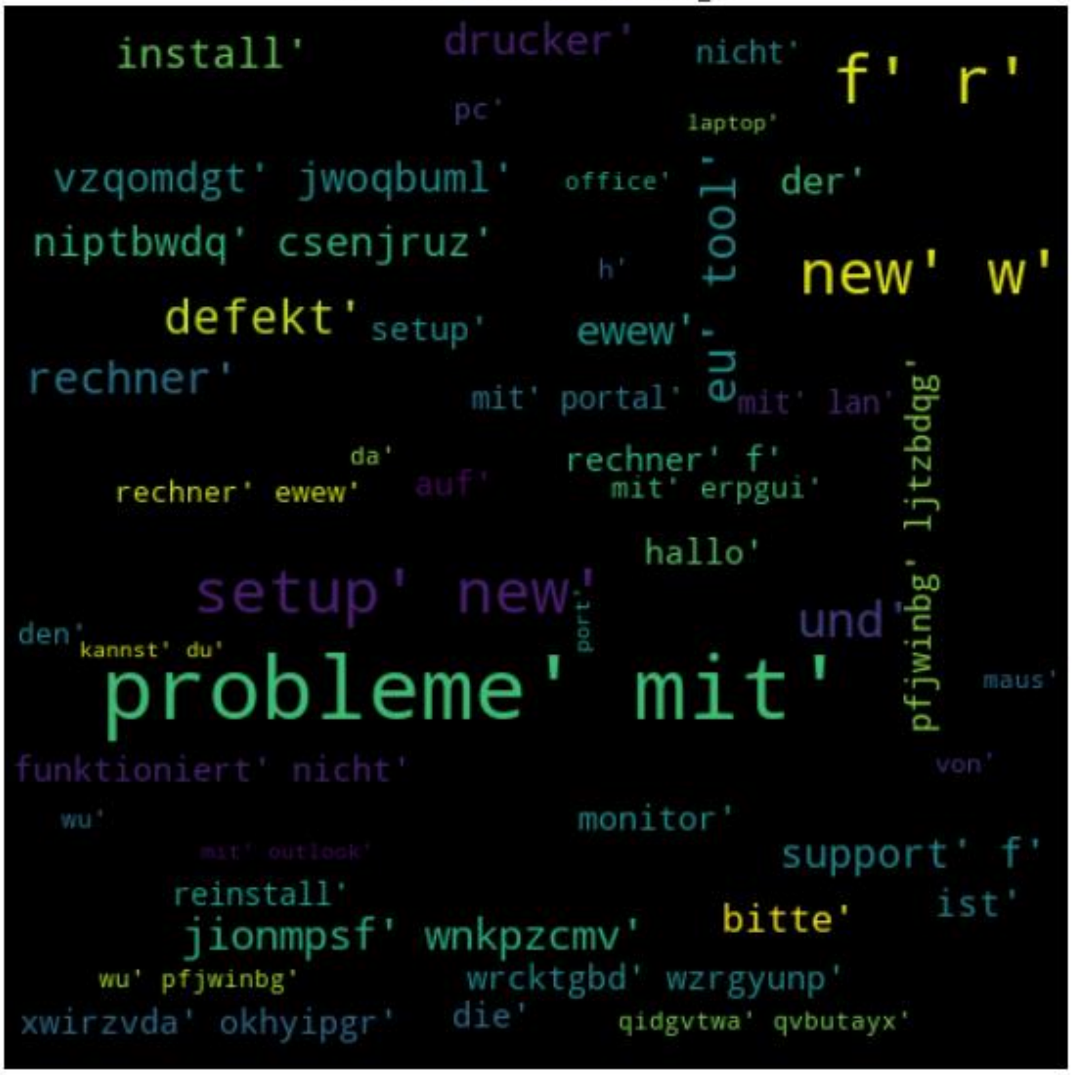
Analysis on GRP_0 which is the most frequent group to assign a ticket to reveals that this group deals with mostly the maintenance problems such as *password, receive, reset* ,outlook, *account lock* , *login issue* , *ticket update* etc.

Most common 50 words of GRP_8



GRP_8 seems to have tickets related *to scheduler, job failures, monitoring tool* etc.

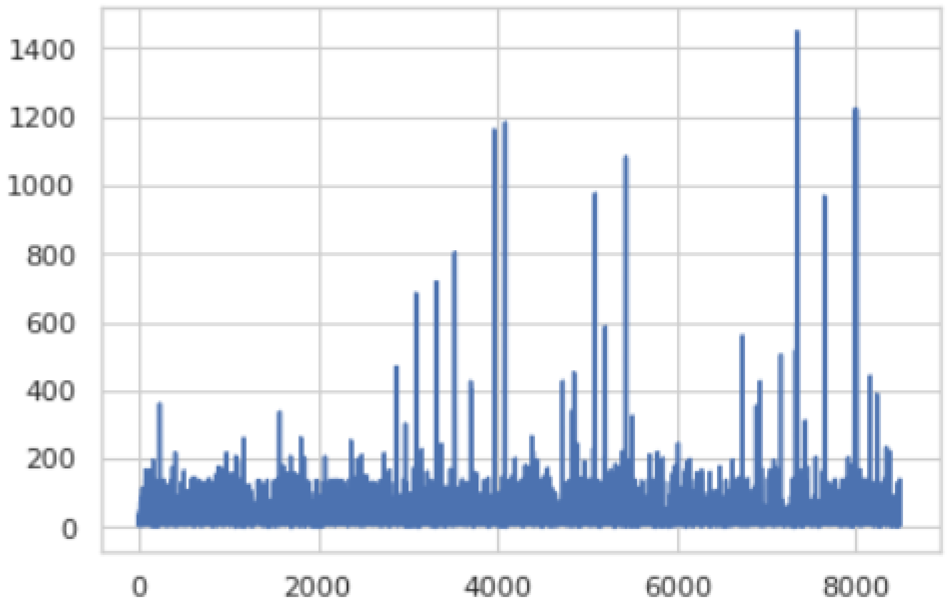
Most common 50 words of GRP_24



GRP_24 - Tickets are mainly in German.

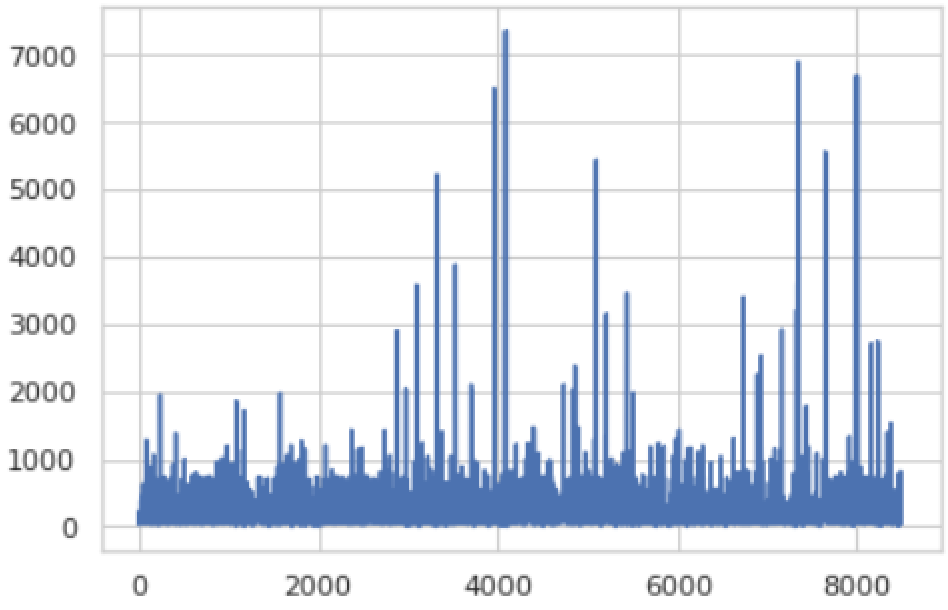
- **Word Count Distribution**

The distribution of the words in the concatenated description attribute is as shown below:

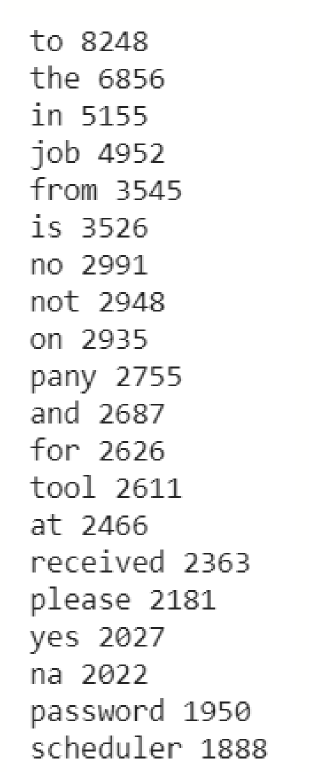


- **Distribution Of The Length**

The distribution of the length of the concatenated description attribute has been shown below:

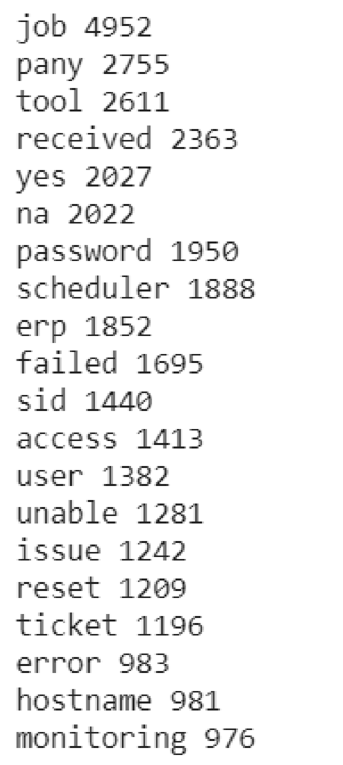


- The distribution of **top unigrams before removing stop words** for Concatenated Description is as shown below:



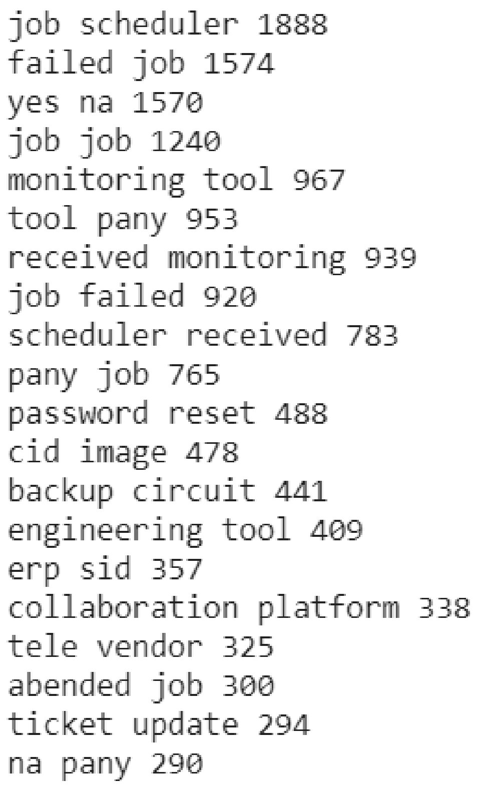
```
to 8248
the 6856
in 5155
job 4952
from 3545
is 3526
no 2991
not 2948
on 2935
pany 2755
and 2687
for 2626
tool 2611
at 2466
received 2363
please 2181
yes 2027
na 2022
password 1950
scheduler 1888
```

- The distribution of **top unigrams after removing stop words** for concatenated Description is as shown below:



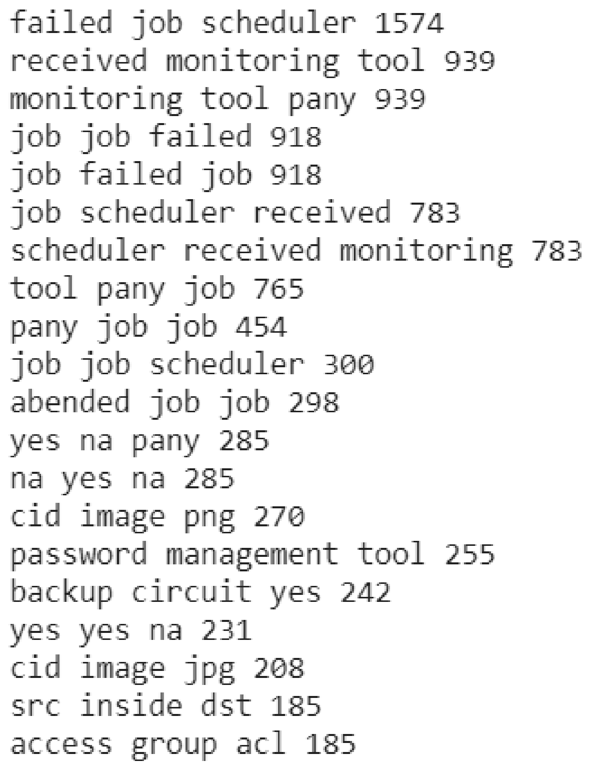
```
job 4952
pany 2755
tool 2611
received 2363
yes 2027
na 2022
password 1950
scheduler 1888
erp 1852
failed 1695
sid 1440
access 1413
user 1382
unable 1281
issue 1242
reset 1209
ticket 1196
error 983
hostname 981
monitoring 976
```

- The distribution of **top bigrams** after removing stop words for Concatenated Description is as shown below:



```
job scheduler 1888
failed job 1574
yes na 1570
job job 1240
monitoring tool 967
tool pany 953
received monitoring 939
job failed 920
scheduler received 783
pany job 765
password reset 488
cid image 478
backup circuit 441
engineering tool 409
erp sid 357
collaboration platform 338
tele vendor 325
abended job 300
ticket update 294
na pany 290
```

- The distribution of **top trigrams** after removing stop words for Concatenated Description is as shown below:



```
failed job scheduler 1574
received monitoring tool 939
monitoring tool pany 939
job job failed 918
job failed job 918
job scheduler received 783
scheduler received monitoring 783
tool pany job 765
pany job job 454
job job scheduler 300
abended job job 298
yes na pany 285
na yes na 285
cid image png 270
password management tool 255
backup circuit yes 242
yes yes na 231
cid image jpg 208
src inside dst 185
access group acl 185
```

Model Selection

Modelling

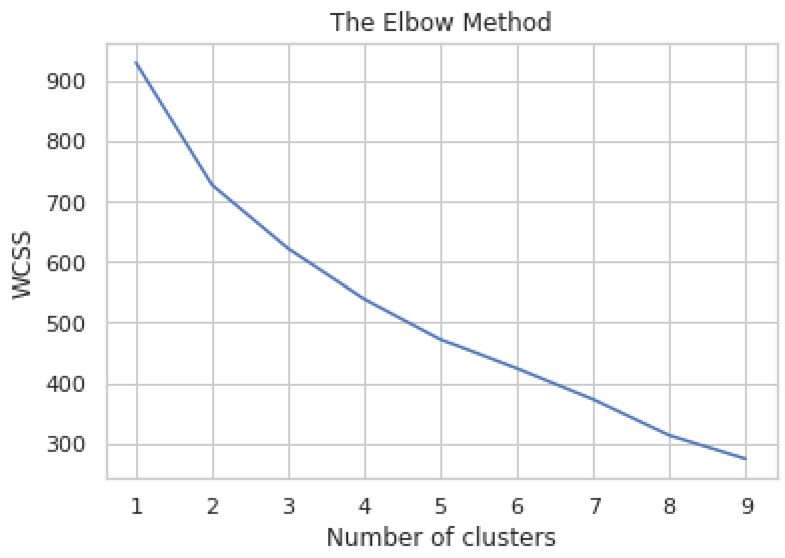
As the target class is completely skewed, various models have been tried with the below set of datasets to compare each performance. Datasets used for each model are:

- Raw data with the target class without any sampling
- Resampled data where all the Grp_0 has been subdivided on the basis of cluster analysis and clubbed with the rest of the dataset.
- Resampled data where all the groups(except Grp_0) has been are sampled with a count of 645 and clubbed with the subdivided dataset for Grp_0.

K-Means Clustering

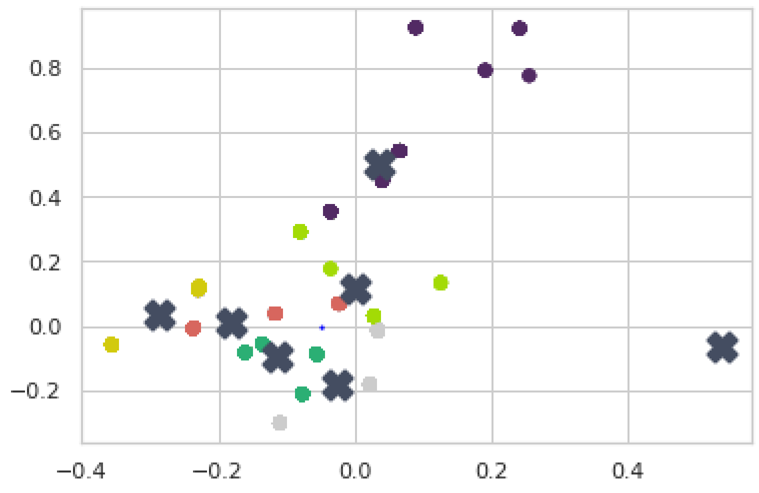
K-means clustering is one of the simplest and popular unsupervised machine learning algorithms. In this method, the **K-means algorithm** identifies **k** number of centroids, and then allocates every data point to the nearest **cluster**, while keeping the centroids as small as possible. In cluster analysis, the **elbow method** is a heuristic used in determining the number of clusters in a data set. The method consists of plotting the explained variation as a function of the number of clusters, and picking the elbow of the curve as the number of clusters to use. The same method can be used to choose the number of parameters in other data-driven models, such as the number of principal components to describe a data set.

The elbow plot for the unsampled data is as shown below:

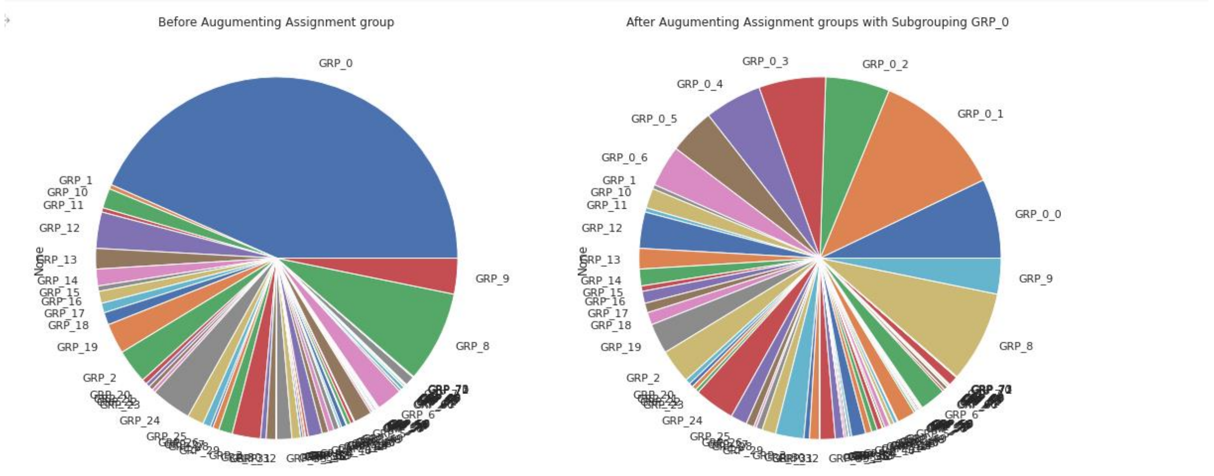


From this plot, we have taken the number of clusters as to be 7.

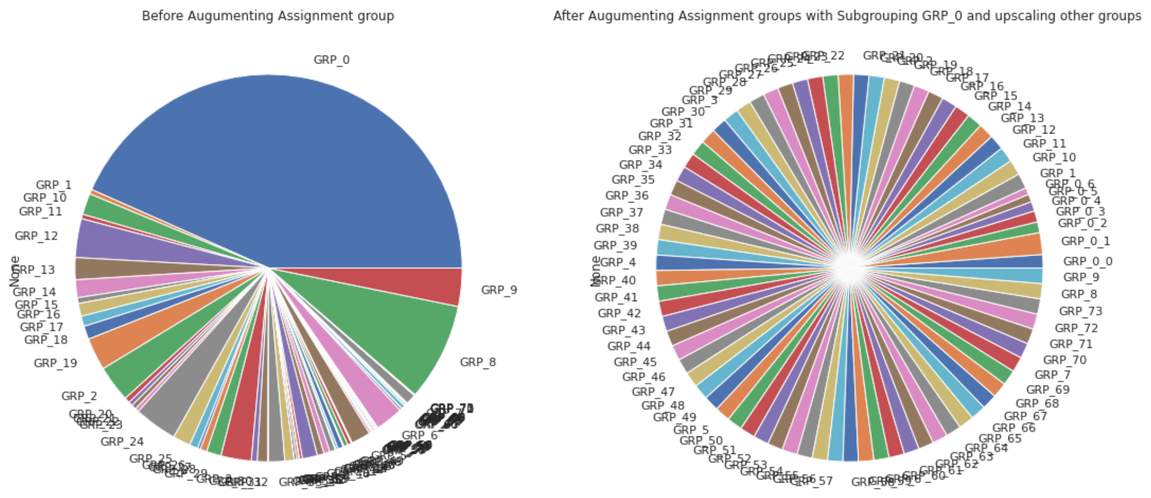
The centroids of the clusters has been visualized as shown below:



On the basis of cluster analysis, Grp_0 has been subdivided into 7 sub-groups and then clubbed with the rest of the dataset. This has been visualized as shown below:



Now, another dataset was prepared in which the rest groups(except Grp_0) has been up-sampled and then clubbed with the subdivided Grp_0 dataset. It has been visualized as shown below:



Defining independent and dependent features

We are concatenating the Short description and Description as “New_Description”.

“New_Description” is considered as ***Independent attribute*** and Target – “Assignment group” is ***Dependent attribute***.

Splitting Datasets

We have used the train_test_split function for splitting a single dataset into training and testing in 70:30 ratios.

The testing subset is for building the model. The testing subset is for using the model on unknown data to evaluate the performance of the model.

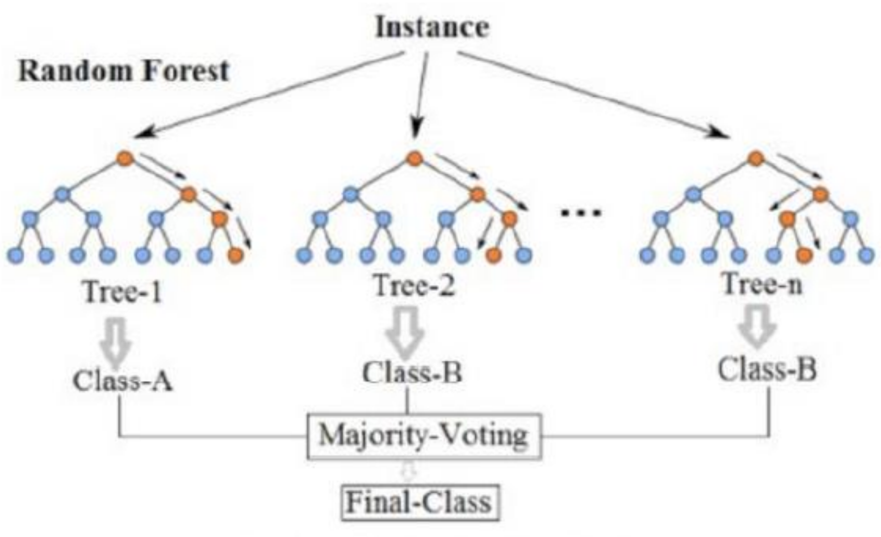
Machine Learning Models

We will be using classification algorithms, to start with we have used below basic Machine Learning algorithm:

RANDOM FOREST CLASSIFIER

Random forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model’s prediction. The fundamental concept behind random forest is a simple but powerful one — the wisdom of crowds. In data science speak, the reason that the random forest model works so well is:

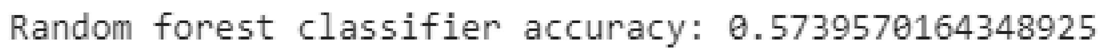
A large number of relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models.



Structure of Random Forest Classification

Below steps have been performed with the initial model:

- Split the data into training and test set.
- Feed the data to the Random Classifier Model.
- Find the accuracy.



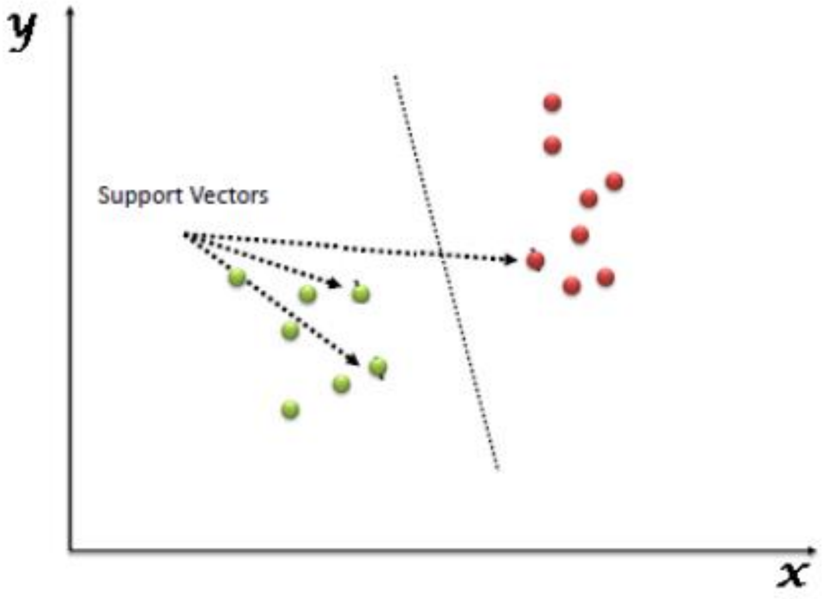
```
Random forest classifier accuracy: 0.5739570164348925
```

Bag of Words

The bag-of-words model is a simplifying representation used in natural language processing and information retrieval. In this model, a text is represented as the bag of its words, disregarding grammar and even word order but keeping multiplicity.

SUPPORT VECTOR MACHINE

“Support Vector Machine” (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well (as shown in the below snapshot).



Below steps have been performed with the initial model:

- Split the data into training and test set using Bag of words.
- Feed the data to the Support vector Classifier Model.
- Find the accuracy.

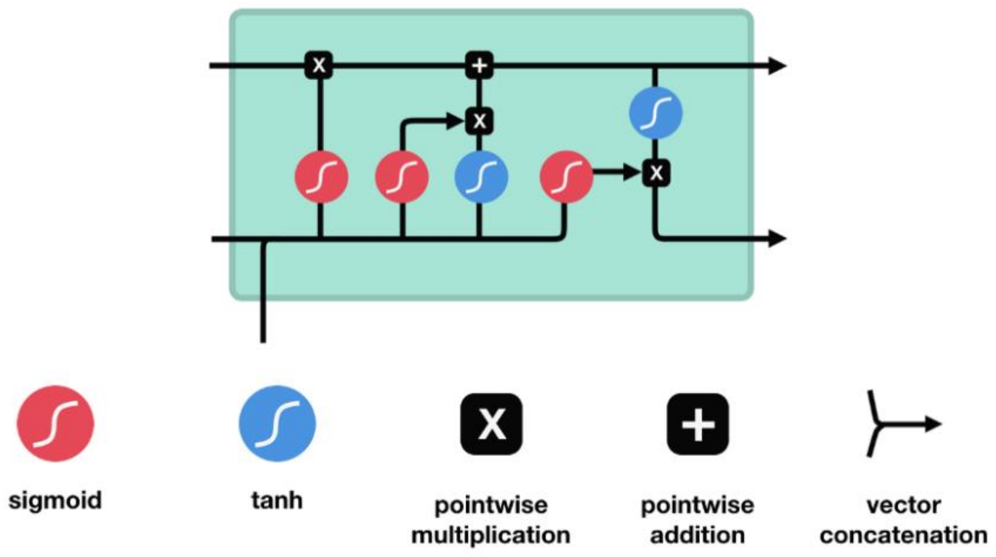
```
SVM-Linear Score for BoW Model is  0.627897176569743
SVM F1 Score for BoW Model is  0.6124185889559783
```

Accuracy result for the sampled data(when only Grp_0 has been resampled)

```
SVM-Linear Score for sampled Model is  0.22187104930467763
SVM F1 Score for sampled Model is  0.23639178929623778
```

LSTM

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network capable of learning order dependence in sequence prediction problems. An LSTM has a similar control flow as a recurrent neural network. It processes data passing on information as it propagates forward. The differences are the operations within the LSTM's cells.



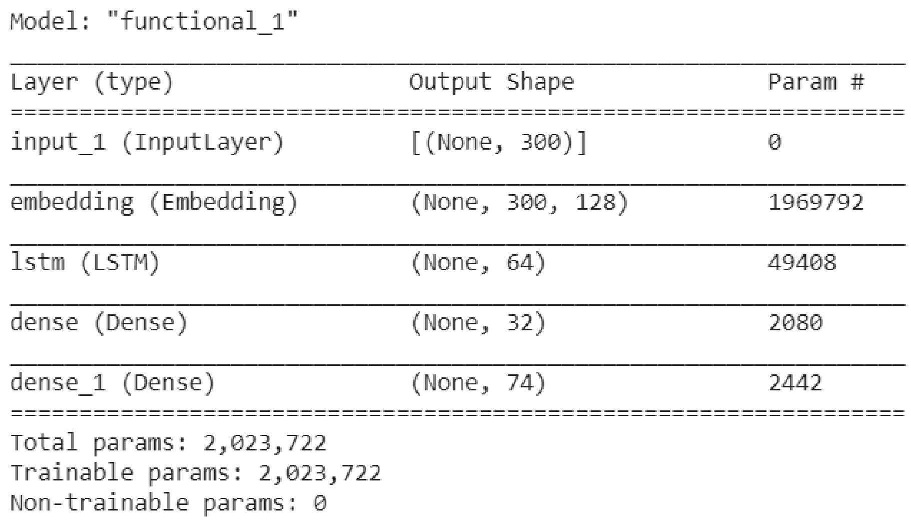
LSTM Cell and It's Operations

These operations are used to allow the LSTM to keep or forget information.

Observation :

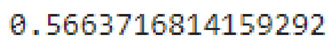
LSTM with unsampled data:

Model:



```
Model: "functional_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_1 (InputLayer)         [(None, 300)]             0         
_________________________________________________________________
embedding (Embedding)        (None, 300, 128)          1969792   
_________________________________________________________________
lstm (LSTM)                  (None, 64)                49408     
_________________________________________________________________
dense (Dense)                (None, 32)                2080      
_________________________________________________________________
dense_1 (Dense)              (None, 74)                2442      
=================================================================
Total params: 2,023,722
Trainable params: 2,023,722
Non-trainable params: 0
```

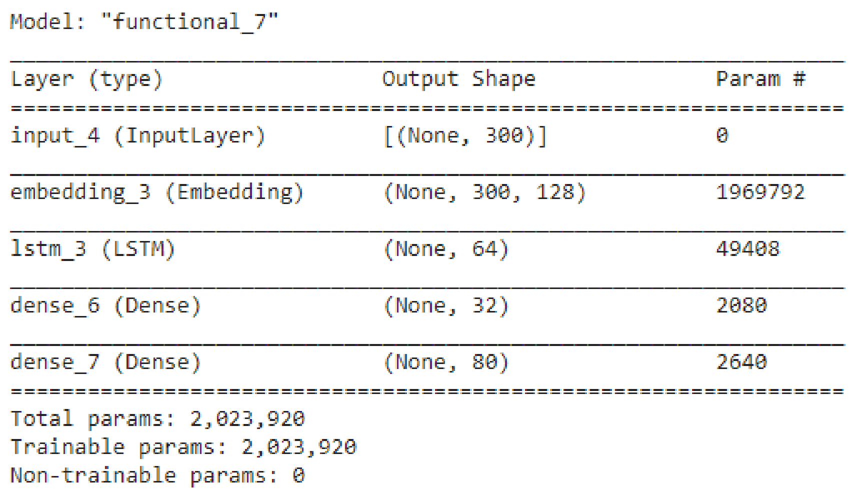
Model Accuracy :



```
0.5663716814159292
```

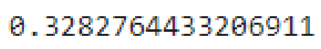
LSTM with sampled data(when only Grp_0 has been resampled):

Model:



```
Model: "functional_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_4 (InputLayer)         [(None, 300)]             0         
_________________________________________________________________
embedding_3 (Embedding)      (None, 300, 128)          1969792   
_________________________________________________________________
lstm_3 (LSTM)                (None, 64)                49408     
_________________________________________________________________
dense_6 (Dense)              (None, 32)                2080      
_________________________________________________________________
dense_7 (Dense)              (None, 80)                2640      
=================================================================
Total params: 2,023,920
Trainable params: 2,023,920
Non-trainable params: 0
```

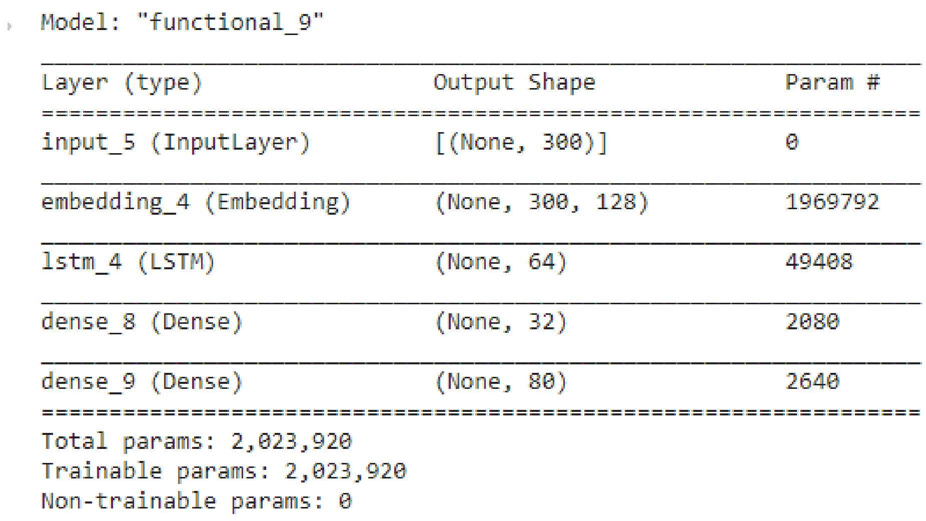
Model Accuracy:



```
0.3282764433206911
```

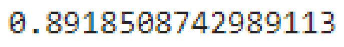
LSTM with sampled data(when only rest groups (except Grp_0) has been resampled and clubbed with the subdivided groups of Grp_0)

Model:



```
Model: "functional_9"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_5 (InputLayer)         [(None, 300)]             0         
_________________________________________________________________
embedding_4 (Embedding)      (None, 300, 128)          1969792   
_________________________________________________________________
lstm_4 (LSTM)                (None, 64)                49408     
_________________________________________________________________
dense_8 (Dense)              (None, 32)                2080      
_________________________________________________________________
dense_9 (Dense)              (None, 80)                2640      
=================================================================
Total params: 2,023,920
Trainable params: 2,023,920
Non-trainable params: 0
```

Model Accuracy:



```
0.8918508742989113
```

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:

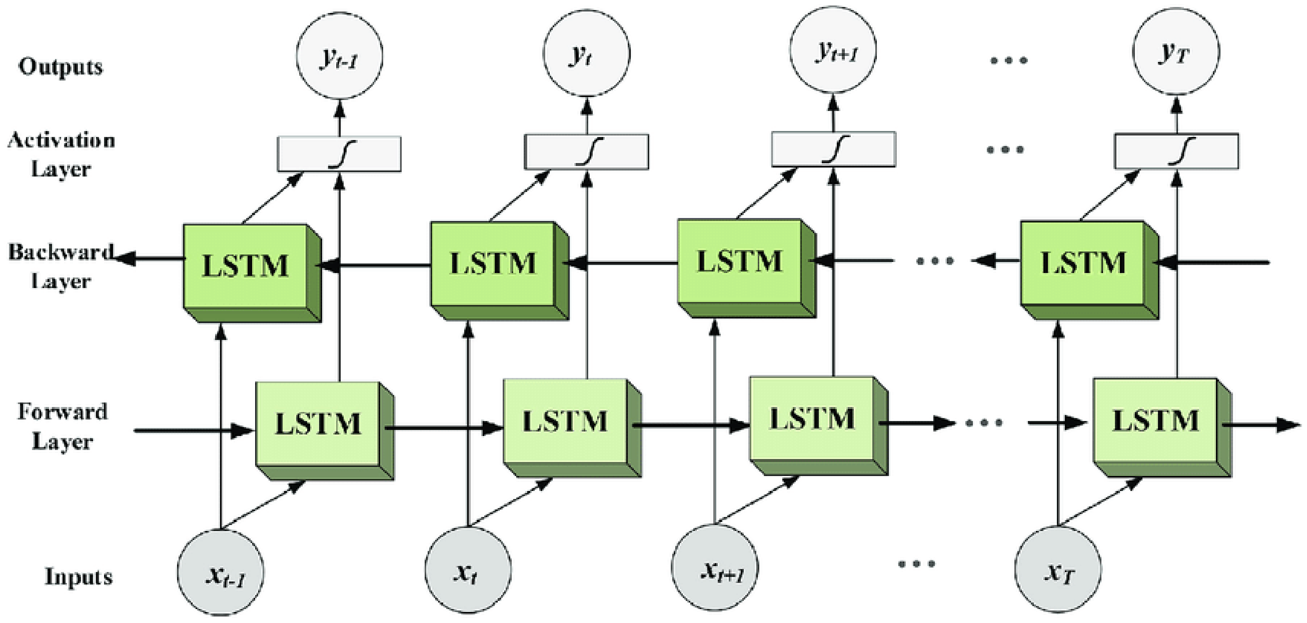
Word2Vec models are shallow, two-layer neural networks that are trained to reconstruct linguistic contexts of words. Word2vec takes as its input a large corpus of text and produces a vector space, typically of several hundred dimensions, with each unique word in the corpus being assigned a corresponding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

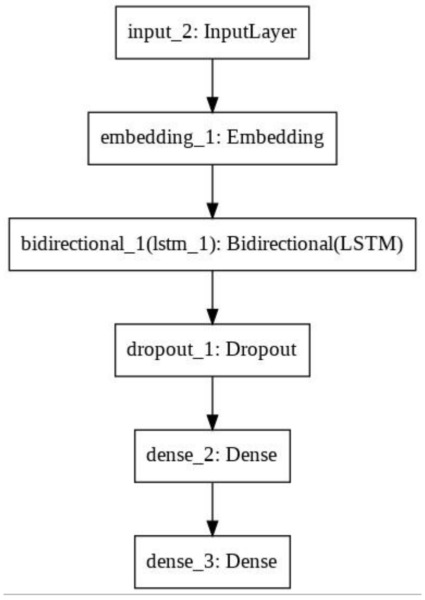
Bidirectional LSTMs are an extension of traditional LSTMs that can improve model performance on classification problems. In problems where all timesteps of the input sequence are available, Bidirectional LSTMs train two instead of one LSTMs on the input sequence. The first on the input sequence as-is 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.



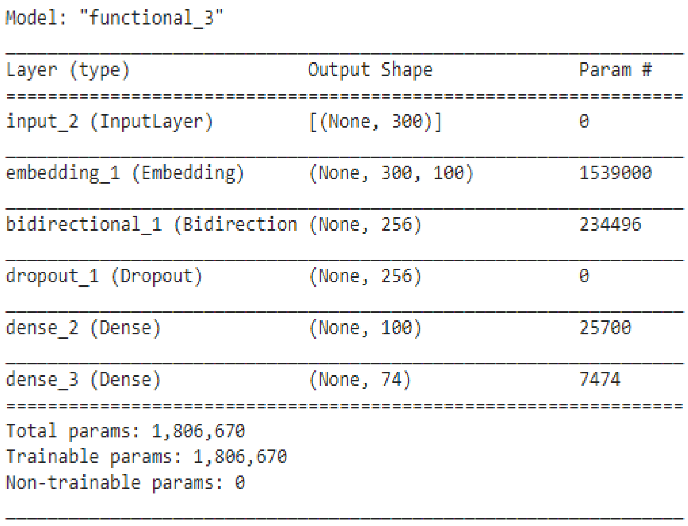
Observations:

Bi-LSTM with unsampled data:

Network flow:

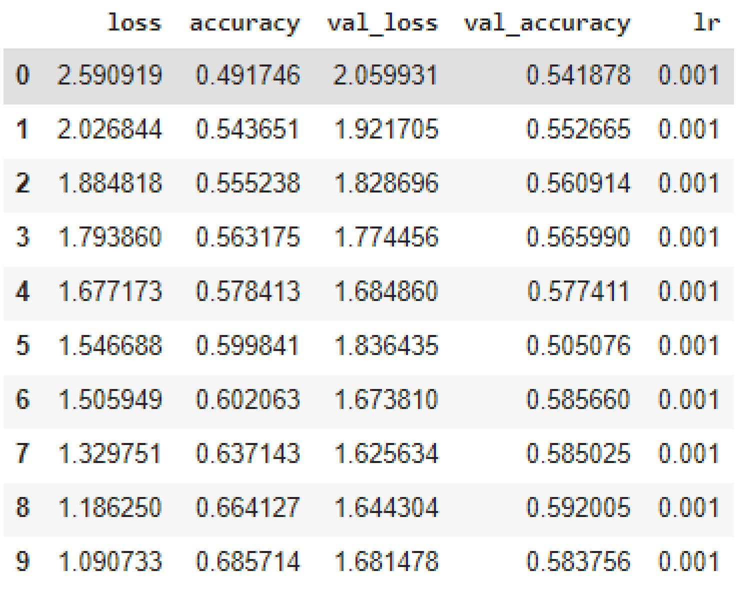


Model:

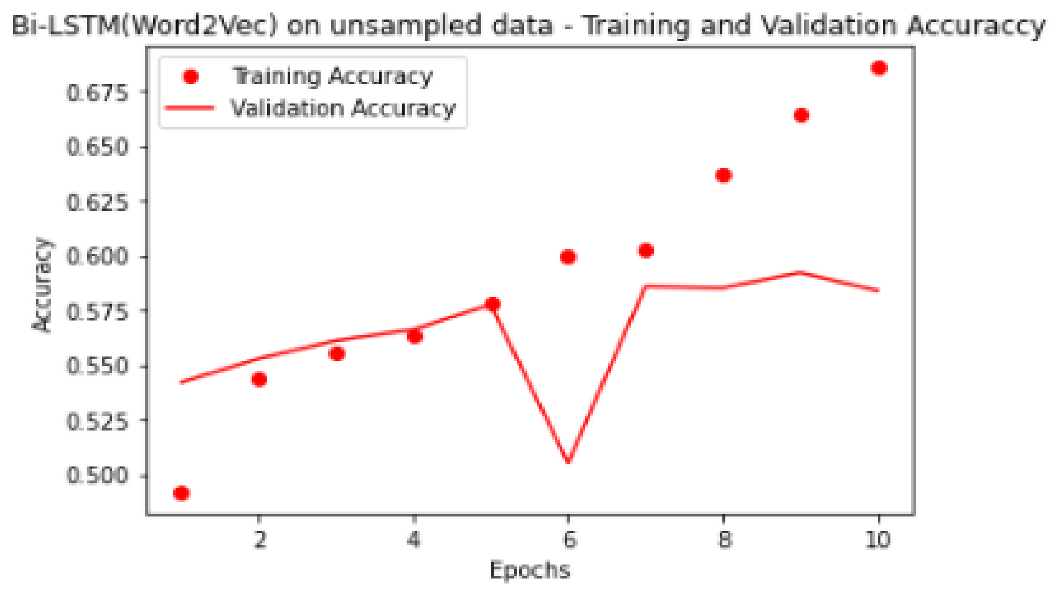


```
Model: "functional_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_2 (InputLayer)         [(None, 300)]             0         
_________________________________________________________________
embedding_1 (Embedding)      (None, 300, 100)          1539000   
_________________________________________________________________
bidirectional_1 (Bidirection (None, 256)               234496    
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0         
_________________________________________________________________
dense_2 (Dense)              (None, 100)               25700     
_________________________________________________________________
dense_3 (Dense)              (None, 74)                7474      
=================================================================
Total params: 1,806,670
Trainable params: 1,806,670
Non-trainable params: 0
_________________________________________________________________
```

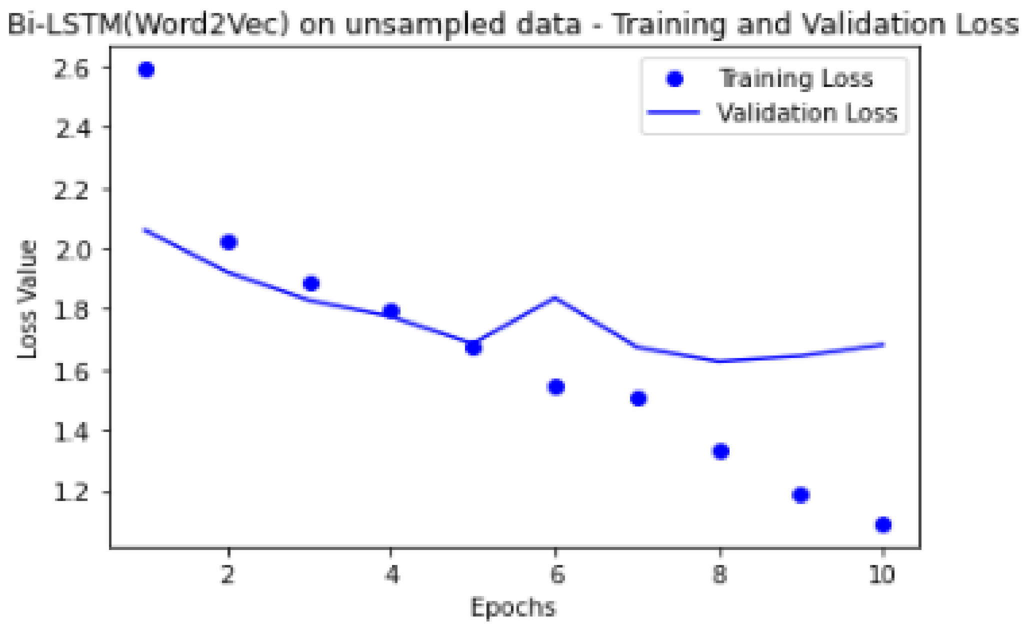
Model History:



	loss	accuracy	val_loss	val_accuracy	lr
0	2.590919	0.491746	2.059931	0.541878	0.001
1	2.026844	0.543651	1.921705	0.552665	0.001
2	1.884818	0.555238	1.828696	0.560914	0.001
3	1.793860	0.563175	1.774456	0.565990	0.001
4	1.677173	0.578413	1.684860	0.577411	0.001
5	1.546688	0.599841	1.836435	0.505076	0.001
6	1.505949	0.602063	1.673810	0.585660	0.001
7	1.329751	0.637143	1.625634	0.585025	0.001
8	1.186250	0.664127	1.644304	0.592005	0.001
9	1.090733	0.685714	1.681478	0.583756	0.001



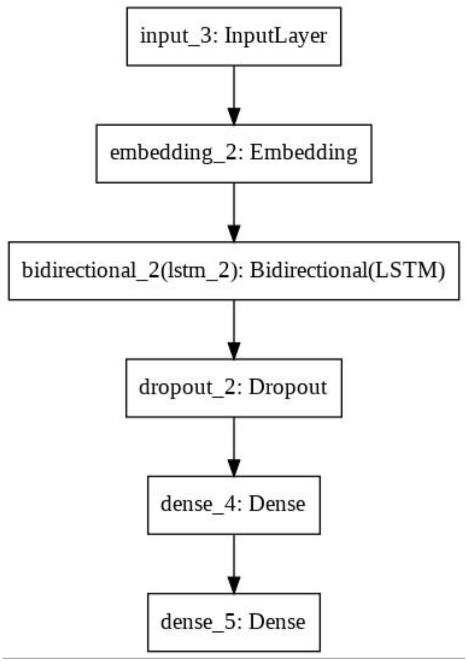
Bi-LSTM(Word2Vec) on unsampled data - Training and Validation Accuraccy
Training Accuracy
Validation Accuracy
Accuracy
0.500
0.525
0.550
0.575
0.600
0.625
0.650
0.675
2
4
6
8
10
Epochs



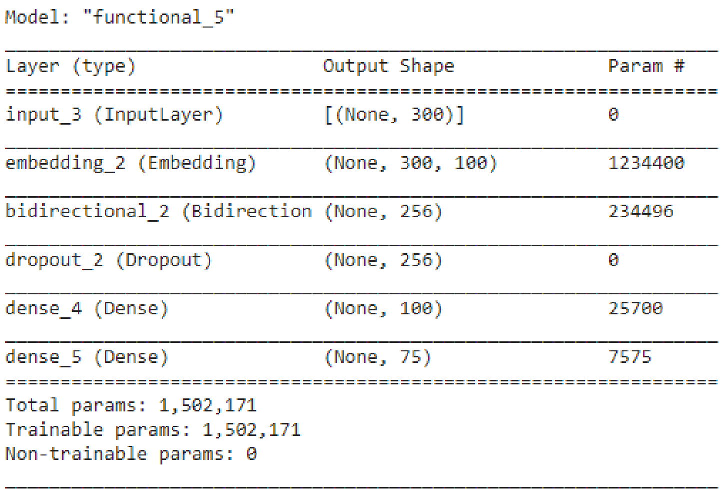
Bi-LSTM(Word2Vec) on unsampled data - Training and Validation Loss
Training Loss
Validation Loss
Loss Value
1.2
1.4
1.6
1.8
2.0
2.2
2.4
2.6
2
4
6
8
10
Epochs

Bi-LSTM(Word2Vec) on sampled data:

Network flow:

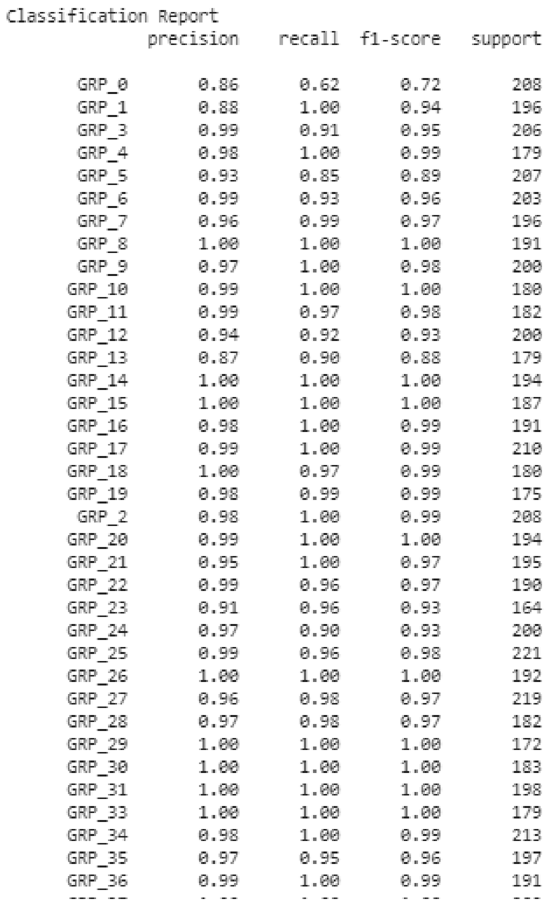


Model:



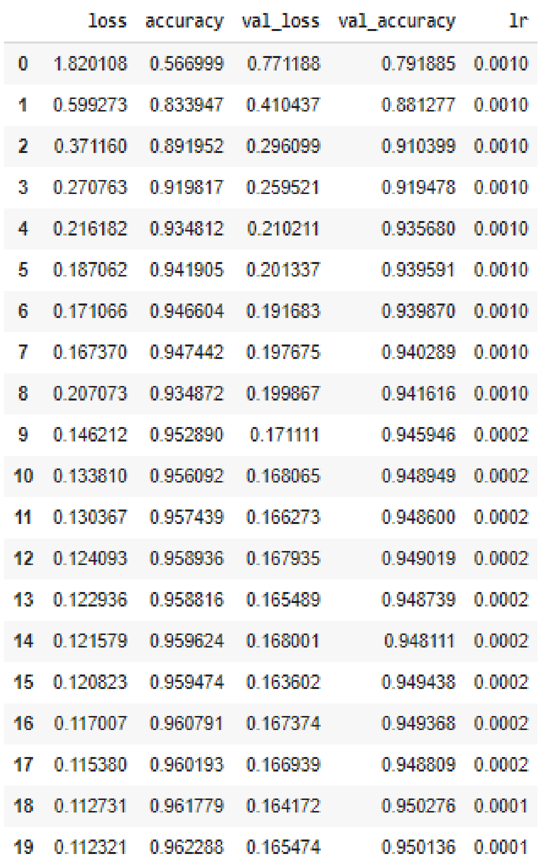
```
Model: "functional_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_3 (InputLayer)         [(None, 300)]             0         
_________________________________________________________________
embedding_2 (Embedding)      (None, 300, 100)          1234400   
_________________________________________________________________
bidirectional_2 (Bidirection (None, 256)               234496    
_________________________________________________________________
dropout_2 (Dropout)          (None, 256)               0         
_________________________________________________________________
dense_4 (Dense)              (None, 100)               25700     
_________________________________________________________________
dense_5 (Dense)              (None, 75)                7575      
=================================================================
Total params: 1,502,171
Trainable params: 1,502,171
Non-trainable params: 0
_________________________________________________________________
```

Sample Class Level accuracies:

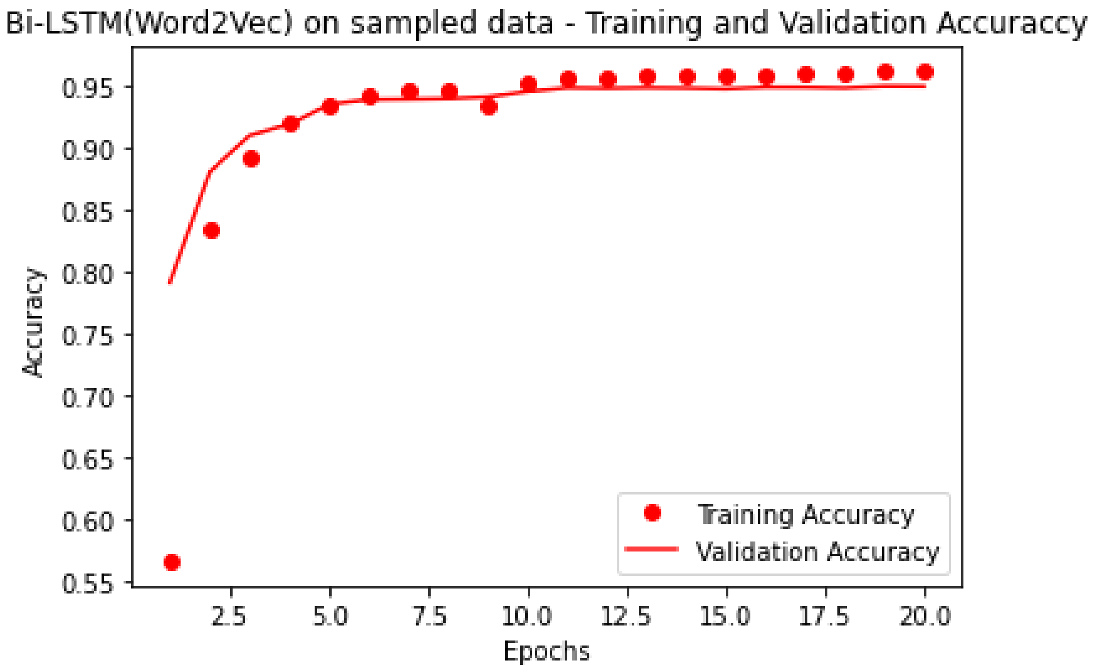


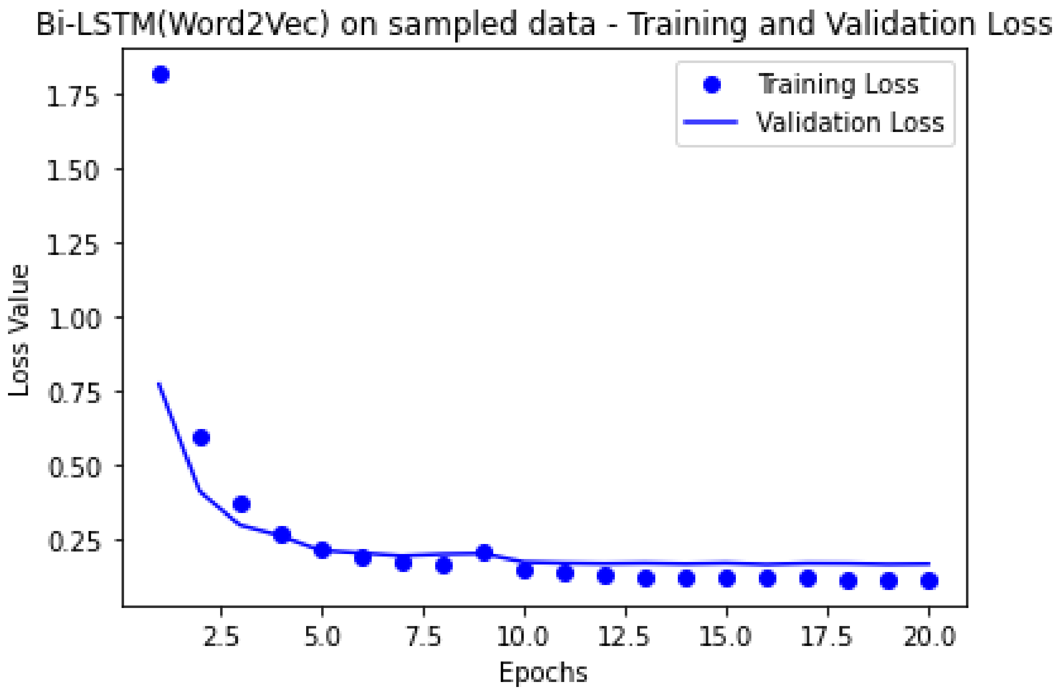
```
Classification Report
              precision    recall  f1-score   support

       GRP_0       0.86      0.62      0.72       208
       GRP_1       0.88      1.00      0.94       196
       GRP_3       0.99      0.91      0.95       206
       GRP_4       0.98      1.00      0.99       179
       GRP_5       0.93      0.85      0.89       207
       GRP_6       0.99      0.93      0.96       203
       GRP_7       0.96      0.99      0.97       196
       GRP_8       1.00      1.00      1.00       191
       GRP_9       0.97      1.00      0.98       200
      GRP_10       0.99      1.00      1.00       180
      GRP_11       0.99      0.97      0.98       182
      GRP_12       0.94      0.92      0.93       200
      GRP_13       0.87      0.90      0.88       179
      GRP_14       1.00      1.00      1.00       194
      GRP_15       1.00      1.00      1.00       187
      GRP_16       0.98      1.00      0.99       191
      GRP_17       0.99      1.00      0.99       210
      GRP_18       1.00      0.97      0.99       180
      GRP_19       0.98      0.99      0.99       175
       GRP_2       0.98      1.00      0.99       208
      GRP_20       0.99      1.00      1.00       194
      GRP_21       0.95      1.00      0.97       195
      GRP_22       0.99      0.96      0.97       190
      GRP_23       0.91      0.96      0.93       164
      GRP_24       0.97      0.90      0.93       200
      GRP_25       0.99      0.96      0.98       221
      GRP_26       1.00      1.00      1.00       192
      GRP_27       0.96      0.98      0.97       219
      GRP_28       0.97      0.98      0.97       182
      GRP_29       1.00      1.00      1.00       172
      GRP_30       1.00      1.00      1.00       183
      GRP_31       1.00      1.00      1.00       198
      GRP_33       1.00      1.00      1.00       179
      GRP_34       0.98      1.00      0.99       213
      GRP_35       0.97      0.95      0.96       197
      GRP_36       0.99      1.00      0.99       191
```

Model Summary:



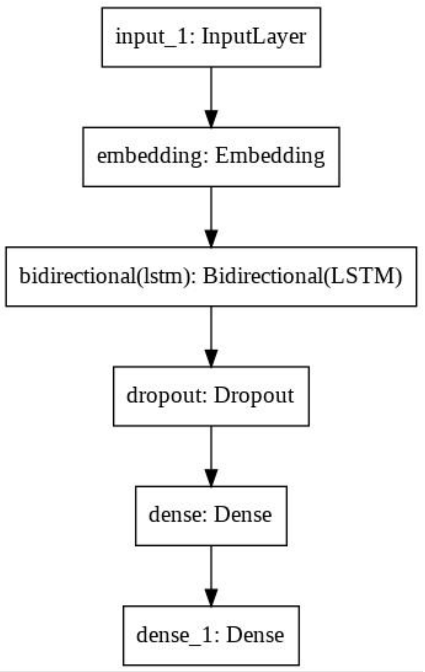
	loss	accuracy	val_loss	val_accuracy	lr
0	1.820108	0.566999	0.771188	0.791885	0.0010
1	0.599273	0.833947	0.410437	0.881277	0.0010
2	0.371160	0.891952	0.296099	0.910399	0.0010
3	0.270763	0.919817	0.259521	0.919478	0.0010
4	0.216182	0.934812	0.210211	0.935680	0.0010
5	0.187062	0.941905	0.201337	0.939591	0.0010
6	0.171066	0.946604	0.191683	0.939870	0.0010
7	0.167370	0.947442	0.197675	0.940289	0.0010
8	0.207073	0.934872	0.199867	0.941616	0.0010
9	0.146212	0.952890	0.171111	0.945946	0.0002
10	0.133810	0.956092	0.168065	0.948949	0.0002
11	0.130367	0.957439	0.166273	0.948600	0.0002
12	0.124093	0.958936	0.167935	0.949019	0.0002
13	0.122936	0.958816	0.165489	0.948739	0.0002
14	0.121579	0.959624	0.168001	0.948111	0.0002
15	0.120823	0.959474	0.163602	0.949438	0.0002
16	0.117007	0.960791	0.167374	0.949368	0.0002
17	0.115380	0.960193	0.166939	0.948809	0.0002
18	0.112731	0.961779	0.164172	0.950276	0.0001
19	0.112321	0.962288	0.165474	0.950136	0.0001


Bi-LSTM(Word2Vec) on sampled data - Training and Validation Accuraccy
Accuracy
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0.55
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Epochs
Training Accuracy
Validation Accuracy

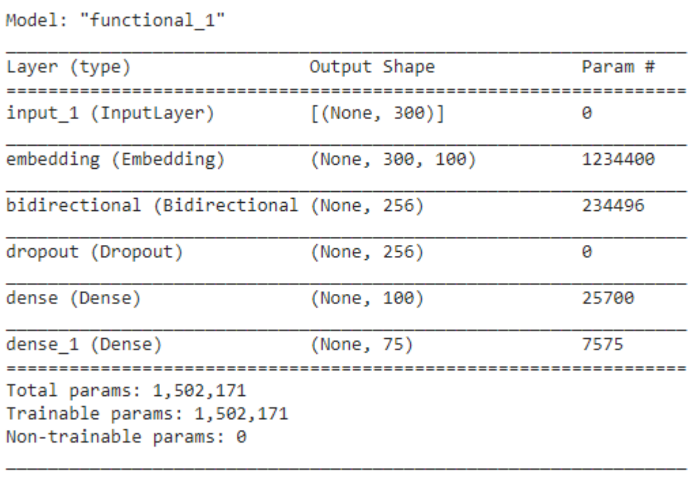

Bi-LSTM(Word2Vec) on sampled data - Training and Validation Loss
Loss Value
1.75
1.50
1.25
1.00
0.75
0.50
0.25
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Epochs
Training Loss
Validation Loss

<u>Bi-LSTM(Glove) on sampled data:</u>

Network flow:

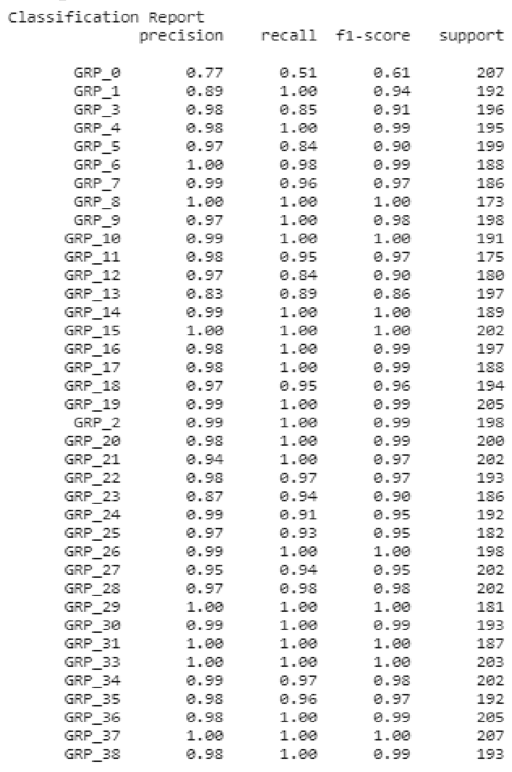


Model:



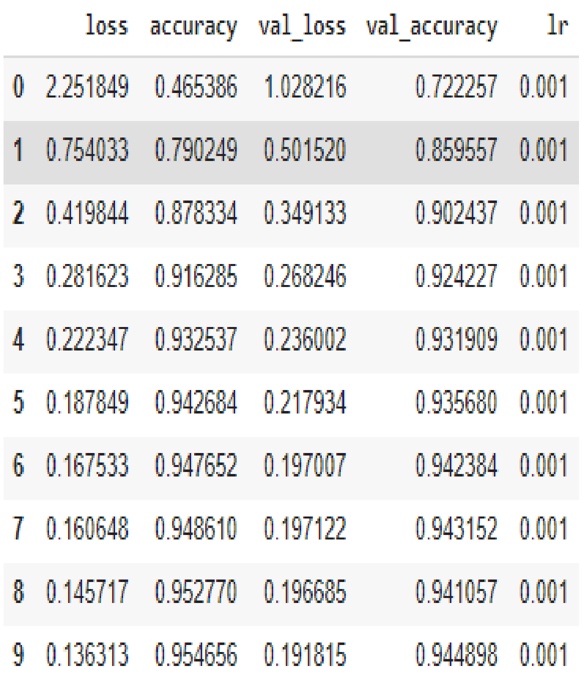
```
Model: "functional_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_1 (InputLayer)         [(None, 300)]             0         
_________________________________________________________________
embedding (Embedding)        (None, 300, 100)          1234400   
_________________________________________________________________
bidirectional (Bidirectional (None, 256)               234496    
_________________________________________________________________
dropout (Dropout)            (None, 256)               0         
_________________________________________________________________
dense (Dense)                (None, 100)               25700     
_________________________________________________________________
dense_1 (Dense)              (None, 75)                7575      
=================================================================
Total params: 1,502,171
Trainable params: 1,502,171
Non-trainable params: 0
_________________________________________________________________
```

Sample Class Level accuracies:

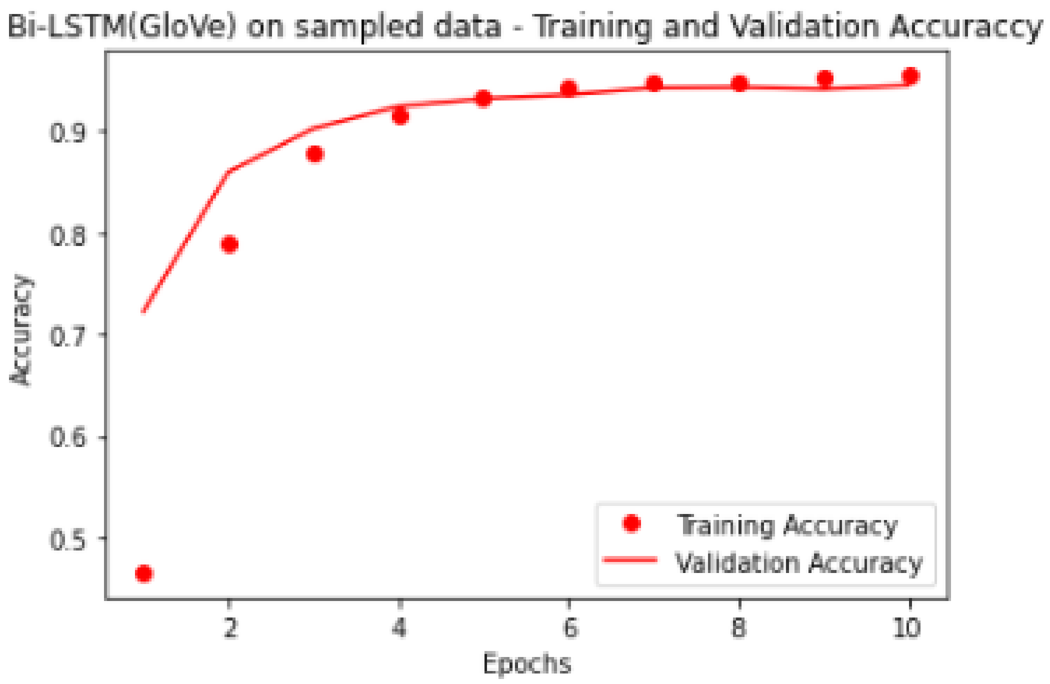


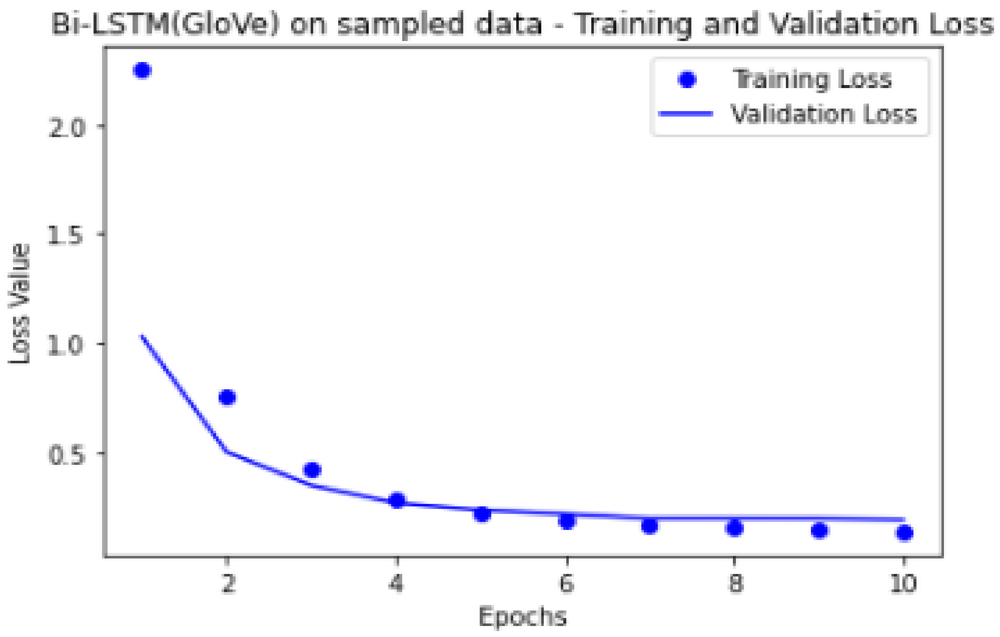
```
Classification Report
              precision    recall  f1-score   support

       GRP_0       0.77      0.51      0.61       207
       GRP_1       0.89      1.00      0.94       192
       GRP_3       0.98      0.85      0.91       196
       GRP_4       0.98      1.00      0.99       195
       GRP_5       0.97      0.84      0.90       199
       GRP_6       1.00      0.98      0.99       188
       GRP_7       0.99      0.96      0.97       186
       GRP_8       1.00      1.00      1.00       173
       GRP_9       0.97      1.00      0.98       198
      GRP_10       0.99      1.00      1.00       191
      GRP_11       0.98      0.95      0.97       175
      GRP_12       0.97      0.84      0.90       180
      GRP_13       0.83      0.89      0.86       197
      GRP_14       0.99      1.00      1.00       189
      GRP_15       1.00      1.00      1.00       202
      GRP_16       0.98      1.00      0.99       197
      GRP_17       0.98      1.00      0.99       188
      GRP_18       0.97      0.95      0.96       194
      GRP_19       0.99      1.00      0.99       205
       GRP_2       0.99      1.00      0.99       198
      GRP_20       0.98      1.00      0.99       200
      GRP_21       0.94      1.00      0.97       202
      GRP_22       0.98      0.97      0.97       193
      GRP_23       0.87      0.94      0.90       186
      GRP_24       0.99      0.91      0.95       192
      GRP_25       0.97      0.93      0.95       182
      GRP_26       0.99      1.00      1.00       198
      GRP_27       0.95      0.94      0.95       202
      GRP_28       0.97      0.98      0.98       202
      GRP_29       1.00      1.00      1.00       181
      GRP_30       0.99      1.00      0.99       193
      GRP_31       1.00      1.00      1.00       187
      GRP_33       1.00      1.00      1.00       203
      GRP_34       0.99      0.97      0.98       202
      GRP_35       0.98      0.96      0.97       192
      GRP_36       0.98      1.00      0.99       205
      GRP_37       1.00      1.00      1.00       207
      GRP_38       0.98      1.00      0.99       193
```

Model Summary:



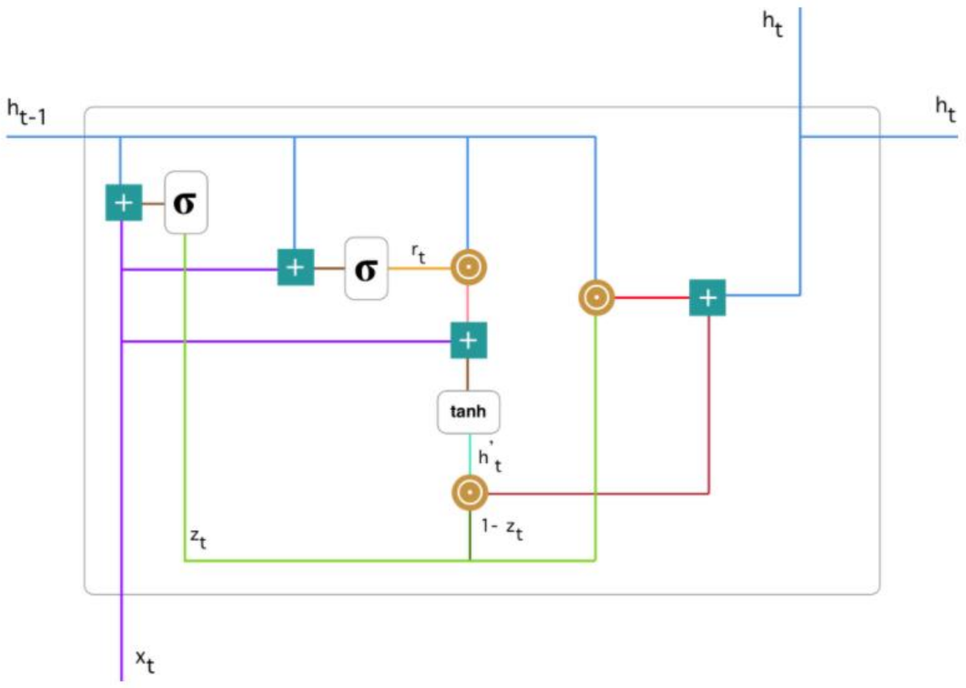
	loss	accuracy	val_loss	val_accuracy	lr
0	2.251849	0.465386	1.028216	0.722257	0.001
1	0.754033	0.790249	0.501520	0.859557	0.001
2	0.419844	0.878334	0.349133	0.902437	0.001
3	0.281623	0.916285	0.268246	0.924227	0.001
4	0.222347	0.932537	0.236002	0.931909	0.001
5	0.187849	0.942684	0.217934	0.935680	0.001
6	0.167533	0.947652	0.197007	0.942384	0.001
7	0.160648	0.948610	0.197122	0.943152	0.001
8	0.145717	0.952770	0.196685	0.941057	0.001
9	0.136313	0.954656	0.191815	0.944898	0.001


Bi-LSTM(GloVe) on sampled data - Training and Validation Accuraccy
Accuracy
0.9
0.8
0.7
0.6
0.5
Epochs
2
4
6
8
10
● Training Accuracy
Validation Accuracy


Bi-LSTM(GloVe) on sampled data - Training and Validation Loss
Loss Value
2.0
1.5
1.0
0.5
Epochs
2
4
6
8
10
● Training Loss
Validation Loss

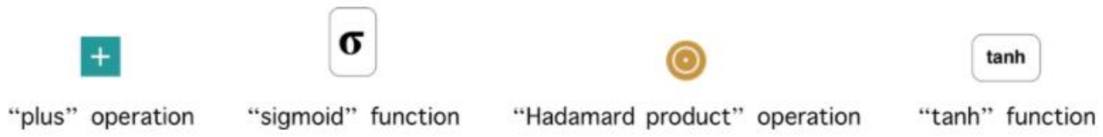
GRU

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 uses, so-called, **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 remove information which is irrelevant to the prediction. Detailed version of a single GRU is as shown below:



Gated Recurrent Unit

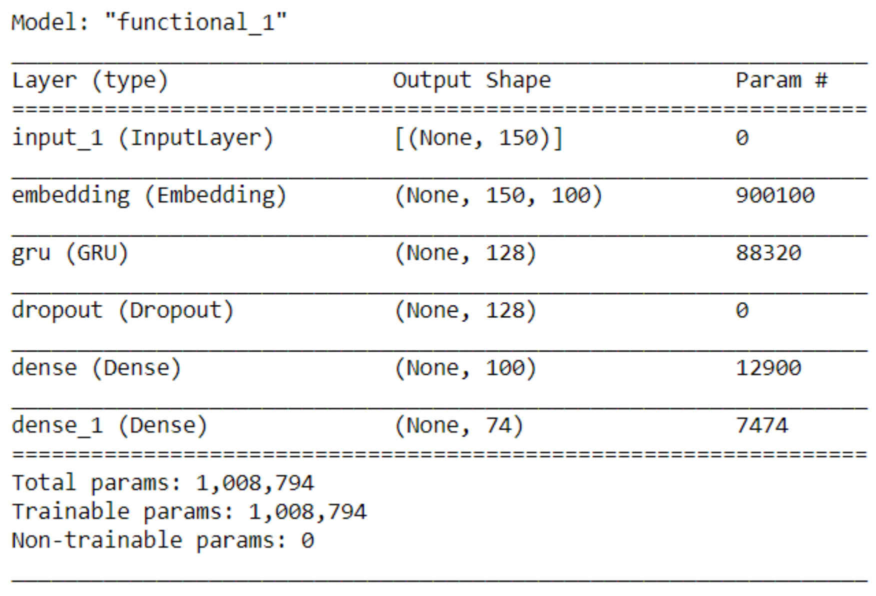
where the notations are as follows:



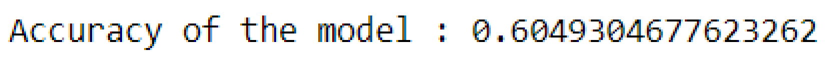
Update Gate: The update gate acts similar to the forget and input gate of an LSTM. It decides what information to throw away and what new information to add.
Reset Gate: The reset gate is another gate used to decide how much past information to forget.
GRU’s has fewer tensor operations; therefore, they are a little speedier to train than LSTM’s.

GRU with unsampled data:

Model:

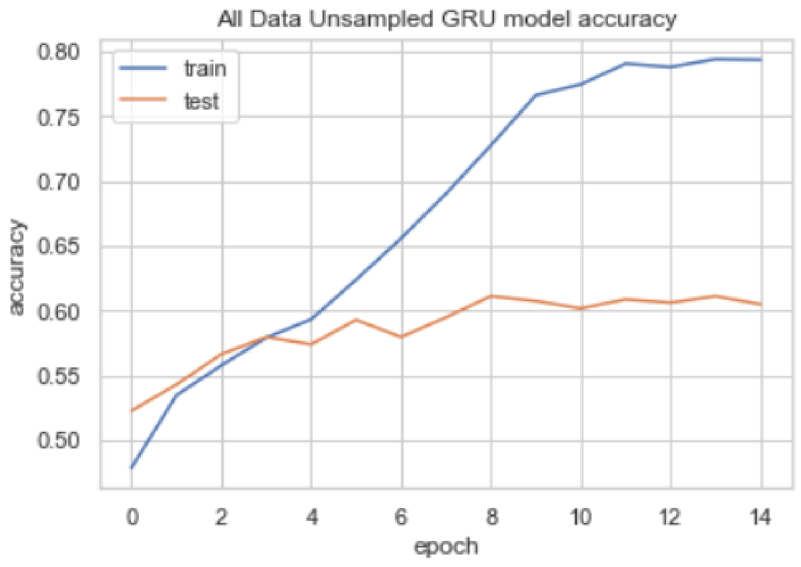


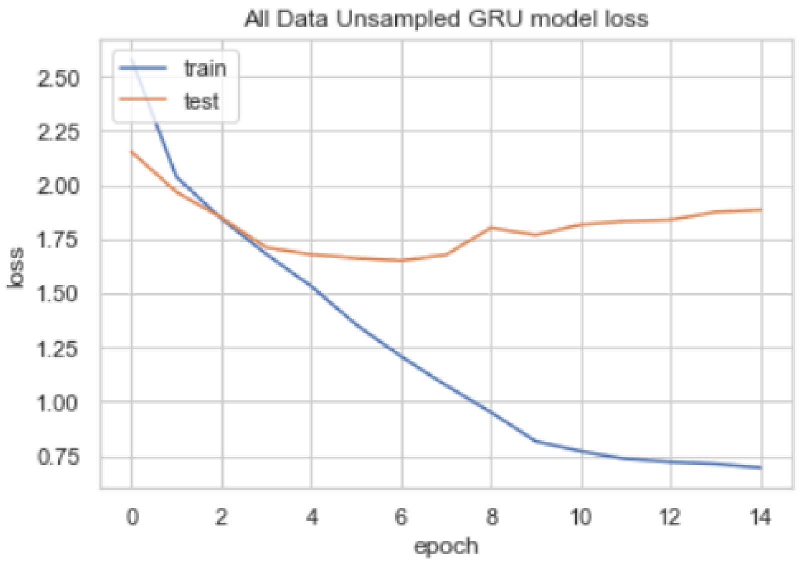
```
Model: "functional_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_1 (InputLayer)         [(None, 150)]             0         
_________________________________________________________________
embedding (Embedding)        (None, 150, 100)          900100    
_________________________________________________________________
gru (GRU)                    (None, 128)               88320     
_________________________________________________________________
dropout (Dropout)            (None, 128)               0         
_________________________________________________________________
dense (Dense)                (None, 100)               12900     
_________________________________________________________________
dense_1 (Dense)              (None, 74)                7474      
=================================================================
Total params: 1,008,794
Trainable params: 1,008,794
Non-trainable params: 0
_________________________________________________________________
```



```
Accuracy of the model : 0.6049304677623262
```

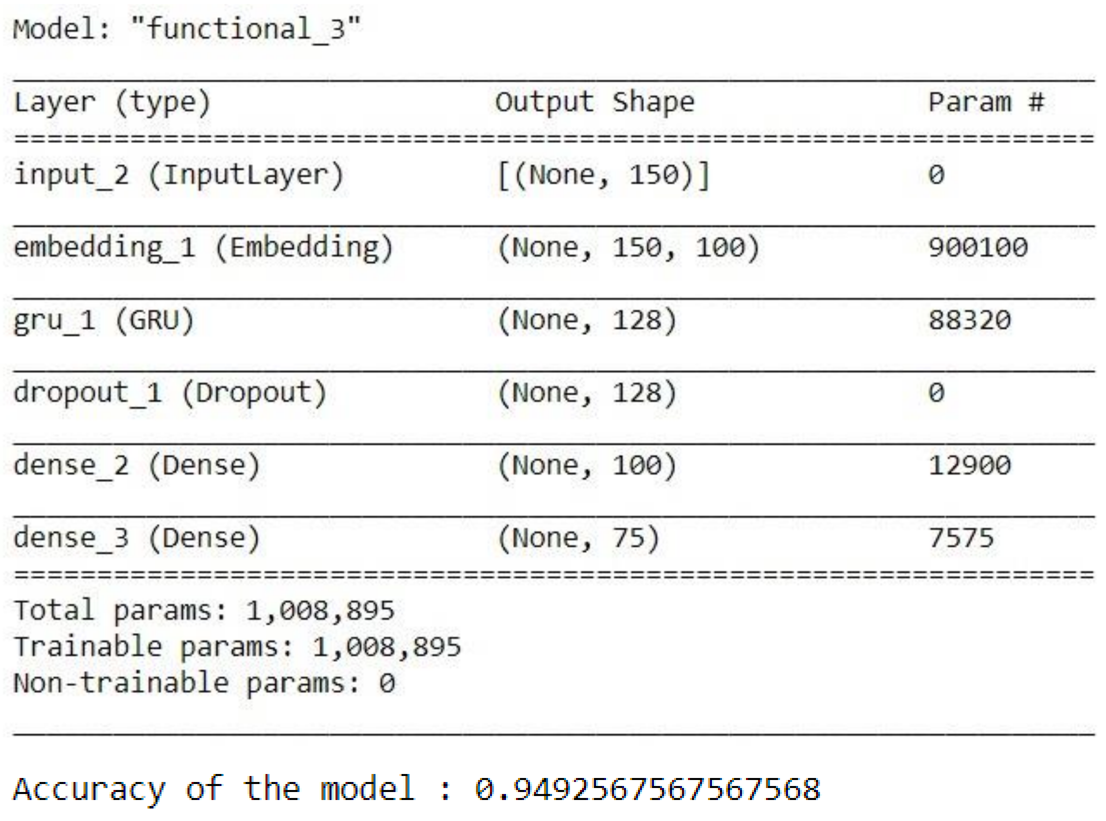
Graphical representation of model accuracy and loss is as shown below:





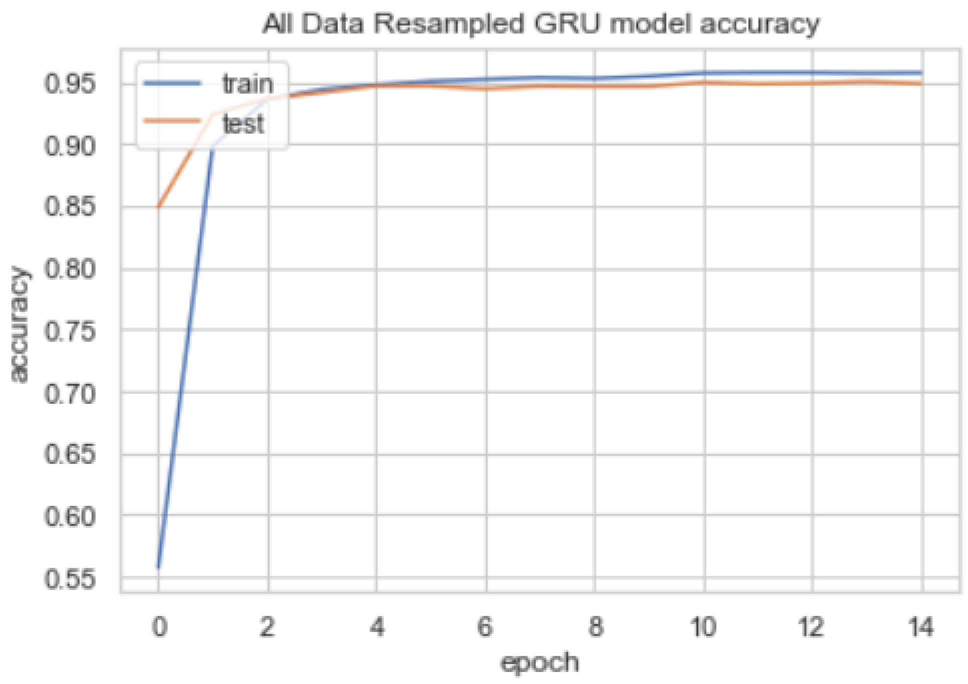
GRU with sampled data:

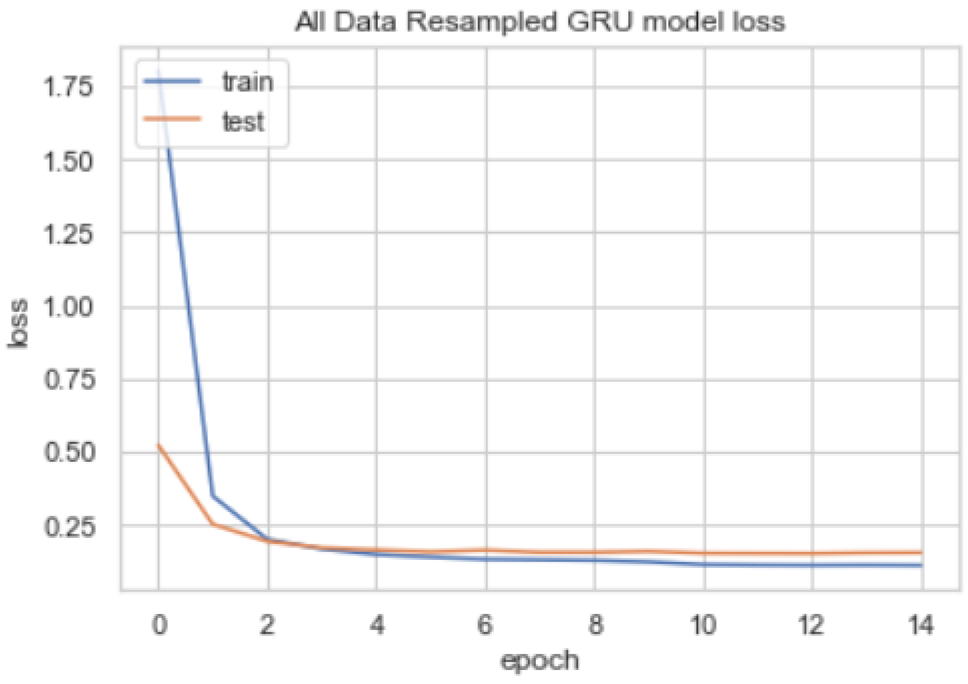
Model:



```
Model: "functional_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 150)]             0
_________________________________________________________________
embedding_1 (Embedding)      (None, 150, 100)          900100
_________________________________________________________________
gru_1 (GRU)                  (None, 128)               88320
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 100)               12900
_________________________________________________________________
dense_3 (Dense)              (None, 75)                7575
=================================================================
Total params: 1,008,895
Trainable params: 1,008,895
Non-trainable params: 0
_________________________________________________________________

Accuracy of the model : 0.9492567567567568
```

Graphical representation of model accuracy and loss is as shown below:





BERT (language model)

What is BERT?

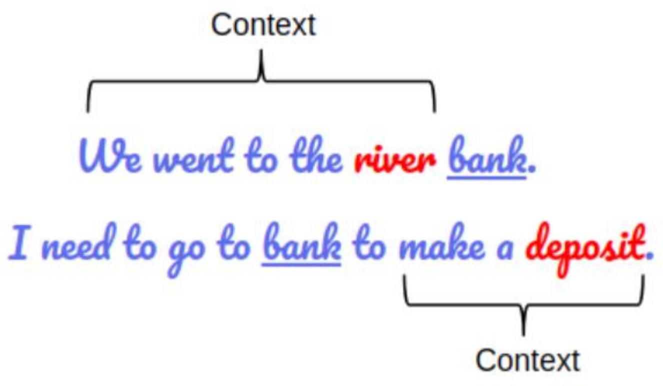
BERT is an open source machine learning framework for natural language processing (NLP). BERT is designed to help computers understand the meaning of ambiguous language in text by using surrounding text to establish context. The BERT framework was pre-trained using text from Wikipedia and can be fine-tuned with question and answer datasets.

BERT, which stands for Bidirectional Encoder Representations from Transformers, is based on Transformers, a deep learning model in which every output element is connected to every input element, and the weightings between them are dynamically calculated based upon their connection.

BERT is pre-trained on a large corpus of unlabelled text including the entire Wikipedia(that's 2,500 million words!) and Book Corpus (800 million words).

BERT is a **"deeply bidirectional"** model. Bidirectional means that BERT learns information from both the left and the right side of a token's context during the training phase.

The bidirectionality of a model is important for truly understanding the meaning of a language. Let's see an example to illustrate this. There are two sentences in this example and both of them involve the word "bank":


Context
We went to the river bank.
I need to go to bank to make a deposit.
Context

If we try to predict the nature of the word "bank" by only taking either the left or the right context, then we will be making an error in at least one of the two given examples.

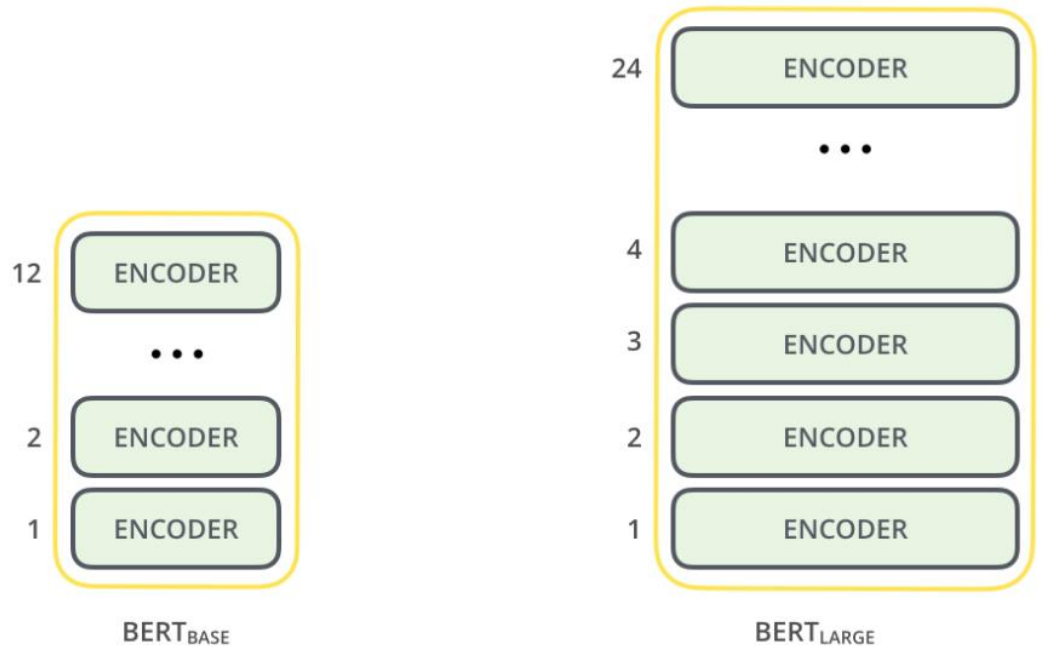
One way to deal with this is to consider both the left and the right context before making a prediction. That's exactly what BERT does!

How Does BERT Work?

1. BERT's Architecture

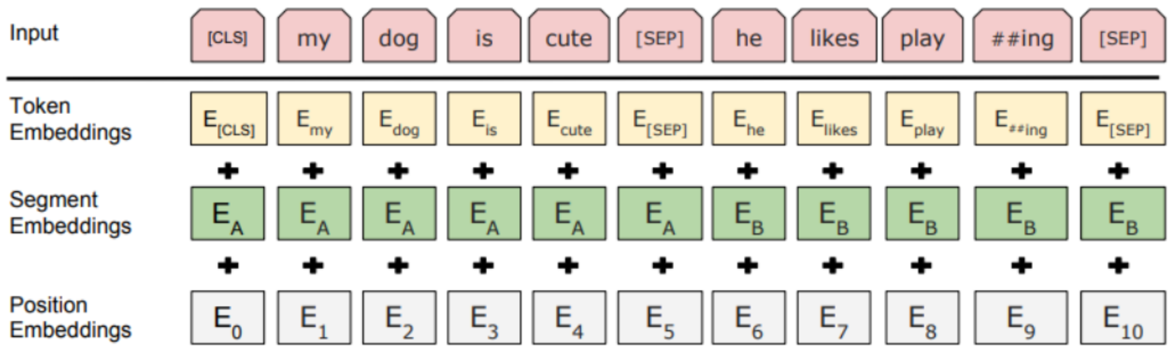
The BERT architecture builds on top of Transformer. We currently have two variants available:

- BERT Base: 12 layers (transformer blocks), 12 attention heads, and 110 million parameters
- BERT Large: 24 layers (transformer blocks), 16 attention heads and, 340 million parameters



Both BERT model sizes have a large number of encoder layers (which the paper calls Transformer Blocks) – twelve for the Base version, and twenty-four for the Large version. These also have larger feedforward-networks (768 and 1024 hidden units respectively), and more attention heads (12 and 16 respectively) than the default configuration in the reference implementation of the Transformer in the initial paper (6 encoder layers, 512 hidden units, and 8 attention heads).

2. Text Pre-processing



Input
[CLS] my dog is cute [SEP] he likes play ##ing [SEP]
Token Embeddings
E[CLS] Emy Edog Eis Ecute E[SEP] Ehe Elikes Eplay E##ing E[SEP]
Segment Embeddings
EA EA EA EA EA EA EB EB EB EB EB
Position Embeddings
E0 E1 E2 E3 E4 E5 E6 E7 E8 E9 E10

The developers behind BERT have added a specific set of rules to represent the input text for the model. Many of these are creative design choices that make the model even better.

For starters, every input embedding is a combination of 3 embeddings:

1. **Position Embeddings**: BERT learns and uses positional embeddings to express the position of words in a sentence. These are added to overcome the limitation of Transformer which, unlike an RNN, is not able to capture "sequence" or "order" information
2. **Segment Embeddings**: BERT can also take sentence pairs as inputs for tasks (Question-Answering). That's why it learns a unique embedding for the first and the second sentences to help the model distinguish between them. In the above example, all the tokens marked as EA belong to sentence A (and similarly for EB)
3. **Token Embeddings**: These are the embeddings learned for the specific token from the Word Piece token vocabulary.

Such a comprehensive embedding scheme contains a lot of useful information for the model.

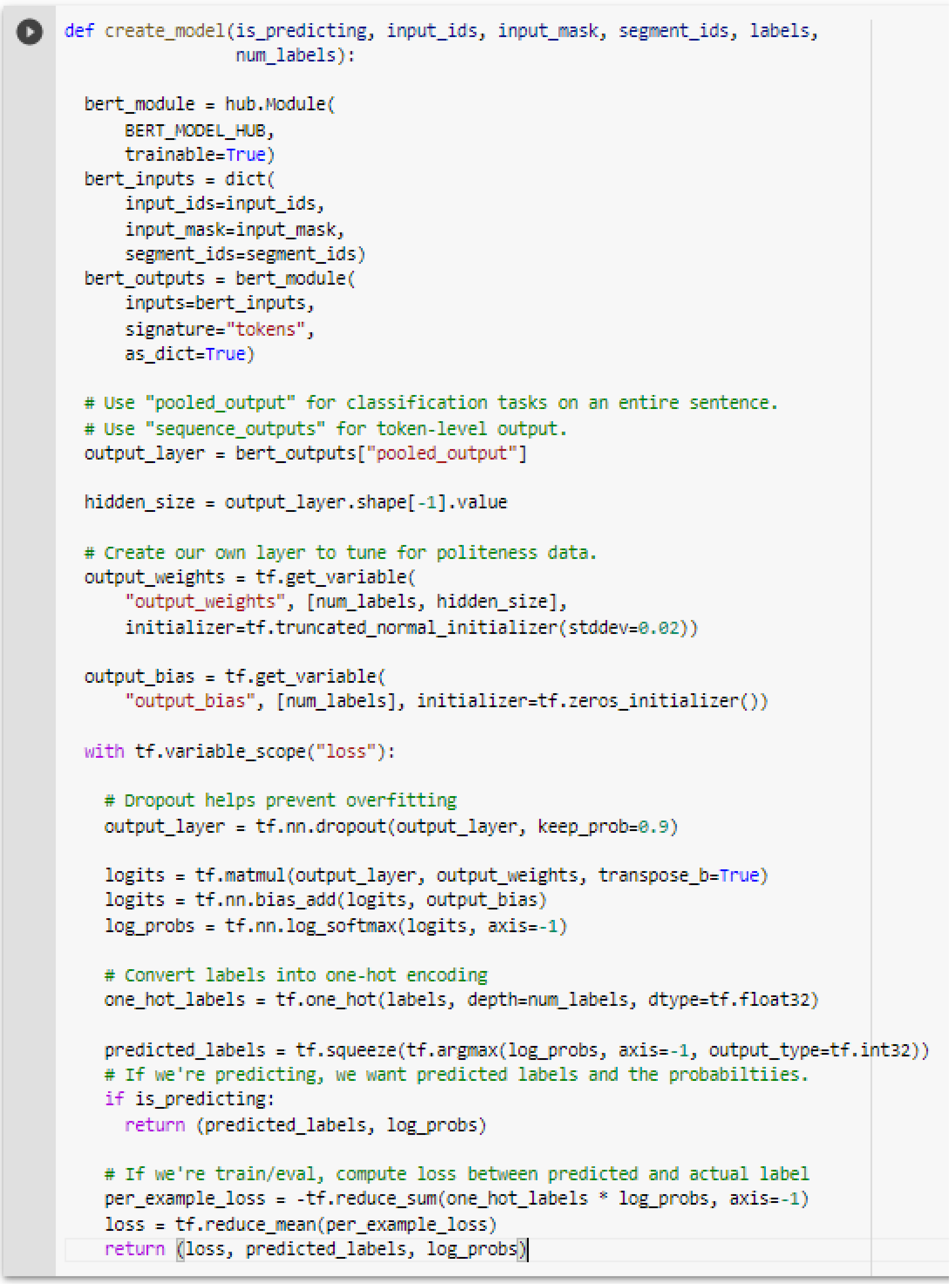
These combinations of pre-processing steps make BERT so versatile. This implies that without making any major change in the model's architecture, we can easily train it on multiple kinds of NLP tasks.

3. Pre-training Tasks

BERT is pre-trained on two NLP tasks:

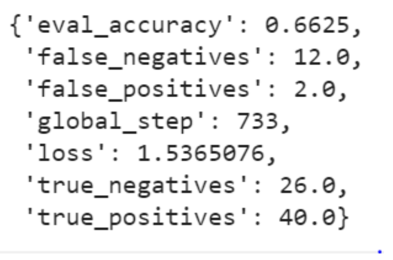
- Masked Language Modelling
- Next Sentence Prediction

We have implemented BERT for multi class Text Classification using Python

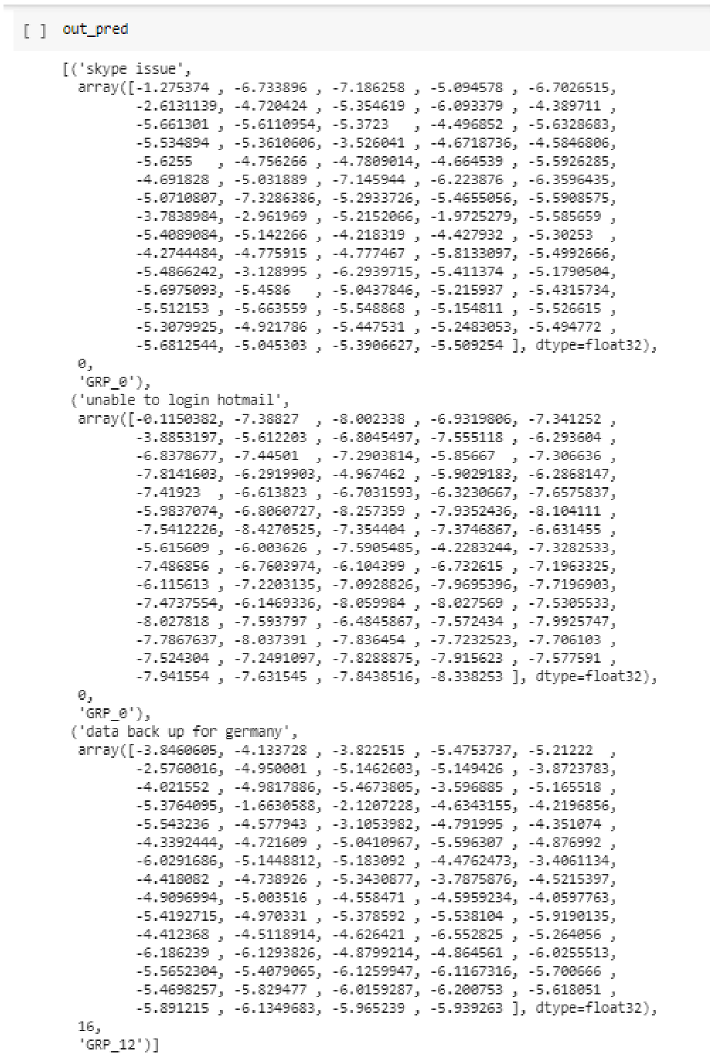


```python
def create_model(is_predicting, input_ids, input_mask, segment_ids, labels,
                 num_labels):

  bert_module = hub.Module(
      BERT_MODEL_HUB,
      trainable=True)
  bert_inputs = dict(
      input_ids=input_ids,
      input_mask=input_mask,
      segment_ids=segment_ids)
  bert_outputs = bert_module(
      inputs=bert_inputs,
      signature="tokens",
      as_dict=True)

  # Use "pooled_output" for classification tasks on an entire sentence.
  # Use "sequence_outputs" for token-level output.
  output_layer = bert_outputs["pooled_output"]

  hidden_size = output_layer.shape[-1].value

  # Create our own layer to tune for politeness data.
  output_weights = tf.get_variable(
      "output_weights", [num_labels, hidden_size],
      initializer=tf.truncated_normal_initializer(stddev=0.02))

  output_bias = tf.get_variable(
      "output_bias", [num_labels], initializer=tf.zeros_initializer())

  with tf.variable_scope("loss"):

    # Dropout helps prevent overfitting
    output_layer = tf.nn.dropout(output_layer, keep_prob=0.9)

    logits = tf.matmul(output_layer, output_weights, transpose_b=True)
    logits = tf.nn.bias_add(logits, output_bias)
    log_probs = tf.nn.log_softmax(logits, axis=-1)

    # Convert labels into one-hot encoding
    one_hot_labels = tf.one_hot(labels, depth=num_labels, dtype=tf.float32)

    predicted_labels = tf.squeeze(tf.argmax(log_probs, axis=-1, output_type=tf.int32))
    # If we're predicting, we want predicted labels and the probabiltiies.
    if is_predicting:
      return (predicted_labels, log_probs)

    # If we're train/eval, compute loss between predicted and actual label
    per_example_loss = -tf.reduce_sum(one_hot_labels * log_probs, axis=-1)
    loss = tf.reduce_mean(per_example_loss)
    return (loss, predicted_labels, log_probs)
```

1. *With Raw dataset*

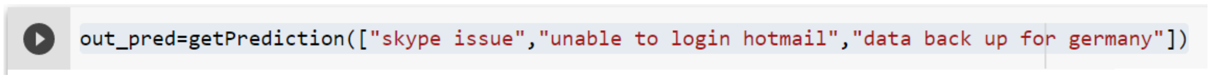


```
{'eval_accuracy': 0.6625,
 'false_negatives': 12.0,
 'false_positives': 2.0,
 'global_step': 733,
 'loss': 1.5365076,
 'true_negatives': 26.0,
 'true_positives': 40.0}
```



```
[ ]  out_pred
```

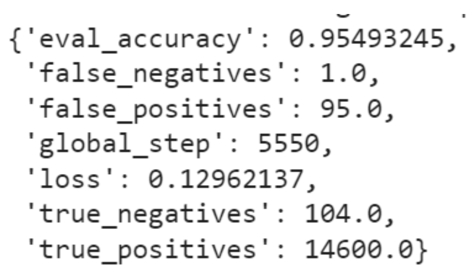
```
[('skype issue',
  array([-1.275374 , -6.733896 , -7.186258 , -5.094578 , -6.7026515,
         -2.6131139, -4.720424 , -5.354619 , -6.093379 , -4.389711 ,
         -5.661301 , -5.6110954, -5.3723    , -4.496852 , -5.6328683,
         -5.534894 , -5.3610606, -3.526041 , -4.6718736, -4.5846806,
         -5.6255   , -4.756266 , -4.7809014, -4.664539 , -5.5926285,
         -4.691828 , -5.031889 , -7.145944 , -6.223876 , -6.3596435,
         -5.0710807, -7.3286386, -5.2933726, -5.4655056, -5.5908575,
         -3.7838984, -2.961969 , -5.2152066, -1.9725279, -5.585659 ,
         -5.4089084, -5.142266 , -4.218319 , -4.427932 , -5.30253  ,
         -4.2744484, -4.775915 , -4.777467 , -5.8133097, -5.4992666,
         -5.4866242, -3.128995 , -6.2939715, -5.411374 , -5.1790504,
         -5.6975093, -5.4586   , -5.0437846, -5.215937 , -5.4315734,
         -5.512153 , -5.663559 , -5.548868 , -5.154811 , -5.526615 ,
         -5.3079925, -4.921786 , -5.447531 , -5.2483053, -5.494772 ,
         -5.6812544, -5.045303 , -5.3906627, -5.509254 ], dtype=float32),
  0,
  'GRP_0'),
 ('unable to login hotmail',
  array([-0.1150382, -7.38827  , -8.002338 , -6.9319806, -7.341252 ,
         -3.8853197, -5.612203 , -6.8045497, -7.555118 , -6.293604 ,
         -6.8378677, -7.44501  , -7.2903814, -5.85667  , -7.306636 ,
         -7.8141603, -6.2919903, -4.967462 , -5.9029183, -6.2868147,
         -7.41923  , -6.613823 , -6.7031593, -6.3230667, -7.6575837,
         -5.9837074, -6.8060727, -8.257359 , -7.9352436, -8.104111 ,
         -7.5412226, -8.4270525, -7.354404 , -7.3746867, -6.631455 ,
         -5.615609 , -6.003626 , -7.5905485, -4.2283244, -7.3282533,
         -7.486856 , -6.7603974, -6.104399 , -6.732615 , -7.1963325,
         -6.115613 , -7.2203135, -7.0928826, -7.9695396, -7.7196903,
         -7.4737554, -6.1469336, -8.059984 , -8.027569 , -7.5305533,
         -8.027818 , -7.593797 , -6.4845867, -7.572434 , -7.9925747,
         -7.7867637, -8.037391 , -7.836454 , -7.7232523, -7.706103 ,
         -7.524304 , -7.2491097, -7.8288875, -7.915623 , -7.577591 ,
         -7.941554 , -7.631545 , -7.8438516, -8.338253 ], dtype=float32),
  0,
  'GRP_0'),
 ('data back up for germany',
  array([-3.8460605, -4.133728 , -3.822515 , -5.4753737, -5.21222  ,
         -2.5760016, -4.950001 , -5.1462603, -5.149426 , -3.8723783,
         -4.021552 , -4.9817886, -5.4673805, -3.596885 , -5.165518 ,
         -5.3764095, -1.6630588, -2.1207228, -4.6343155, -4.2196856,
         -5.543236 , -4.577943 , -3.1053982, -4.791995 , -4.351074 ,
         -4.3392444, -4.721609 , -5.0410967, -5.596307 , -4.876992 ,
         -6.0291686, -5.1448812, -5.183092 , -4.4762473, -3.4061134,
         -4.418082 , -4.738926 , -5.3430877, -3.7875876, -4.5215397,
         -4.9096994, -5.003516 , -4.558471 , -4.5959234, -4.0597763,
         -5.4192715, -4.970331 , -5.378592 , -5.538104 , -5.9190135,
         -4.412368 , -4.5118914, -4.626421 , -6.552825 , -5.264056 ,
         -6.186239 , -6.1293826, -4.8799214, -4.864561 , -6.0255513,
         -5.5652304, -5.4079065, -6.1259947, -6.1167316, -5.700666 ,
         -5.4698257, -5.829477 , -6.0159287, -6.200753 , -5.618051 ,
         -5.891215 , -6.1349683, -5.965239 , -5.939263 ], dtype=float32),
  16,
  'GRP_12')]
```



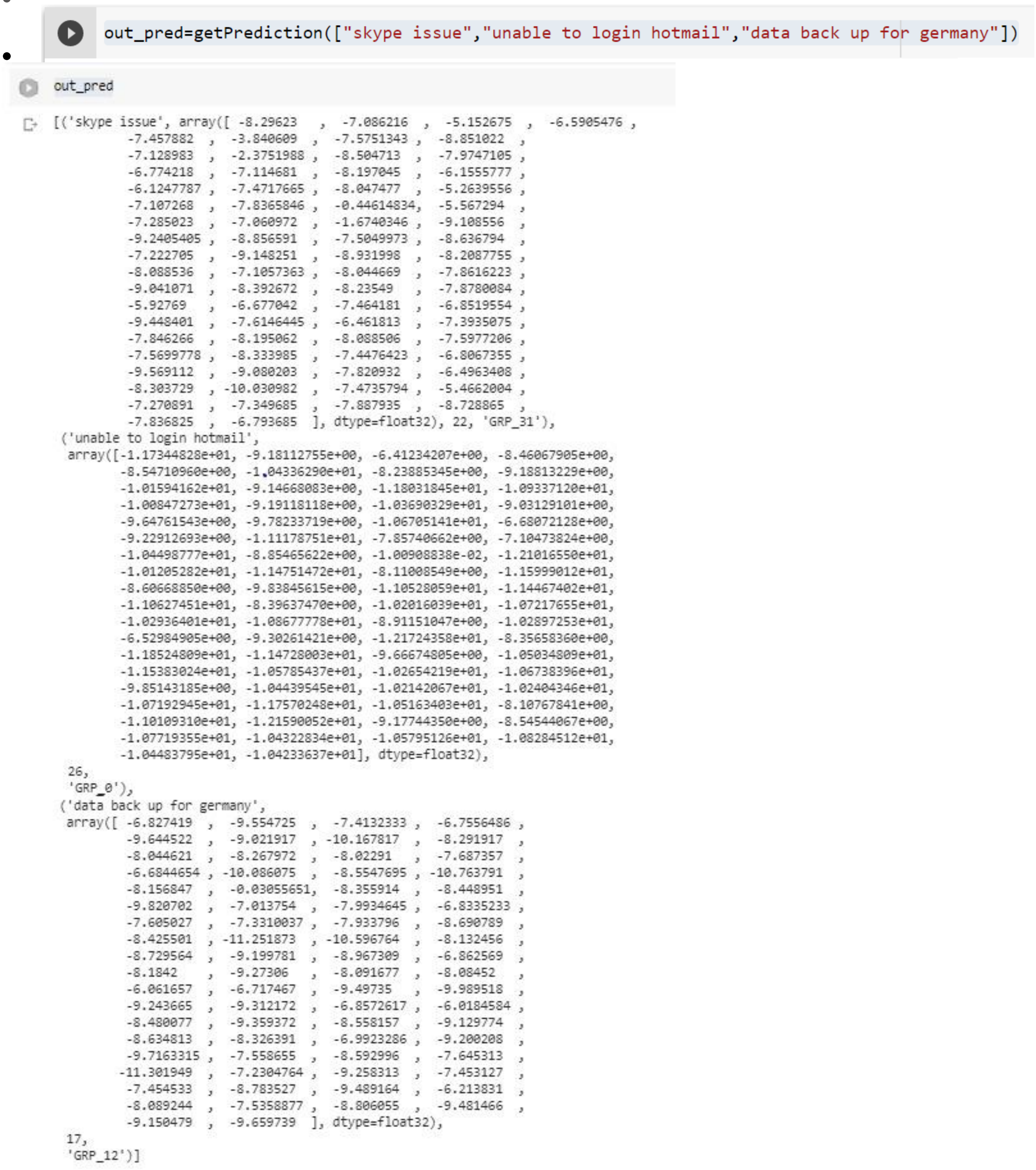
```
out_pred=getPrediction(["skype issue","unable to login hotmail","data back up for germany"])
```

`

2. *With up sampling*



```
{'eval_accuracy': 0.95493245,
 'false_negatives': 1.0,
 'false_positives': 95.0,
 'global_step': 5550,
 'loss': 0.12962137,
 'true_negatives': 104.0,
 'true_positives': 14600.0}
```



```
out_pred=getPrediction(["skype issue","unable to login hotmail","data back up for germany"])
```

```
out_pred
```

```
[('skype issue', array([ -8.29623   ,  -7.086216  ,  -5.152675  ,  -6.5905476 ,
        -7.457882  ,  -3.840609  ,  -7.5751343 ,  -8.851022  ,
        -7.128983  ,  -2.3751988 ,  -8.504713  ,  -7.9747105 ,
        -6.774218  ,  -7.114681  ,  -8.197045  ,  -6.1555777 ,
        -6.1247787 ,  -7.4717665 ,  -8.047477  ,  -5.2639556 ,
        -7.107268  ,  -7.8365846 ,  -0.44614834,  -5.567294  ,
        -7.285023  ,  -7.060972  ,  -1.6740346 ,  -9.108556  ,
        -9.2405405 ,  -8.856591  ,  -7.5049973 ,  -8.636794  ,
        -7.222705  ,  -9.148251  ,  -8.931998  ,  -8.2087755 ,
        -8.088536  ,  -7.1057363 ,  -8.044669  ,  -7.8616223 ,
        -9.041071  ,  -8.392672  ,  -8.23549   ,  -7.8780084 ,
        -5.92769   ,  -6.677042  ,  -7.464181  ,  -6.8519554 ,
        -9.448401  ,  -7.6146445 ,  -6.461813  ,  -7.3935075 ,
        -7.846266  ,  -8.195062  ,  -8.088506  ,  -7.5977206 ,
        -7.5699778 ,  -8.333985  ,  -7.4476423 ,  -6.8067355 ,
        -9.569112  ,  -9.080203  ,  -7.820932  ,  -6.4963408 ,
        -8.303729  , -10.030982  ,  -7.4735794 ,  -5.4662004 ,
        -7.270891  ,  -7.349685  ,  -7.887935  ,  -8.728865  ,
        -7.836825  ,  -6.793685  ], dtype=float32), 22, 'GRP_31'),
 ('unable to login hotmail',
  array([-1.17344828e+01, -9.18112755e+00, -6.41234207e+00, -8.46067905e+00,
        -8.54710960e+00, -1.04336290e+01, -8.23885345e+00, -9.18813229e+00,
        -1.01594162e+01, -9.14668083e+00, -1.18031845e+01, -1.09337120e+01,
        -1.00847273e+01, -9.19118118e+00, -1.03690329e+01, -9.03129101e+00,
        -9.64761543e+00, -9.78233719e+00, -1.06705141e+01, -6.68072128e+00,
        -9.22912693e+00, -1.11178751e+01, -7.85740662e+00, -7.10473824e+00,
        -1.04498777e+01, -8.85465622e+00, -1.00908838e-02, -1.21016550e+01,
        -1.01205282e+01, -1.14751472e+01, -8.11008549e+00, -1.15999012e+01,
        -8.60668850e+00, -9.83845615e+00, -1.10528059e+01, -1.14467402e+01,
        -1.10627451e+01, -8.39637470e+00, -1.02016039e+01, -1.07217655e+01,
        -1.02936401e+01, -1.08677778e+01, -8.91151047e+00, -1.02897253e+01,
        -6.52984905e+00, -9.30261421e+00, -1.21724358e+01, -8.35658360e+00,
        -1.18524809e+01, -1.14728003e+01, -9.66674805e+00, -1.05034809e+01,
        -1.15383024e+01, -1.05785437e+01, -1.02654219e+01, -1.06738396e+01,
        -9.85143185e+00, -1.04439545e+01, -1.02142067e+01, -1.02404346e+01,
        -1.07192945e+01, -1.17570248e+01, -1.05163403e+01, -8.10767841e+00,
        -1.10109310e+01, -1.21590052e+01, -9.17744350e+00, -8.54544067e+00,
        -1.07719355e+01, -1.04322834e+01, -1.05795126e+01, -1.08284512e+01,
        -1.04483795e+01, -1.04233637e+01], dtype=float32),
  26,
  'GRP_0'),
 ('data back up for germany',
  array([ -6.827419  ,  -9.554725  ,  -7.4132333 ,  -6.7556486 ,
        -9.644522  ,  -9.021917  , -10.167817  ,  -8.291917  ,
        -8.044621  ,  -8.267972  ,  -8.02291   ,  -7.687357  ,
        -6.6844654 , -10.086075  ,  -8.5547695 , -10.763791  ,
        -8.156847  ,  -0.03055651,  -8.355914  ,  -8.448951  ,
        -9.820702  ,  -7.013754  ,  -7.9934645 ,  -6.8335233 ,
        -7.605027  ,  -7.3310037 ,  -7.933796  ,  -8.690789  ,
        -8.425501  , -11.251873  , -10.596764  ,  -8.132456  ,
        -8.729564  ,  -9.199781  ,  -8.967309  ,  -6.862569  ,
        -8.1842    ,  -9.27306   ,  -8.091677  ,  -8.08452   ,
        -6.061657  ,  -6.717467  ,  -9.49735   ,  -9.989518  ,
        -9.243665  ,  -9.312172  ,  -6.8572617 ,  -6.0184584 ,
        -8.480077  ,  -9.359372  ,  -8.558157  ,  -9.129774  ,
        -8.634813  ,  -8.326391  ,  -6.9923286 ,  -9.200208  ,
        -9.7163315 ,  -7.558655  ,  -8.592996  ,  -7.645313  ,
       -11.301949  ,  -7.2304764 ,  -9.258313  ,  -7.453127  ,
        -7.454533  ,  -8.783527  ,  -9.489164  ,  -6.213831  ,
        -8.089244  ,  -7.5358877 ,  -8.806055  ,  -9.481466  ,
        -9.150479  ,  -9.659739  ], dtype=float32),
  17,
  'GRP_12')]
```

CONCLUSION:

Amongst all the models that we have run on the given dataset, BiLSTM with Glove has given us the maximum accuracy. We also 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.