AUTOMATIC TICKET CLASSIFICATION

-- INTERIM REPORT--

MENTOR:

SURVESH CHAUHAN

STUDENTS:

BRIJESH ARORA

GULAB S

SHIVAKUMAR RAMAN

SWAMINATHAN KANNAN

CONTENTS

WHAT IS TICKET CLASSIFICATION?

When an issue or support incident ticket receives, first it needs to be processed and assigned to specific category so that it's routed to the correct team member. This involves reading and classify the ticket.

WHY IT IS IMPORTANT?

Lot of time need to spend opening, reading, responding or deleting, and categorising on each incident tickets. Time that could be better spent on more fulfilling tasks. To manage all data without spending hours and hours sorting it manually, need to use ticket classification tools that will automatically sort and classify.

OBJECTIVE

This interim report presents to build a classifier by powerful AI techniques that can classify incidents to right functional groups can help organizations to reduce the resolving time of the issue and can focus on more productive tasks.

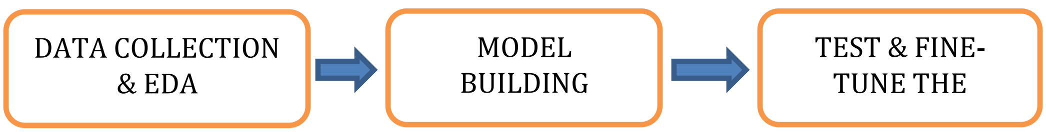
PROBLEM STATEMENT

One of the key activities of any IT function is to "Keep the lights on" to ensure there is no impact to the Business operations. IT leverages Incident Management process to achieve the above Objective. An incident is something that is unplanned interruption to an IT service or reduction in the quality of an IT service that affects the Users and the Business. The main goal of Incident Management process is to provide a quick fix / workarounds or solutions that resolves the interruption and restores the service to its full capacity to ensure no business impact. In most of the organizations, incidents are created by various Business and IT Users, End Users/ Vendors if they have access to ticketing systems, and from the integrated monitoring systems and tools. Assigning the incidents to the appropriate person or unit in the support team has critical importance to provide improved user satisfaction while ensuring better allocation of support resources. The assignment of incidents to appropriate IT groups is still a manual process in many of the IT organizations. Manual assignment of incidents is time consuming and requires human efforts. There may be mistakes due to human errors and resource consumption is carried out ineffectively because of the misaddressing. On the other hand, manual assignment increases the response and resolution times which result in user satisfaction deterioration / poor customer service.

BUSINESS DOMAIN VALUE

In the support process, incoming incidents are analyzed and assessed by organization's support teams to fulfill the request. In many organizations, better allocation and effective usage of the valuable support resources will directly result in substantial cost savings. Currently the incidents are created by various stakeholders (Business Users, IT Users and Monitoring Tools) within IT Service Management Tool and are assigned to Service Desk teams (L1 / L2 teams). This team will review the incidents for right ticket categorization, priorities and then carry out initial diagnosis to see if they can resolve. Around ~54% of the incidents are resolved by L1 / L2 teams. Incase L1 / L2 is unable to resolve, they will then escalate / assign the tickets to Functional teams from Applications and Infrastructure (L3 teams). Some portions of incidents are directly assigned to L3 teams by Monitoring tools or Callers / Requestors. L3 teams will carry out detailed diagnosis and resolve the incidents. Around ~56% of incidents are resolved by Functional / L3 teams. Incase if vendor support is needed, they will reach out for their support towards incident closure.L1 / L2 needs to spend time reviewing Standard Operating Procedures (SOPs) before assigning to Functional teams (Minimum ~25-30% of incidents needs to be reviewed for SOPs before ticket assignment). 15 min is being spent for SOP review for each incident. Minimum of ~1 FTE effort needed only for incident assignment to L3 teams. During the process of incident assignments by L1 / L2 teams to functional groups, there were multiple instances of incidents getting assigned to wrong functional groups. Around ~25% of Incidents are wrongly assigned to functional teams. Additional effort needed for Functional teams to re-assign to right functional groups. During this process, some of the incidents are in queue and not addressed timely resulting in poor customer service. Guided by powerful AI techniques that can classify incidents to right functional groups can help organizations to reduce the resolving time of the issue and can focus on more productive tasks.

PROCESS OVERVIEW



MILESTONE 1: PRE-PROCESSING, DATA VISUALISATION AND EDA

OVERVIEW

1) Exploring the given Data files
2) Understanding the structure of data
3) Missing points in data
4) Finding inconsistencies in the data
5) Visualizing different patterns
6) Visualizing different text features
7) Dealing with missing values
8) Text preprocessing
9) Creating word vocabulary from the corpus of report text data
10) Creating tokens as required

MILESTONE 2: MODEL BUILDING

OVERVIEW

1) Building a model architecture which can classify.
2) Trying different model architectures by researching state of the art for similar tasks.
3) Train the model
4) To deal with large training time, save the weights so that you can use them when training the model for the second time without starting from scratch.

MILESTONE 3: TEST THE MODEL, FINE-TUNING AND REPEAT

OVERVIEW

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

DATASET

NAME: *input_data.xlsx*

No. Of. Columns: *4*

Name of Columns: *1. Short description 2. Description 3. Caller 4. Assignment group*

Google Drive Link:

https://drive.google.com/file/d/1OZNJm81JXucV3HmZroMq6qCT2m7ez7IJ

USED LIBRARIES

- ✓ NUMPY
- ✓ PANDAS
- ✓ SEABORN
- ✓ MATPLOTLIB
- ✓ SKLEARN
- ✓ RE (REG EX)
- ✓ SPACY
- ✓ NLTK
- ✓ os
- ✓ zipfile
- ✓ cv2
- ✓ TENSORFLOW
- ✓ KERAS
- ✓ GENSIM
- ✓ WORD2VEC
- ✓ DOC2VEC
- ✓ Google API (FOR LANGUAGE TRANSLATION)

1. SUMMARY OF PROBLEM STATEMENT, DATA AND FINDINGS

The assignment of incidents to appropriate IT groups is still a manual process in many of the IT organizations. Manual assignment of incidents is time consuming and requires human efforts. There may be mistakes due to human errors and resource consumption is carried out ineffectively because of the misaddressing. Almost 30–40% of incident tickets are not routed to the right team .On the other hand, manual assignment increases the response and resolution times which result in user satisfaction deterioration / poor customer service.

DATA:

We used the given dataset (input_data.xlsx) and tried to classify those using AI/ML techniques. We just kept the Short description, Description from the ticket and went ahead with analytics on it to come up with Assignment Group for incidents.

.

2. SUMMARY OF THE APPROACH TO EDA & PRE-PROCESSING

The provided data file comprises of 4 columns

1. Short description (this gives the short description of the ticket)
2. Description (A longer explanation of the ticket)
3. Caller (The person who issued the ticket)
4. Assignment group (The Group which finally answered the ticket)

The total no. of records in the database is 8500.

The objective of the exercise is to build an accurate model using the first 3 variables to predict to which group the ticket should be directed to.

THE KEY POINTS THAT THE EDA THREW UP:

1. There are multiple languages present in the description columns ; thus the first step was to translate all these languages into English. Google Sheets was used for the translation.
2. The need to concatenate the Short Description and Description Columns.
    a. In quite a few records only “the” was appearing in the Description Column (this would get eliminated in the removal of Stop words process and hence these would become blank)
    b. In more than a fourth of the records (2889) , the Short description and Description columns had exactly the same text
    c. There were minimal Null values in both these columns but no cases where both were Null (Null values were replaced with blank string)

The Assignment Group column has 74 groups of which 39 groups has a record base of less than 30 (indicating that it would be better to merge these groups into a single group). These 39 groups accounted for 4.2% of the total record base

PRE-PROCESSING

1. All text was converted to lower case
2. Numbers were removed
3. Punctuations, spaces, indents, etc were removed
4. Stop words were removed (including custom stop words)
5. E-mail id's mentioned were removed
6. Caller name was removed where present (by comparing with Caller Column)
7. Post the above, lemmatization was done

Once the above was done, the total vocabulary size was 10327 words

N-grams (uni-gram, bi-gram and tri-grams) were generated to understand the corpus better. Word associations were explored to check how intuitive the Word2Vec model was and whether it made sense.

Word embedding was then done with both Glove as well as Word2Vec as input into the Neural Network.

3. DECIDING MODELS AND MODEL BUILDING

Based on the nature of the problem, decide what algorithms will be suitable and why?

Experiment with different algorithms and get the performance of each algorithm.

As we have seen problem of classification type (i.e many to one) - based on ticket description it need to be assigned to appropriate group.

Bidirectional LSTM model will be suitable here because

- it can preserve information from both past and future at any point in time so it can understand the context better.
- due to above mentioned it can give better performance on sequence classification problems in NLP.

For pre-processing, we will use below to convert words to embedding's and check model performance with both.

- Glove
- Word2Vec

In order to avoid memory issue, we will not use TF-IDF feature here.

Here is the model performance results:

- Glove - validation accuracy is 66%
- Word2Vec - validation accuracy is 59%

In the next step 4, we will see how model performance can be improved further.

4. IMPROVE YOUR MODEL PERFORMANCE

How to improve your model performance? What are the approaches you can take to improve your model? Can you do some feature selection, data manipulation and model improvements.

Currently, we have used a 200-dimension Glove embedding on a 2-layer bi-directional LSTM model that provides an overall accuracy of 53.59% in 20 epochs. This model also provides us with an f-1 score of 0.83 for the largest group. However, we also observed a few classes with a 0 f1-score

We believe that, going forward, in the next phase of the project, we will have to improve the model across multiple areas:

1. **Input data transformation:**
    a. We will need to look at additional ways of further clubbing various disparate groups. 35 seem like a large number of target groups for a classification model.
    b. We will assess whether up-sampling of the smaller groups or down-sampling of the larger groups will need to be done. With Group 0 having 46% of all samples, it is able to provide us with a satisfactory f1-score of 0.83. However the f1-score is lackluster in all other groups
    c. Explore possibility of applying an SVD to the input data to identify and remove correlated dimensions
    d. Further clean out names and surnames. Since many of the descriptions are in the email format, there are a few names in cc which are not part of the original user names column. In a few cases, while testing similarities in our Word2Vec model (not included in the deliverable), these names did pop out as a relevant term
2. **Word embedding's:**
    a. We have currently used the Glove 200d data model to prepare our embedding's. We will further explore the following word embedding's to assess improvement to our model:
        i. Glove 100d and 300d word embedding's
        ii. Word2Vec unigrams, bi-grams and tri-grams
        iii. Word2Vec formats such as Google News vectors
        iv. Elmo word embedding's

b. We will run all these embedding's on the same base model and identify the appropriate word embedding's to use prior to tuning our model.

3. **Models:**
   a. We have currently used a bi-directional 2-layer RNN model as the reference model for this interim milestone.
   b. Going forward, we will fine tune the existing model based on multiple hyper parameters. This will include:
      i. Number of layers in the model
      ii. Dropout rates (in case over fitting is not observed)
      iii. Increase number of nodes / layer in our model
      iv. Assess impact of learning rates on the model
   c. We will also explore the feasibility of deploying transformer models such as BERT and RoBERTa for our model to see if the same results in higher scores.
   d. We will plot our results based on epoch-wise, loss and accuracy of our train and test models
4. **Housekeeping:**
   a. As part of the overall governance of the project, we will ensure adequate work distribution and timely updates with our mentor and Divya to ensure that everyone gets an adequate opportunity to participate in the same.

DETAILED REPORT WITH CODE SNIPPET

MILESTONE 1: PRE-PROCESSING, DATA VISUALISATION AND EDA

DATA PRE-PROCESSING AND EDA

IMPORTING REQUIRED LIBRARIES

```
!pip install spacy
!pip install nltk
pip install plotly==4.7.1
pip install cufflinks
!pip install gensim
```

```
# Utilities
import os
import zipfile, warnings
import sys
!{sys.executable} -m spacy download en
from time import time
import cv2

# Numerical calculation
import numpy as np

# Data Handling
import pandas as pd

# Tools & Evaluation metrics
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder,scale
from sklearn.metrics import confusion_matrix, classification_report, roc_curve, auc, accuracy_score, precision_recall_curve
from sklearn.manifold import TSNE

# Data Visualization
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
from bokeh.plotting import figure, show, output_notebook
from bokeh.models import HoverTool, ColumnDataSource, value
output_notebook()
import cufflinks
import plotly as py
import plotly.graph_objs as go
import plotly.figure_factory as ff
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
cufflinks.go_offline()
cufflinks.set_config_file(world_readable=True, theme='pearl')
```

```python
import re # for applying Regex pattern to subject strings

# NLP toolkits
import spacy
import nltk as nltk
from nltk.corpus import stopwords
nltk.download('stopwords')
from nltk.tokenize import word_tokenize
nltk.download('punkt')
import gensim
import gensim.corpora as corpora
from gensim.utils import simple_preprocess
from gensim.parsing import preprocessing
from gensim.test.utils import common_texts
from gensim.models.doc2vec import Doc2Vec, TaggedDocument
from gensim.models.phrases import Phraser
from gensim.models import Phrases, CoherenceModel
from gensim.models import Word2Vec

# Sequential Modeling
import tensorflow as tf
from tensorflow import keras
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import *
from keras.utils.np_utils import to_categorical
from keras.initializers import Constant
import keras.backend as K
from keras import initializers
from keras.engine.topology import Layer, InputSpec
from keras.models import Model, Sequential
from keras.layers import Dense, LSTM, TimeDistributed, Conv1D, MaxPooling1D
from keras.layers import Embedding, Activation, Dropout, Flatten, Bidirectional
from keras.layers import Permute, merge, Input, multiply, concatenate
from keras.callbacks import Callback, ModelCheckpoint, EarlyStopping
from keras.constraints import max_norm, unit_norm
from keras.preprocessing.text import Tokenizer, text_to_word_sequence
from keras.preprocessing.sequence import pad_sequences
```

LOAD THE DATASET AND SEE SOME RECORDS

```python
df = pd.read_csv('input_data.csv')
df.head()
```

	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

SHAPE OF THE DATASET

```
df.shape
```

```
(8500, 4)
```

There are 8500 rows and 4 columns in the dataset.

EYEBALL THE DATA FOR RANDOM ROWS

Before Moving Onto EDA, It's Useful To Eyeball The Data For Random Rows

```
random_state=1
random_subset = df.sample(n=50)
print(random_subset)
```

```
                                       Short description  \
7809  international payments rejected for payment me...
5235                            bobj - erp business objects
4689                                 erp issue with batches
1099                                 finance portal on the hub
2785                                       crm installation
8005  please move client from office 2010 to 2016 pe...
5567  circuit outage : apac,apac-dmvpn circuit is do...
2119                                 erp SID_1 account unlock
8232  stepfhryhan needs access to below collaboratio...
1119                                     outlook not updating
5080                                   skype meeting code/pin
7932                  abended job in job_scheduler: Job_3182
7842  outlook issues,yesterday my outlook took about...
6893                           owa installation in mobile device
2645                          ms crm dynamics : outlook issue
5232                                 erp access issue bex hana
3838               erp user profile for pthsqroz moedyanvess
5918  job bk_hana_SID_61_erp_wky_dp failed in job_sc...
```

OBSERVATIONS FROM EYE BALLING:

Row 1081 has perhaps chinese characters; Row 1537 seems to be German - Thus the database comprises of Descriptions in Multiple Languages and we would need to translate these into English

Further the Description column has words like "received from", "from", "hello", "hallo" and perhaps other such words that do not add any value to the analysis and hence they need to be removed as part of the Stop words

Also the email id of the caller is given in the Description column (for e.g. Row 1976, 7456) and it is unlikely that would add value to the analysis and hence most likely these would need to be removed

LANGUAGE TRANSLATION

Given the problems with using Google API from within Python, Google Sheets was used for the translation. The Excel sheet was opened in Google Sheets and by applying the formulae "=DetectLanguage(Cell)" and "=GoogleTranslate(Cell, "Auto", en)", we were able to understand the language in which the text was as well as get the translation of the text to English respectively and the translated file was downloaded. From this point onward, the translated Excel sheet has been used and this Sheet is also being submitted as part of the project

```
new_df = pd.read_csv('input_data_translated.csv', encoding='mac_roman')
new_df.head()
```

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

CHECKING FOR TRANSLATION

```
print(df['Description'][1081])
print(new_df['Description'][1081])

print(df['Description'][1537])
print(new_df['Description'][1537])
```

```
打开已关闭的销售订单时，显示"不能在手动或分布事物方式下创建新的链接"
When you open sales orders have been closed, display "can not create new links in manual or distributed way things are."
probleme mit lan am rechner \ we_wu113 \ essa \wrcktgbd wzrgyunp
problems with lan on computer \ we_wu113 \ essa \ wrcktgbd wzrgyunp
```

The above shows that the Chinese and German have been translated into English

EXPLORATORY DATA ANALYSIS (EDA)

Understanding of the columns is important in order to understand whether to retain/ drop some of them.

CALLER COLUMN

This perhaps indicates the person who filed the ticket; it would be useful to understand the number of such callers (If there are many unique callers with low sample bases, then it might not make sense to include this column

```
new_df.Caller.nunique() # Understanding the unique no. of callers
```

```
2950
```

Thus across 8500 Rows, there are 2950 unique callers and hence each caller is unlikely to have a sufficient base for analysis. However would be interesting to check the counts of at least the callers who put in tickets more frequently.

VALUE COUNTS OF EACH COLUMN

```
new_df.Caller.value_counts()
```

```
bpctwhsn kzqsbmtp    810
ZkBogxib QsEJzdZO    151
fumkcsji sarmtlhy    134
rbozivdq gmlhrtvp     87
rkupnshb gsmzfojw     71
jloygrwh acvztedi     64
spxqmiry zpwgoqju     63
oldrctiu bxurpsyi     57
olckhmvx pcqobjnd     54
jyoqwxhz clhxsoqy     51
dkmcfreg anwmfvlg     51
efbwiadp dicafxhv     45
afkstcev utbnkyop     32
gzhapcld fdigznbk     30
mnlazfsr mtqrkhnx     28
uvrbhlnt bjrmalzi     27
entuakhp xrnhtdmk     25
jionmpsf wnkpzcmv     24
vzqomdgt jwoqbuml     24
bozdftwx smylqejw     23
utyeofsk rdyzpwhi     21
rxoynvgi ntgdsehl     21
qasdhyzm yuglsrwx     21
```

There are only 11 callers who have a sample base > 50; While the top caller has 810 records, the sample per caller diminishes rapidly; thus there does not seem to be any value in retaining this column and it will be dropped.

DESCRIPTION OF THE EACH COLUMN

```
new_df.describe().transpose()
```

	count	unique	top	freq
Short description	8498	7386	password reset	48
Description	8497	7751	the	56
Caller	8500	2950	bpctwhsn kzqsbmtp	810
Assignment group	8500	74	GRP_0	3976
Language	8497	27	en	7733

Above description says below observations:

1. 'Short description' column has 2 blank values and 'Desciption' column has 3 blank values
2. Out of 8498 there are 7386 unique values in 'Short description' column, out of 8497 there are 7751 unique values in 'Description' column and out of 8500 there are 74 unique values in 'Assignment group' column which is target column. Also there are 27 unique languages in the database
3. "Password reset" is the top frequent value with 48 occurrences in 'Short description' column, "the" is the top frequent value with 56 occurrences in 'Description' column and "GRP_0" is the group which has maximum assignment of tickets i.e 3976. English is the most common language with 7733 records

Comments: In the Description Column 'the' by itself does not add any value since it would get removed as a stop word; hence useful to see what is Short Description that is in the rows with "the" in the Description Column

ROWS WITH "THE" IN THE DESCRIPTION COLUMN

To See What Is Short Description That Is In The Rows With "The" In The Description Column

```
new_df[new_df.Description == 'the'].head(10)
```

	Short description	Description	Caller	Assignment group	Language
1049	reset passwords for soldfnbq uhnbsvqd using pa...	the	soldfnbq uhnbsvqd	GRP_17	en
1054	reset passwords for fygrwuna gomcekzi using pa...	the	fygrwuna gomcekzi	GRP_17	en
1144	reset passwords for wvdxnkhf jirecvta using pa...	the	wvdxnkhf jirecvta	GRP_17	en
1184	reset passwords for pxvjczdt kizsjfpq using pa...	the	pxvjczdt kizsjfpq	GRP_17	en
1292	reset passwords for cubdsrml znewqgop using pa...	the	cubdsrml znewqgop	GRP_17	en
1476	reset passwords for bnoupaki cpeioxdz using pa...	the	bnoupaki cpeioxdz	GRP_17	en
1558	reset passwords for usa feathers using passwor...	the	lmqysdec ljvbnpqw	GRP_17	en
1693	reset passwords for eglavnhx uprodleq using pa...	the	eglavnhx uprodleq	GRP_17	en
1834	reset passwords for hybiaxlk lawptzir using pa...	the	hybiaxlk lawptzir	GRP_17	en
1850	reset passwords for fylrosuk kedgmiul using pa...	the	fylrosuk kedgmiul	GRP_17	en

Most of the "the" in Description seem to be associated with "reset passwords" in the Short Description Column; thus in order to develop a reasonable model it points to the fact that it would be good to concatenate both these columns

CHECK THE DATA TYPE OF COLUMNS

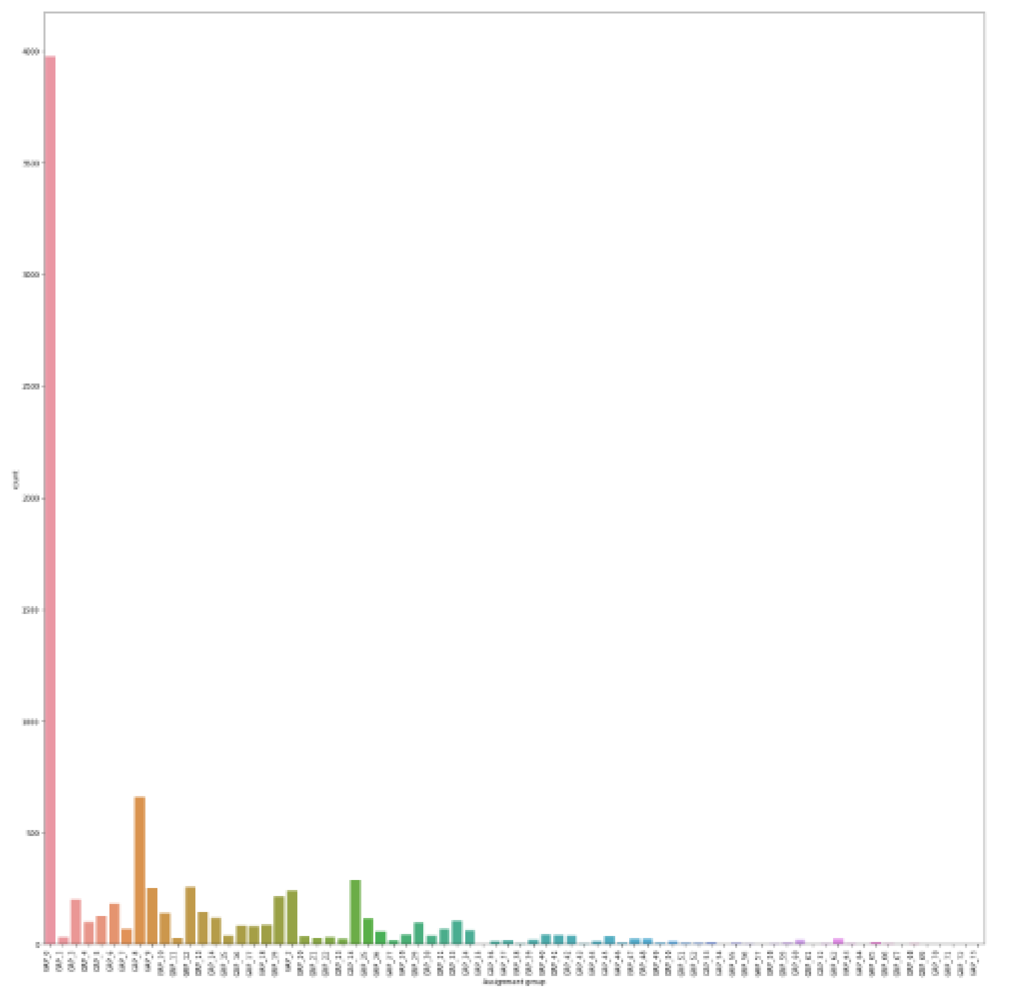
```
new_df.dtypes
```

```
Short description    object
Description          object
Caller               object
Assignment group     object
Language             object
dtype: object
```

UNDERSTANDING THE TARGET COLUMN

Let's see number of tickets assigned to each group

```
fig_dims = (20, 20)
fig, ax = plt.subplots(figsize=fig_dims)
chart=sns.countplot(x = "Assignment group", ax=ax, data=new_df)
chart.set_xticklabels(chart.get_xticklabels(), rotation=90, ha="right")
plt.tight_layout()
plt.show()
```



SEE NUMBER OF TICKETS ASSIGNED TO EACH GROUP

```
print(new_df['Assignment group'].value_counts())
```

```
GRP_0     3976
GRP_8      661
GRP_24     289
GRP_12     257
GRP_9      252
GRP_2      241
GRP_19     215
GRP_3      200
GRP_6      184
GRP_13     145
GRP_10     140
GRP_5      129
GRP_14     118
GRP_25     116
GRP_33     107
GRP_4      100
GRP_29      97
GRP_18      88
GRP_16      85
GRP_17      81
GRP_31      69
GRP_7       68
GRP_34      62
GRP_26      56
GRP_40      45
```

The Group variable has a very long tail and it might not be beneficial to keep the ones that have a low record base; One could potentially club the groups with low record base into 1 group called as "Others". Classic statistics state that beyond a sample of 30, distributions tend to be normal and thus this value could be set as a cut-off for a group to be included and all groups with a base of <30 can be combined into "Others".

CHECK FOR NULL VALUES

```
new_df.isnull().apply(pd.value_counts)
```

	Short description	Description	Caller	Assignment group	Language
False	8498	8497	8500.0	8500.0	8497
True	2	3	NaN	NaN	3

There are 2 null values in 'Short description' column and 3 in 'Description' column.There is no null value in target column.

If there are rows, with Null values for both, then we would need to remove them; however we need to check for the same.

ROWS WITH NULL VALUES

```
# Let's look at the rows with null values
new_df[pd.isnull(new_df).any(axis=1)]
```

	Short description	Description	Caller	Assignment group	Language
2604	NaN	\n\nreceived from: ohdrnswl.rezuibdt@gmail.com...	ohdrnswl rezuibdt	GRP_34	en
3383	NaN	\n-connected to the user system using teamview...	qftpazns fxpnytmk	GRP_0	en
4395	i am locked out of skype	NaN	viyglzfo ajtfzpkb	GRP_0	NaN
6371	authorization add/delete members	NaN	hpmwliog kqtnfvrl	GRP_0	NaN
7397	browser issue :	NaN	fgejnhux fnkymoht	GRP_0	NaN

There are no records which have Null values for both Short Description and Description; Given that for prediction it is likely that both Columns would be used, replacement of Null values can be done with a blank string (as it is likely that both Short Description and Description would be concatenated while building the model).

REPLACE NULL VALUES

```
# NULL replacement
new_df.fillna(str(), inplace=True)
new_df[pd.isnull(new_df).any(axis=1)]
```

	Short description	Description	Caller	Assignment group	Language

Now there is no null value in any column.

CHECKING FOR DUPLICATES ACROSS SHORT DESCRIPTION AND DESCRIPTION

```
df_common=new_df[new_df[["Short description","Description"]].apply(lambda x : x[0]==x[1],axis=1)].
```

```
.reset_index(drop=True).copy()
```

```
df_common.head()
```

	Short description	Description	Caller	Assignment group	Language
0	unable to access hr_tool page	unable to access hr_tool page	xbkucsvz gcpydteq	GRP_0	en
1	skype error	skype error	owlgqjme qhcozdfx	GRP_0	en
2	unable to log in to engineering tool and skype	unable to log in to engineering tool and skype	eflahbxn ltdgrvkz	GRP_0	en
3	ticket_no1550391- employment status - new non-...	ticket_no1550391- employment status - new non-...	eqzibjhw ymebpoih	GRP_0	en
4	unable to disable add ins on outlook	unable to disable add ins on outlook	mdbegvct dbvichlg	GRP_0	en

```
df_common.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2889 entries, 0 to 2888
Data columns (total 5 columns):
Short description    2889 non-null object
Description          2889 non-null object
Caller               2889 non-null object
Assignment group     2889 non-null object
Language             2889 non-null object
dtypes: object(5)
memory usage: 112.9+ KB
```

Comments: this shows that out of the 8500 records, 2889 have the same information in both Short description and Description - This would point out that it is better to concatenate the two columns for the modeling stage.

DROP THE LANGUAGE COLUMN AS IT IS NOT REQUIRED

```
new_df=new_df.drop('Language',axis=1)
new_df.shape
new_df.head()
```

	Short description	Description	Caller	Assignment group
0	login issue	-verified user details.(employee# & manager na...	spxjnwir pjlcoqds	GRP_0
1	outlook	\n\nreceived from: hmjdrvpb.komuaywn@gmail.com...	hmjdrvpb komuaywn	GRP_0
2	cant log in to vpn	\n\nreceived from: eylqgodm.ybqkwiam@gmail.com...	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

DATA UNDERSTANDING THUS FAR

1. All text has been converted into English
2. Need to concatenate both the short Description and Description Columns
3. Long tails in "Assignment group" column need to be taken care of
4. Null values solved for by replacing with blank string
5. Need to remove email id's from text

DATA CLEANING

In this step there is a need to

1. Convert all text to lower case
2. Remove numbers
3. Remove puntuations
4. Remove blank spaces
5. Remove stop words (along with other words identified earlier that would not contribute)
6. Remove email id's

```
import re # for applying Regex pattern to subject strings
```

```
# Fixing the different patterns
email_pat = r"([\w.+-]+@[a-z\d-]+\.[a-z\d.-]+)"
punct_pat = r"[,|.|_|@|\|?|\\|$&*|%|\r|\n|.:|\s+|/|//|\\|/|\||-|<|>|;|(|)|=|+|#|-|\"|[-\]]|{|}]"
num_pat = r"(?<!)(\d+(?:\.\d+)?)"

# Define a function to treat the texts
def preText(text):
    # Make the text unicase (lower)
    text = str(text).lower()
    # Remove email adresses
    text = re.sub(email_pat, ' ', text, flags=re.IGNORECASE)
    # Remove all numbers
    text = re.sub(r'\d+',' ',text)# remove numbers
    text = re.sub(num_pat, ' ', text)
    # Replace all punctuations with blank space
    text = re.sub(r'[^\w\s]', ' ', text)
    text = re.sub(punct_pat, " ", text, flags=re.MULTILINE)
    text = re.sub(r'\s+', ' ', text)
    # remove HTML tags
    text = re.sub('<.*?>', '', text)
    # Replace multiple spaces from prev step to single
    text = re.sub(r' {2,}', " ", text, flags=re.MULTILINE)
    text = text.replace('`',"'")
    return text.strip()
```

```
# Checking to see how the cleaning function has worked for a record
print('\033[1mOriginal text:\033[0m')
print(new_df['Description'][0])
print('_'*100)
print('\033[1mCleaned text:\033[0m')
print(preText(new_df['Description'][0]))
```

```
Original text:
-verified user details.(employee# & manager name)
-checked the user name in ad and reset the password.
-advised the user to login and check.
-caller confirmed that he was able to login.
-issue resolved.
____________________________________________________________________________________________________
Cleaned text:
verified user details employee manager name checked the user name in ad and reset the password advised the
user to login and check caller confirmed that he was able to login issue resolved
```

Now given that The text seems to have been pre-processed correctly.

APPLYING *IT* ON THE TOTAL DATABASE

```
new_df['Description'] = new_df['Description'].apply(preText)
new_df['Short description'] = new_df['Short description'].apply(preText)

# Verify the data
new_df.head()
```

	Short description	Description	Caller	Assignment group
0	login issue	verified user details employee manager name ch...	spxjnwir pjlcoqds	GRP_0
1	outlook	received from hello team my meetings skype mee...	hmjdrvpb komuaywn	GRP_0
2	cant log in to vpn	received from hi i cannot log on to vpn best	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

TEXT PRE-PROCESSING

As first steps, we would be concatenating Short Description and Description columns (given that more than a fourth of records have exactly the same data in both of them)

```
new_df.insert(loc=4,
              column='Total',
              allow_duplicates=True,
              value=list(new_df['Short description'].str.strip() + ' ' + new_df['Descripti
ion'].str.strip()))
```

```
new_df.head()
```

	Short description	Description	Caller	Assignment group	Total
0	login issue	verified user details employee manager name ch...	spxjnwir pjlcoqds	GRP_0	login issue verified user details employee man...
1	outlook	received from hello team my meetings skype mee...	hmjdrvpb komuaywn	GRP_0	outlook received from hello team my meetings s...
2	cant log in to vpn	received from hi i cannot log on to vpn best	eylqgodm ybqkwiam	GRP_0	cant log in to vpn received from hi i cannot l...
3	unable to access hr tool page	unable to access hr tool page	xbkucsvz gcpydteq	GRP_0	unable to access hr tool page unable to access...
4	skype error	skype error	owlgqjme qhcozdfx	GRP_0	skype error skype error

Comments: As seen in case 4, there is duplication and this needs to be removed.

REMOVING DUPLICATES

```
new_df["Total"] = new_df["Total"].apply(lambda x: ' '.join(pd.unique(x.split())))
```

```
new_df.head()
```

	Short description	Description	Caller	Assignment group	Total
0	login issue	verified user details employee manager name ch...	spxjnwir pjlcoqds	GRP_0	login issue verified user details employee man...
1	outlook	received from hello team my meetings skype mee...	hmjdrvpb komuaywn	GRP_0	outlook received from hello team my meetings s...
2	cant log in to vpn	received from hi i cannot log on to vpn best	eylqgodm ybqkwiam	GRP_0	cant log in to vpn received from hi i cannot o...
3	unable to access hr tool page	unable to access hr tool page	xbkucsvz gcpydteq	GRP_0	unable to access hr tool page
4	skype error	skype error	owlgqjme qhcozdfx	GRP_0	skype error

LEMMATIZATION

We are using SPACY for this given that it also takes of POS and works well on cleaned data

INITIALIZE SPACY 'EN' MEDIUM MODEL

```
# Initialize spacy 'en' medium model, keeping only tagger component needed for lemmatization
nlp = spacy.load('en_core_web_sm', disable=['parser', 'ner'])

# Define a function to lemmatize the descriptions
def lemmatizer(sentence):
    # Parse the sentence using the loaded 'en' model object `nlp`
    doc = nlp(sentence)
    return " ".join([token.lemma_ for token in doc if token.lemma_ !='-PRON-'])
```

```
# Checking to see how the lemmatizer function has worked for a record
print('\033[1mOriginal text:\033[0m')
print(new_df['Total'][0])
print('_'*100)
print('\033[1mLemmatized text:\033[0m')
print(lemmatizer(new_df['Total'][0]))
```

```
Original text:
login issue verified user details employee manager name checked the in ad and reset password advised to check caller confirmed that he was able resolved
____________________________________________________________________________________________________
Lemmatized text:
login issue verify user detail employee manager name check the in ad and reset password advise to check caller confirm that be able resolve
```

APPLYING ON THE DATAFRAME

```
# Applying on the database
new_df['Total'] = new_df['Total'].apply(lemmatizer)

# Verify the data
new_df.head()
```

	Short description	Description	Caller	Assignment group	Total
0	login issue	verified user details employee manager name ch...	spxjnwir pjlcoqds	GRP_0	login issue verify user detail employee manage...
1	outlook	received from hello team my meetings skype mee...	hmjdrvpb komuaywn	GRP_0	outlook receive from hello team meeting skype ...
2	cant log in to vpn	received from hi i cannot log on to vpn best	eylqgodm ybqkwiam	GRP_0	can not log in to vpn receive from hi i can no...
3	unable to access hr tool page	unable to access hr tool page	xbkucsvz gcpydteq	GRP_0	unable to access hr tool page
4	skype error	skype error	owlgqjme qhcozdfx	GRP_0	skype error

PREPARING LIST OF STOP WORDS

```
!pip install nltk
import nltk as nltk
from nltk.corpus import stopwords
nltk.download('stopwords')
from nltk.tokenize import word_tokenize
```

```
allstp=np.array(stopwords.words('english'))
allstp.size
```

```
179
```

```
#Creating an additional of stopwords that  we see as irrelevant to the modelling inputs
new_words=np.array(['yes','hi', 'receive','hello','sir','madam', 'best','morning','evening
new_words.size
```

```
nputs
'evening','afternoon' 'regards','thanks','from','greeting', 'forward','reply','will','pleas
```

```
,'greeting', 'forward','reply','will','please','see','help','able'])
```

```
20
```

CONCATENATING NLTK LIST AND OUR LIST OF STOPWORDS

```
stopwords=np.concatenate([allstp,new_words])
stopwords.size
```

```
199
```

FIND INDEX OF 'NOT' IN THE STOPWORDS

```
index_not = np.where(stopwords == 'not')
index_not
```

```
(array([118], dtype=int64),)
```

```
final_list=np.delete(stopwords, index_not)
final_list.size
```

```
198
```

The cleaning process below removes the stopwords defined above as a string irrespective of whether it is part of another word. Example: it will remove "i" in input as "i" is a stopword. Hence, to prevent the same, we will append a space before and after every term to defined as a word

```
final_list1=[]
for i in final_list:
#     print("initial")
#     print(i)
    txt=i
    txt=" "+txt+" "
    i=txt
    final_list1.append(i)
#     print("final:")
#     print(i)
# final_list1
```

CLEANING USERNAMES AND STOP WORDS FROM THE DESCRIPTIONS

```
uniq=new_df['Caller'].unique()
print(uniq)
uniq.size
```

```
['spxjnwir pjlcoqds' 'hmjdrvpb komuaywn' 'eylqgodm ybqkwiam' ...
 'bjitvswa yrmugfnq' 'oybwdsgx oxyhwrfz' 'kqvbrspl jyzoklfx']
```

```
2950
```

Preparing final list of terms that need to be deleted. This includes usernames and stopwords

```
uniq1=np.concatenate([uniq,final_list1])
uniq1.size
```

```
3148
```

```
new_df['Clean Description']=new_df['Total'].copy()
```

```
s=" "
for key, value in new_df['Total'].iteritems():
    r=value

    if(pd.isnull(value)):
        s=''
    else:
#         print(key)
        for u in range(uniq1.size):
            if(r.find(uniq1[u])!=-1):

        #print(uniq[u])
#                 print('un found: ',uniq1[u])
                s = r.replace(uniq1[u],' ')
                r=s
#                 print('Key: ',key)
#                 print('Original string: ',r)
#                 print('Revised string: ',s)
            else:
                s=r
#     print(key)
#     print(r)
#     print(s)
    new_df.at[key,'Clean Description']= s
#     print(key)
```

```
new_df.head()
```

	Short description	Description	Caller	Assignment group	Total	Clean Description
0	login issue	verified user details employee manager name ch...	spxjnwir pjlcoqds	GRP_0	login issue verify user detail employee manage...	login issue verify user detail employee manage...
1	outlook	received from hello team my meetings skype mee...	hmjdrvpb komuaywn	GRP_0	outlook receive from hello team meeting skype ...	outlook team meeting skype etc not appear cale...
2	cant log in to vpn	received from hi i cannot log on to vpn best	eylqgodm ybqkwiam	GRP_0	can not log in to vpn receive from hi i can no...	can not log vpn not best
3	unable to access hr tool page	unable to access hr tool page	xbkucsvz gcpydteq	GRP_0	unable to access hr tool page	unable access hr tool page
4	skype error	skype error	owlgqjme qhcozdfx	GRP_0	skype error	skype error

Comments: Now the Column "Clean Description" has completely clean data where all the stop words, names of people removed.

EDA ON DESCRIPTION

Lets understand the total number of words in the corpus.

```
cumulative_words = {}
cumulative_column = []

for x in new_df["Clean Description"].values:
    cumulative_words.update(dict.fromkeys(set(x.lower().split())))
    cumulative_column.append(cumulative_words.keys())
```

```
new_df["Column B"] = cumulative_column
new_df["Column C"] = new_df["Column B"].apply(len)
```

```
new_df.head()
```

	Short description	Description	Caller	Assignment group	Total	Clean Description	Column B	Column C
0	login issue	verified user details employee manager name ch...	spxjnwir pjlcoqds	GRP_0	login issue verify user detail employee manage...	login issue verify user detail employee manage...	(detail, resolve, issue, employee, advise, man...	10327
1	outlook	received from hello team my meetings skype mee...	hmjdrvpb komuaywn	GRP_0	outlook receive from hello team meeting skype ...	outlook team meeting skype etc not appear cale...	(detail, resolve, issue, employee, advise, man...	10327
2	cant log in to vpn	received from hi i cannot log on to vpn best	eylqgodm ybqkwiam	GRP_0	can not log in to vpn receive from hi i can no...	can not log vpn not best	(detail, resolve, issue, employee, advise, man...	10327
3	unable to access hr tool page	unable to access hr tool page	xbkucsvz gcpydteq	GRP_0	unable to access hr tool page	unable access hr tool page	(detail, resolve, issue, employee, advise, man...	10327
4	skype error	skype error	owlgqjme qhcozdfx	GRP_0	skype error	skype error	(detail, resolve, issue, employee, advise, man...	10327

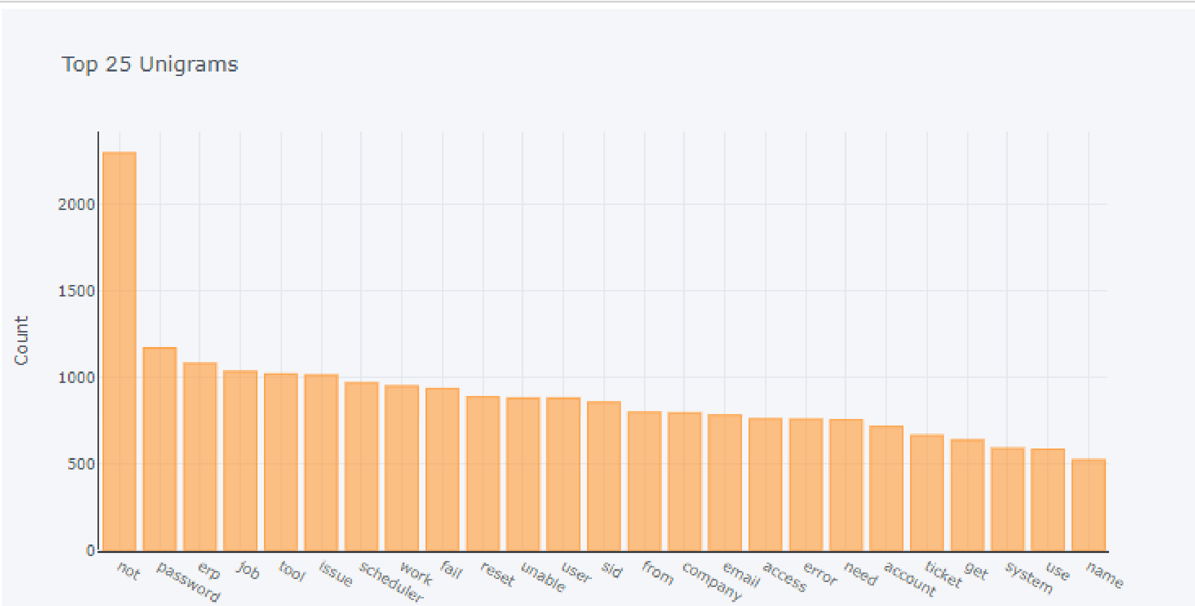
Comments: We have a total of 10327 unique words in our corpus

```
# Dropping the Column B & C
new_df=new_df.drop({'Column B','Column C'}, axis=1)
new_df.head()
```

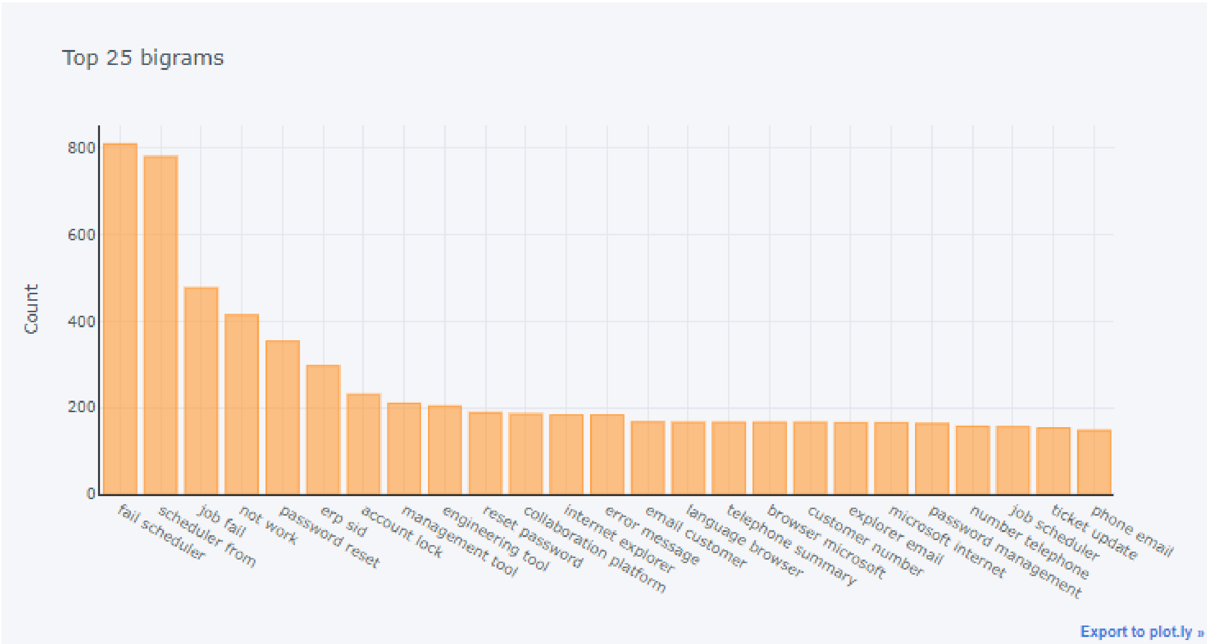
	Short description	Description	Caller	Assignment group	Total	Clean Description
0	login issue	verified user details employee manager name ch...	spxjnwir pjlcoqds	GRP_0	login issue verify user detail employee manage...	login issue verify user detail employee manage...
1	outlook	received from hello team my meetings skype mee...	hmjdrvpb komuaywn	GRP_0	outlook receive from hello team meeting skype ...	outlook team meeting skype etc not appear cale...
2	cant log in to vpn	received from hi i cannot log on to vpn best	eylqgodm ybqkwiam	GRP_0	can not log in to vpn receive from hi i can no...	can not log vpn not best
3	unable to access hr tool page	unable to access hr tool page	xbkucsvz gcpydteq	GRP_0	unable to access hr tool page	unable access hr tool page
4	skype error	skype error	owlgqjme qhcozdfx	GRP_0	skype error	skype error

UNDERSTANDING N-GRAMS

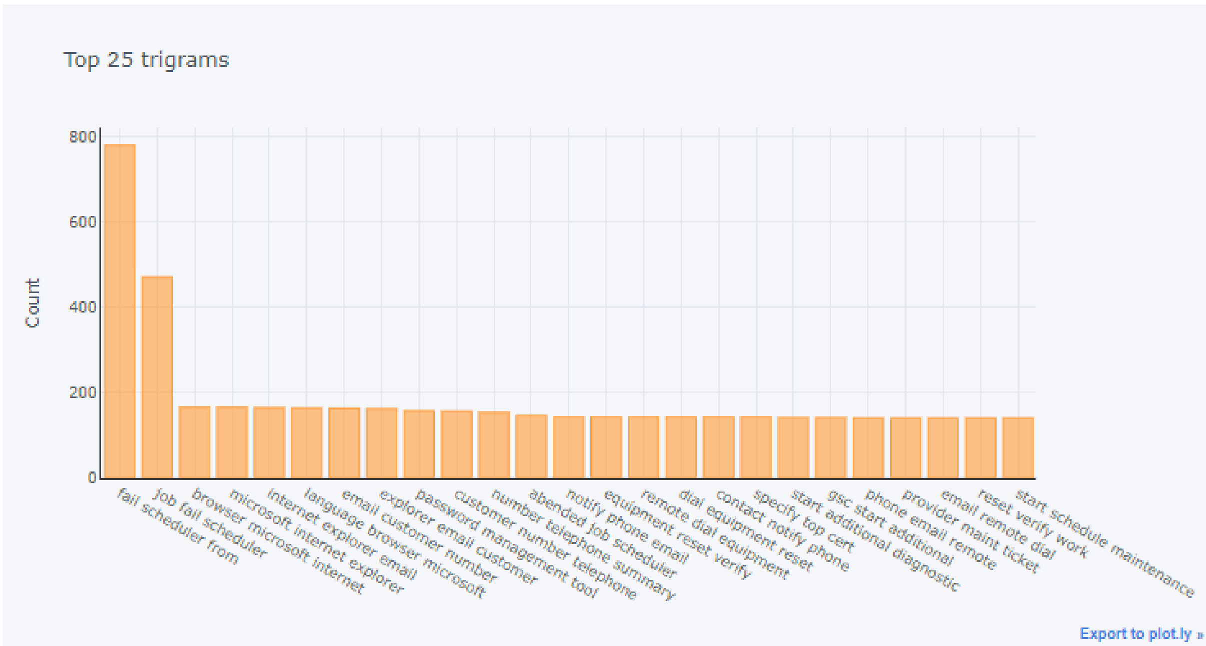
```
def get_top_n_words(corpus, n=None):
    vec = CountVectorizer().fit(corpus)
    bag_of_words = vec.transform(corpus)
    sum_words = bag_of_words.sum(axis=0)
    words_freq = [(word, sum_words[0, idx]) for word, idx in vec.vocabulary_.items()]
    words_freq =sorted(words_freq, key = lambda x: x[1], reverse=True)
    return words_freq[:n]
common_words = get_top_n_words(new_df['Clean Description'], 25)
df1 = pd.DataFrame(common_words, columns = ['Clean Description' , 'count'])
df1.groupby('Clean Description').sum()['count'].sort_values(ascending=False).iplot(
    kind='bar', yTitle='Count', linecolor='black',title='Top 25 Unigrams')
```



```
def get_top_n_bigram(corpus, n=None):
    vec = CountVectorizer(ngram_range=(2, 2)).fit(corpus)
    bag_of_words = vec.transform(corpus)
    sum_words = bag_of_words.sum(axis=0)
    words_freq = [(word, sum_words[0, idx]) for word, idx in vec.vocabulary_.items()]
    words_freq =sorted(words_freq, key = lambda x: x[1], reverse=True)
    return words_freq[:n]
common_words = get_top_n_bigram(new_df['Clean Description'], 25)
df2 = pd.DataFrame(common_words, columns = ['Clean Description' , 'count'])
df2.groupby('Clean Description').sum()['count'].sort_values(ascending=False).iplot(
    kind='bar', yTitle='Count', linecolor='black', title='Top 25 bigrams')
```



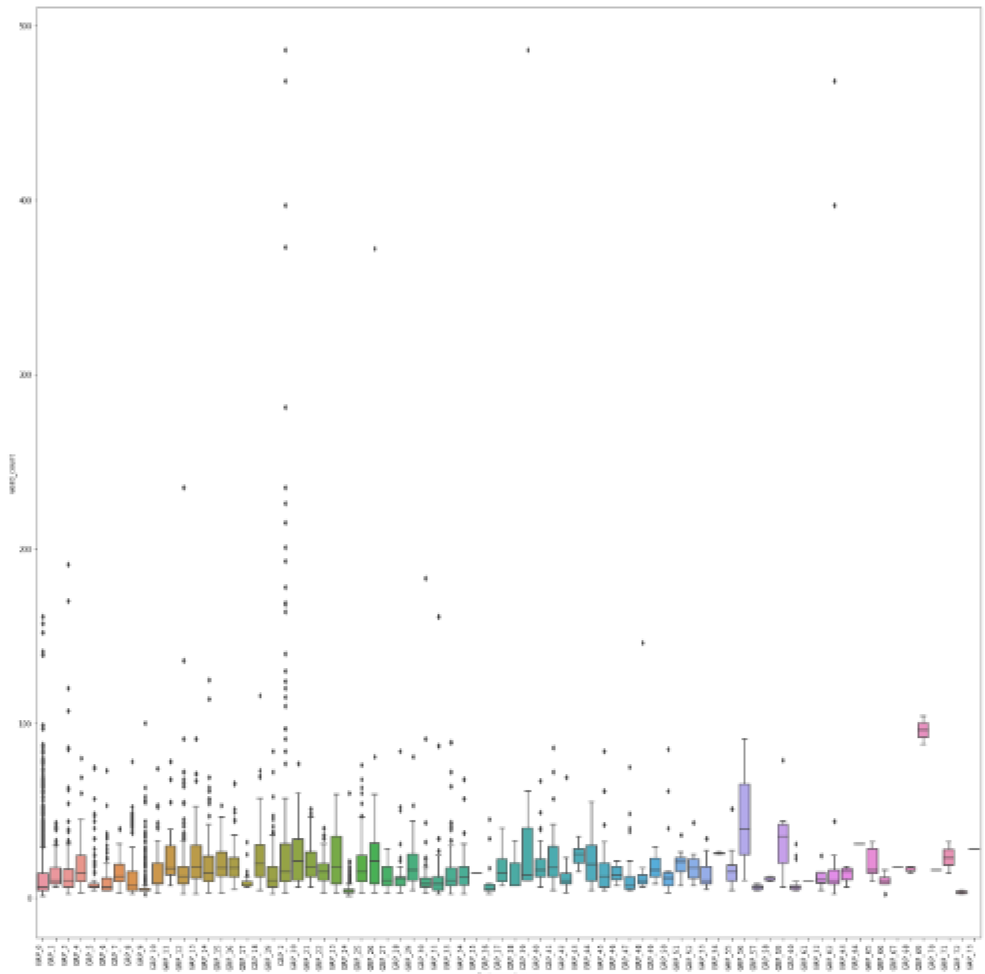
```python
def get_top_n_trigram(corpus, n=None):
    vec = CountVectorizer(ngram_range=(3, 3)).fit(corpus)
    bag_of_words = vec.transform(corpus)
    sum_words = bag_of_words.sum(axis=0)
    words_freq = [(word, sum_words[0, idx]) for word, idx in vec.vocabulary_.items()]
    words_freq =sorted(words_freq, key = lambda x: x[1], reverse=True)
    return words_freq[:n]
common_words = get_top_n_trigram(new_df['Clean Description'], 25)
df3 = pd.DataFrame(common_words, columns = ['Clean Description' , 'count'])
df3.groupby('Clean Description').sum()['count'].sort_values(ascending=False).iplot(
    kind='bar', yTitle='Count', linecolor='black', title='Top 25 trigrams')
```



UNDERSTANDING WHETHER THE LENGTH OF THE TICKET HAS AN IMPLICATION

```python
#creating a column for determining word count
new_df['word_count'] = new_df['Clean Description'].str.split().map(len)
```

```python
fig_dims = (20, 20)
fig, ax = plt.subplots(figsize=fig_dims)
chart=sns.boxplot(x="Assignment group", y="word_count",ax=ax, data=new_df)
chart.set_xticklabels(chart.get_xticklabels(), rotation=90, ha="right")
plt.tight_layout()
plt.show()
```



Excepting for some groups like group 56 & 70 that stand out in terms of the large no. of words used, difficult to see a pattern emerging across other groups.

THEMATIC ANALYSIS OF DATA

```
df_theme=new_df.copy()
```

```
df_theme.head(5)
```

	Short description	Description	Caller	Assignment group	Total	Clean Description	New grouping
0	login issue	verified user details employee manager name ch...	spxjnwir pjlcoqds	GRP_0	login issue verify user detail employee manage...	login issue verify user detail employee manage...	GRP_0
1	outlook	received from hello team my meetings skype mee...	hmjdrvpb komuaywn	GRP_0	outlook receive from hello team meeting skype ...	outlook team meeting skype etc not appear cale...	GRP_0
2	cant log in to vpn	received from hi i cannot log on to vpn best	eylqgodm ybqkwiam	GRP_0	can not log in to vpn receive from hi i can no...	can not log vpn not best	GRP_0
3	unable to access hr tool page	unable to access hr tool page	xbkucsvz gcpydteq	GRP_0	unable to access hr tool page	unable access hr tool page	GRP_0
4	skype error	skype error	owlgqjme qhcozdfx	GRP_0	skype error	skype error	GRP_0

```
corpus_text = '\n'.join(df_theme[:]['Clean Description'])
sentences = corpus_text.split('\n')
sentences = [line.lower().split(' ') for line in sentences]
```

```
sentences[5]
```

```
['unable', 'log', 'engineering', 'tool', 'skype']
```

REMOVING ADDITIONAL PUNCTUATION

```
def clean(s): #removing additional punctuation
    return [w.strip(',."!?:;()\'') for w in s]
sentences = [clean(s) for s in sentences if len(s) > 0]
```

WORD2VEC MODEL

```
from gensim.models import Word2Vec

model = Word2Vec(sentences, size=8000, window=3, min_count=3, workers=4)
model.save("word2vec.mdl")
```

```
vectors = model.wv #keeping only the vector values
len(model.wv.vocab)
```

```
3135
```

TAKING SAMPLE VALUE AND ANALYSING THE RELATIONSHIPS

```
print(vectors.similarity('access', 'login'))
print(vectors.similarity('access', 'skype'))
```

```
0.91378766
0.9864943
```

```
vectors.most_similar('vpn')
```

```
[('connect', 0.9780881404876709),
 ('collaboration', 0.9752592444419861),
 ('platform', 0.9695403575897217),
 ('tologin', 0.9675502777099609),
 ('khrtyujuine', 0.9626268744468689),
 ('can', 0.9617358446121216),
 ('sync', 0.9580341577529907),
 ('access', 0.9579378366470337),
 ('skype', 0.9573770761489868),
 ('engineering', 0.9566475749015808)]
```

So we can see strong connects with **into**, **sign**, **access**,**log** - words which are typically used for a connectivity session which is what a VPN is used for. This implies that our vectors seem to be defined well.

```
term = ['reset','account']
for i in term:
    print('\n')
    print('For term: \033[1m',i,'\033[0m the most similar words are:')
    for word, similarity in model.most_similar(positive=i, topn=10): #
          print (word, round(similarity, 4))
```

```
For term:  reset  the most similar words are:
password 0.9699
user 0.9552
login 0.9514
passwords 0.948
management 0.9443
windows 0.943
pwd 0.9323
unlocked 0.9321
use 0.9316
manager 0.9308
```

```
For term:  account  the most similar words are:
unlock 0.9979
lock 0.9954
erp 0.9919
ad 0.973
unlocked 0.9721
windows 0.9713
system 0.9599
haunm 0.9597
pasword 0.9554
pwd 0.9544
```

However as expected, **pwd**, **'password'**, **'passwords'**, **'verify'** in the top 10 words similar to **'reset'**.Ironically, **'manager'** is also part of the list. This can be explained by the fact, that of the 476 tickets that contain **'reset'**, 55 of them also have the word **'manager'**(11.5%).

Similarly, for **account**, the highest ranking words are **unlock** and **lock** apart from **erp** and **windows** which are systems that are account-controlled

```
ordered_vocab = [(term, voc.index, voc.count) for term, voc in model.wv.vocab.items()]
ordered_vocab = sorted(ordered_vocab, key=lambda k: -k[2])
ordered_terms, term_indices, term_counts = zip(*ordered_vocab)
# print(ordered_terms)
# create a DataFrame with the vectors as data,
# and the terms as row labels
word_vectors = pd.DataFrame(model.wv.syn0norm[term_indices, :], index=ordered_terms)
```

```
word_vectors
```

	0	1	2	3	4	5	6	7	8	9
	0.010920	-0.015687	-0.001783	-0.000403	0.006413	-0.006828	-0.010189	0.023540	-0.005165	-0.015204
not	0.015736	-0.010790	0.000101	0.006203	-0.003666	-0.018243	-0.017409	0.028857	0.004644	-0.022141
password	0.013606	-0.009779	-0.006441	0.005316	-0.002068	-0.020070	-0.012629	0.023219	-0.003234	-0.020001
erp	0.015552	-0.007558	-0.003524	0.007158	-0.008652	-0.017091	-0.011709	0.025378	-0.001648	-0.015725
job	0.019201	-0.003413	0.000356	0.009801	-0.010739	-0.001276	-0.007831	0.024045	0.006138	0.003692
tool	0.019768	-0.011661	0.002929	0.009314	-0.007545	-0.015586	-0.012077	0.029238	0.002615	-0.017398
issue	0.016079	-0.010681	0.000359	0.006820	-0.004187	-0.014994	-0.015921	0.028925	0.003370	-0.019710
scheduler	0.019583	-0.003784	0.000058	0.009761	-0.010587	-0.001732	-0.008273	0.024455	0.006284	0.003347
work	0.016963	-0.007242	-0.015408	0.007706	0.000024	-0.025655	-0.020185	0.024464	0.002677	-0.015774
fail	0.019030	-0.003291	0.000140	0.009962	-0.010637	-0.001280	-0.007891	0.023955	0.006582	0.003911
reset	0.012858	-0.005737	-0.013771	0.005416	-0.001640	-0.022467	-0.015790	0.020776	-0.005854	-0.016528
user	0.014612	-0.009280	-0.003662	0.006151	-0.003871	-0.018539	-0.015259	0.025640	-0.001105	-0.020234
unable	0.017235	-0.012525	0.001253	0.007570	-0.004258	-0.017138	-0.014600	0.029913	0.004873	-0.020944
sid	0.016562	-0.003736	-0.000896	0.009214	-0.014835	-0.010360	-0.007375	0.022581	0.000307	-0.004615
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
santiago	0.019197	-0.011821	-0.003618	0.008542	-0.002831	-0.010334	-0.014560	0.030893	0.004142	-0.011938
zcnc	0.021315	-0.011212	-0.003729	0.011237	-0.003355	-0.011140	-0.013811	0.031100	0.008102	-0.010370
olibercsu	0.020056	-0.010146	-0.003551	0.008861	-0.002740	-0.010911	-0.015338	0.030659	0.005239	-0.011762
sbinuxja	0.018617	-0.010489	-0.003020	0.007504	-0.002865	-0.012097	-0.015324	0.030324	0.003189	-0.013925
vtbegcho	0.019356	-0.009696	-0.003856	0.008255	-0.002907	-0.011828	-0.015654	0.030054	0.003993	-0.012428
nicolmghyu	0.020394	-0.009613	-0.003257	0.009643	-0.004586	-0.011011	-0.014837	0.030326	0.005613	-0.010562
docx	0.020341	-0.010973	-0.003460	0.009793	-0.003680	-0.011399	-0.014287	0.030950	0.006024	-0.011491
division	0.018238	-0.009857	-0.003570	0.007439	-0.002073	-0.011107	-0.015777	0.029891	0.003776	-0.013287
rak	0.019813	-0.011919	-0.003733	0.008184	-0.002510	-0.011405	-0.014717	0.030768	0.005325	-0.011301
gncpezhx	0.018472	-0.009457	-0.002999	0.007840	-0.002352	-0.009661	-0.015836	0.029661	0.004024	-0.012543
hopqcvza	0.018908	-0.009674	-0.002358	0.008209	-0.003101	-0.009782	-0.015146	0.029755	0.004839	-0.011519
wilsfgtjl	0.019713	-0.011611	-0.002973	0.008286	-0.002339	-0.011975	-0.015134	0.031199	0.004395	-0.013583

3135 rows × 8000 columns

```
from sklearn.manifold import TSNE
tsne_input = word_vectors
```

```
tsne = TSNE()
tsne_vectors = tsne.fit_transform(tsne_input.values)
```

```
tsne_vectors
```

```
array([[-77.30749  ,  -3.0828352],
       [-73.649376 , -17.936193 ],
       [-78.73836  , -23.170765 ],
       ...,
       [-41.374584 ,  26.841705 ],
       [ -0.6887137,  13.049257 ],
       [-61.09315  ,  -1.040639 ]], dtype=float32)
```

```
tsne_vectors1 = pd.DataFrame(tsne_vectors,
                             index=pd.Index(tsne_input.index),
                             columns=['x_coord', 'y_coord'])
tsne_vectors1['word'] = tsne_vectors1.index
```

```
from bokeh.plotting import figure, show, output_notebook
from bokeh.models import HoverTool, ColumnDataSource, value

output_notebook()
```

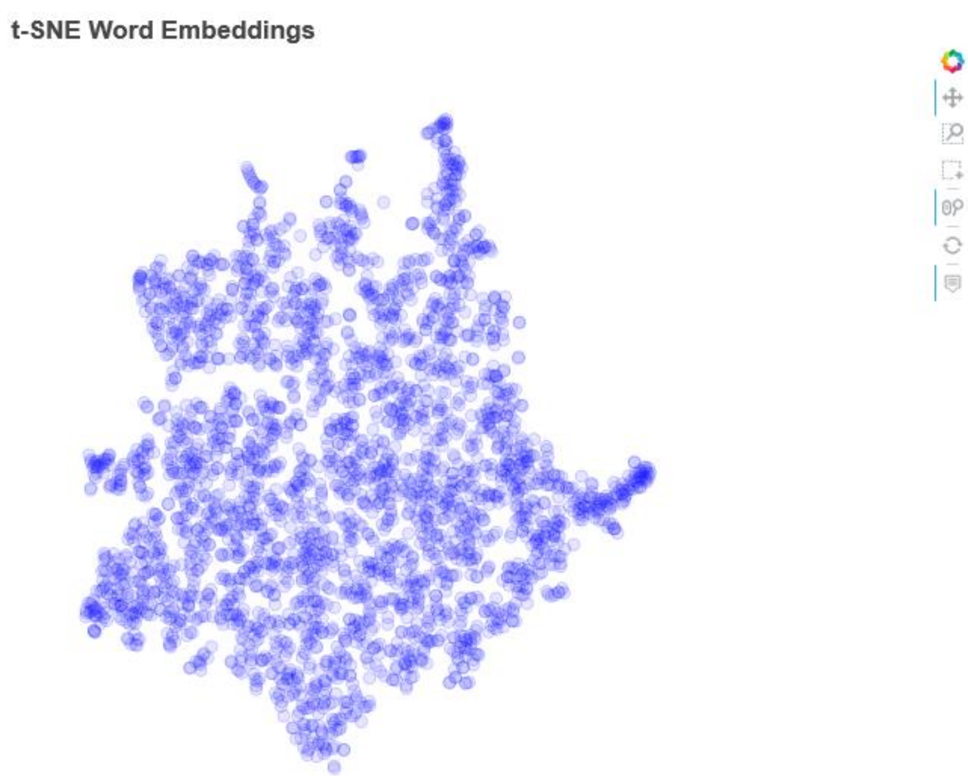
BokehJS 1.0.4 successfully loaded.

```python
# add our DataFrame as a ColumnDataSource for Bokeh
plot_data = ColumnDataSource(tsne_vectors1)

# create the plot and configure the
# title, dimensions, and tools
tsne_plot = figure(title='t-SNE Word Embeddings',
                   plot_width = 800,
                   plot_height = 800,
                   tools= ('pan, wheel_zoom, box_zoom,'
                           'box_select, reset'),
                   active_scroll='wheel_zoom')
tsne_plot.add_tools( HoverTool(tooltips = '@word') )
# draw the words as circles on the plot
tsne_plot.circle('x_coord', 'y_coord', source=plot_data,
                 color='blue', line_alpha=0.2, fill_alpha=0.1,
                 size=10, hover_line_color='black')

# configure visual elements of the plot
tsne_plot.title.text_font_size = value(u'16pt')
tsne_plot.xaxis.visible = False
tsne_plot.yaxis.visible = False
tsne_plot.grid.grid_line_color = None
tsne_plot.outline_line_color = None

# engage!
show(tsne_plot);
```



COMBINING ASSIGNMENT GROUPS

We had earlier mentioned that a lof assignment groups had very less records and thus we would need to combine all such groups into a "Other" Group.

```
new_df['New grouping']=new_df['Assignment group'].copy()
```

ALL GROUPS WITH LESS THAN 30 RECORDS WILL BE COMBINED INTO OTHERS

```
cols = ['New grouping']
for col in cols:
    val = new_df[col].value_counts()
    y = val[val < 30].index # all groups with less than 30 records will be combined
    new_df[col] = new_df[col].replace({x:'other' for x in y})
```

```
new_df.head()
```

	Short description	Description	Caller	Assignment group	Total	Clean Description	word_count	New grouping
0	login issue	verified user details employee manager name ch...	spxjnwir pjlcoqds	GRP_0	login issue verify user detail employee manage...	login issue verify user detail employee manage...	17	GRP_0
1	outlook	received from hello team my meetings skype mee...	hmjdrvpb komuaywn	GRP_0	outlook receive from hello team meeting skype ...	outlook team meeting skype etc not appear cale...	12	GRP_0
2	cant log in to vpn	received from hi i cannot log on to vpn best	eylqgodm ybqkwiam	GRP_0	can not log in to vpn receive from hi i can no...	can not log vpn not best	6	GRP_0
3	unable to access hr tool page	unable to access hr tool page	xbkucsvz gcpydteq	GRP_0	unable to access hr tool page	unable access hr tool page	5	GRP_0
4	skype error	skype error	owlgqjme qhcozdfx	GRP_0	skype error	skype error	2	GRP_0

```
new_df['New grouping'].value_counts()
```

```
GRP_0     3976
GRP_8      661
other      357
GRP_24     289
GRP_12     257
GRP_9      252
GRP_2      241
GRP_19     215
GRP_3      200
GRP_6      184
GRP_13     145
GRP_10     140
GRP_5      129
GRP_14     118
GRP_25     116
GRP_33     107
GRP_4      100
GRP_29      97
GRP_18      88
GRP_16      85
GRP_17      81
GRP_31      69
GRP_7       68
GRP_34      62
GRP_26      56
GRP_40      45
GRP_28      44
GRP_41      40
GRP_30      39
GRP_15      39
GRP_42      37
GRP_20      36
GRP_45      35
GRP_1       31
GRP_22      31
GRP_11      30
Name: New grouping, dtype: int64
```

Thus Now from the earlier 74 groups, we now have only 36 groups - with a group called as "others". While the groups are quite unbalanced in terms of no. of records in each group, at this stage, we would not be undertaking any balancing exercise but check the accuracies of the models and of required at the next stage try and balance the groups.

MILESTONE 2: MODEL BUILDING

● Building a model architecture which can classify.
● Trying different model architectures by researching state of the art for similar tasks.
● Train the model
● To deal with large training time, save the weights so that you can use them when training the model for the second time without starting from scratch.

ANALYSE TO SET THE PARAMETER VALUES FOR MODEL

This is meant to be a trial run of an initial model to understand likely accuracies. At this stage, we have two columns of interest, "Clean Description" and "New grouping". A new dataframe will be created using only these two columns and a trial model would be run

```
model_df = new_df[{'Clean Description','New grouping'}]
```

```
model_df.head()
```

	New grouping	Clean Description
0	GRP_0	login issue verify user detail employee manage...
1	GRP_0	outlook team meeting skype etc not appear cale...
2	GRP_0	can not log vpn not best
3	GRP_0	unable access hr tool page
4	GRP_0	skype error

```
model_df.shape
```

```
(8500, 2)
```

```
model_df.describe().transpose()
```

	count	unique	top	freq
New grouping	8500	36	GRP_0	3976
Clean Description	8500	6634	job fail scheduler from	445

GET THE LENGTH OF EACH LINE AND FIND THE MAXIMUM LENGTH

As different lines are of different length. We need to pad our sequences using the max length.

```
#creating a column for determining word count
model_df['word_count'] = model_df['Clean Description'].str.split().map(len)
```

```
model_df.head()
```

	New grouping	Clean Description	word_count
0	GRP_0	login issue verify user detail employee manage...	17
1	GRP_0	outlook team meeting skype etc not appear cale...	12
2	GRP_0	can not log vpn not best	6
3	GRP_0	unable access hr tool page	5
4	GRP_0	skype error	2

```
# understanding the max no. of words
max (model_df['word_count'])
```

```
486
```

Thus the maximum no. of words is 486

```
np.mean (model_df['word_count'])
```

```
13.77070588235294
```

```
np.std (model_df['word_count'])
```

```
20.593414440671673
```

Given that the average no. of words is ~ 14 and standard deviation is ~21, we could safely fix the max number of words at 100 without too much loss in data

SET PARAMETERS FOR THE MODEL

```
max_features = 10000
maxlen = 100
embedding_size = 200
```

APPLY KERAS TOKENIZER OF HEADLINE COLUMN OF YOUR DATA

First creating a tokenizer instance using Tokenizer(num_words=max_features) And then fit this tokenizer instance on your data column model_df['Clean Description'] using .fit_on_texts()

```
tokenizer = Tokenizer(num_words=max_features,filters= '!"#$%&()*+,-./:;<=>?@[\]^_`{|}\n“~”')
```

```
tokenizer.fit_on_texts(model_df['Clean Description'])
```

DEFINING X AND Y FOR THE MODEL

```
# First to Create a target categorical column
model_df['New grouping'] = model_df['New grouping'].astype('category').cat.codes
model_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8500 entries, 0 to 8499
Data columns (total 3 columns):
New grouping         8500 non-null int8
Clean Description    8500 non-null object
word_count           8500 non-null int64
dtypes: int64(1), int8(1), object(1)
memory usage: 141.2+ KB
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
```

```
model_df.head()
```

	New grouping	Clean Description	word_count
0	0	login issue verify user detail employee manage...	17
1	0	outlook team meeting skype etc not appear cale...	12
2	0	can not log vpn not best	6
3	0	unable access hr tool page	5
4	0	skype error	2

```
X = tokenizer.texts_to_sequences(model_df['Clean Description'])
X = pad_sequences(X, maxlen = maxlen)
y = np.asarray(model_df['New grouping'])

print("Number of Samples:", len(X))
print(X[0])
print("Number of Labels: ", len(y))
print(y[0])
```

```
Number of Samples: 8500
[   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0   30    6  102   11   89  129   59
   25   32  207   10    2  161   32  257  142   78]
Number of Labels:  8500
0
```

GET THE VOCABULARY SIZE

```
vocab_size = len(tokenizer.word_index) + 1
print('Vocabulary size: %d\nDocuments count: %d' % (vocab_size, tokenizer.document_count))
```

```
Vocabulary size: 10328
Documents count: 8500
```

Note:

That + 1 is because of reserving padding (i.e. index zero).

GET GLOVE WORD EMBEDDINGS

```
glove_file = "glove.6B.zip"
```

```
#Extract Glove embedding zip file
from zipfile import ZipFile
with ZipFile(glove_file, 'r') as z:
  z.extractall()
```

GET THE WORD EMBEDDINGS USING EMBEDDING FILE

Get The Word Embeddings Using Embedding File And Creating A Weight Matrix For Words In Training Docs

```
# load the whole embedding into memory
embeddings_index = dict()
f = open('glove.6B.200d.txt', encoding="utf8")
for line in f:
    values = line.split()
    word = values[0]
    coefs = np.asarray(values[1:], dtype='float32')
    embeddings_index[word] = coefs
f.close()
print('Loaded %s word vectors.' % len(embeddings_index))

# create a weight matrix for words in training docs
embedding_matrix = np.zeros((vocab_size, 200))


for word, i in tokenizer.word_index.items():
    embedding_vector = embeddings_index.get(word)
    if embedding_vector is not None:
        embedding_matrix[i] = embedding_vector
```

```
Loaded 400000 word vectors.
```

SPLITTING THE DATA INTO TRAINING AND VALIDATION SAMPLES

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=1, stratify=y)
```

CREATE AND COMPILE YOUR MODEL

Using a Sequential model instance and then add Embedding layer, Bidirectional(LSTM) layer, then dense and dropout layers as required. In the end add a final dense layer with sigmoid activation for binary classification.

```
# Build the model
embedding_dim = 200

model = Sequential()
model.add(Embedding(vocab_size,
                    embedding_dim,
                    embeddings_initializer=Constant(embedding_matrix),
                    input_length=maxlen,
                    trainable=True))
model.add(SpatialDropout1D(0.2))
model.add(Bidirectional(CuDNNLSTM(128, return_sequences=True)))
model.add(Bidirectional(CuDNNLSTM(64)))
model.add(Dropout(0.25))
#model.add(Dense(units=2, activation='sigmoid')) # found softmax to work better
model.add(Dense(units=36, activation='softmax'))
model.compile(loss = 'categorical_crossentropy', optimizer='adam',metrics = ['accuracy'])
print(model.summary())
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
embedding_1 (Embedding)      (None, 100, 200)          2065600   
_________________________________________________________________
spatial_dropout1d_1 (Spatial (None, 100, 200)          0         
_________________________________________________________________
bidirectional_1 (Bidirection (None, 100, 256)          337920    
_________________________________________________________________
bidirectional_2 (Bidirection (None, 128)               164864    
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0         
_________________________________________________________________
dense_1 (Dense)              (None, 36)                4644      
=================================================================
Total params: 2,573,028
Trainable params: 2,573,028
Non-trainable params: 0
_________________________________________________________________
None
```

FITTING THE MODEL

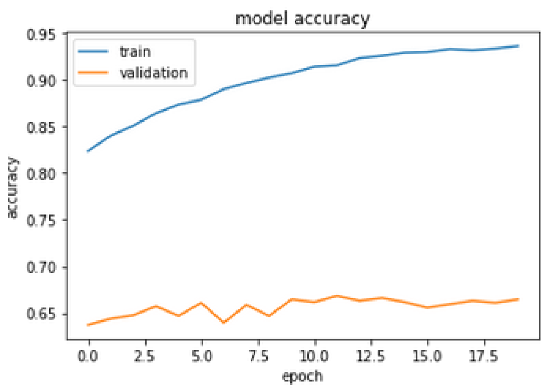
Fitting the model with a batch size of 100 and validation_split = 0.2

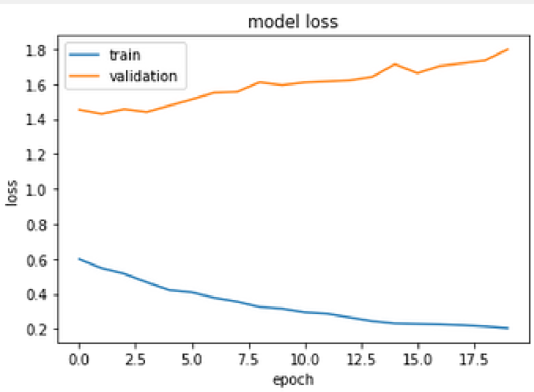
```
# Converting to categorical data
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
```

```
batch_size = 100
epochs = 20
history = model.fit(X_train, y_train, epochs=epochs, batch_size=batch_size, verbose=1, validation_split=0.2)
```

```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()
```





Model clearly overfits, on validation data it has an accuracy of only ~ 67% whereas train accuracy is 94%.

PREDICTION ON TEST DATA

```python
# Generic method to print the classification report
def classification_summary(y_test, y_pred, y_proba):
    print('\033[1mModel accuracy:\033[0m %.2f%%' % (accuracy_score(y_test, y_pred) * 100))
    print('_'*80)
    print('\033[1mConfusion matrix:\033[0m\n %s' % (confusion_matrix(y_test.argmax(axis=1), y_pred.argmax(axi
    print('_'*80)
    print('\033[1mClassification report:\033[0m\n %s' % (classification_report(y_test, y_pred)))
    print('_'*80)
```

```python
ssification_summary(y_test, y_pred, y_proba):
nt('\033[1mModel accuracy:\033[0m %.2f%%' % (accuracy_score(y_test, y_pred) * 100))
nt('_'*80)
nt('\033[1mConfusion matrix:\033[0m\n %s' % (confusion_matrix(y_test.argmax(axis=1), y_pred.argmax(axis=1))))
nt('_'*80)
nt('\033[1mClassification report:\033[0m\n %s' % (classification_report(y_test, y_pred)))
nt('_'*80)
```

```python
# Analyze Classification Summary
y_proba = model.predict([X_test])
y_pred = (y_proba > 0.5).astype('int32')
classification_summary(y_test, y_pred, y_proba)
```

```
Model accuracy: 61.24%
________________________________________________________________________________
Confusion matrix:
 [[721   0   1 ...   0   1  12]
 [  0   0   0 ...   0   0   1]
 [  4   0  19 ...   0   0   3]
 ...
 [ 45   0   1 ...  72   0   1]
 [ 48   0   0 ...   1   0   2]
 [ 32   1   1 ...   3   1  13]]
________________________________________________________________________________
Classification report:
               precision    recall  f1-score   support

           0       0.83      0.88      0.85       795
           1       0.00      0.00      0.00         6
           2       0.73      0.68      0.70        28
           3       0.00      0.00      0.00         6
           4       0.69      0.53      0.60        51
           5       0.42      0.34      0.38        29
           6       0.54      0.29      0.38        24
           7       0.67      0.25      0.36         8
           8       0.46      0.35      0.40        17
           9       0.82      0.88      0.85        16
          10       0.58      0.39      0.47        18
          11       0.39      0.21      0.27        43
          12       0.57      0.35      0.44        48
```

3	0.00	0.00	0.00	6
4	0.69	0.53	0.60	51
5	0.42	0.34	0.38	29
6	0.54	0.29	0.38	24
7	0.67	0.25	0.36	8
8	0.46	0.35	0.40	17
9	0.82	0.88	0.85	16
10	0.58	0.39	0.47	18
11	0.39	0.21	0.27	43
12	0.57	0.35	0.44	48
13	0.00	0.00	0.00	7
14	1.00	0.17	0.29	6
15	0.95	0.66	0.78	58
16	0.33	0.30	0.32	23
17	0.25	0.18	0.21	11
18	0.40	0.22	0.29	9
19	0.69	0.45	0.55	20
20	0.30	0.40	0.34	40
21	1.00	0.25	0.40	8
22	0.50	0.21	0.30	14
23	0.50	0.14	0.22	21
24	0.33	0.25	0.29	12
25	0.30	0.30	0.30	20
26	0.43	0.33	0.38	9
27	0.62	0.62	0.62	8
28	0.33	0.14	0.20	7
29	0.33	0.14	0.20	7
30	0.52	0.46	0.49	26
31	0.84	0.57	0.68	37
32	0.57	0.29	0.38	14
33	0.81	0.55	0.65	132
34	0.00	0.00	0.00	51
35	0.25	0.18	0.21	71
micro avg	0.71	0.61	0.66	1700
macro avg	0.50	0.33	0.38	1700
weighted avg	0.67	0.61	0.63	1700
samples avg	0.61	0.61	0.61	1700

For the smaller groups, the accuracy is really low; we can potentially think of merging groups with a record base of less than 50 (instead of 30).

INSTEAD OF GLOVE USING WORD2VEC

Perhaps more appropriate in this situation since quite a few of the words like http, vpn, etc might not be present in the Glove Vocabulary and thus creating a vocabulary for this specific study might be better

```
!pip install gensim
```

MODELING WITH WORD2VEC

```
# Load the Word2Vec model
wmodel = Doc2Vec.load('word2vec.mdl')
```

```
w2v_weights = wmodel.wv.vectors
vocab_size, embedding_size = w2v_weights.shape
print("Vocabulary Size: {} - Embedding Dim: {}".format(vocab_size, embedding_size))
```

```
Vocabulary Size: 3135 - Embedding Dim: 8000
```

```
# CREATE the MODEL

# Samples of categories with less than this number of samples will be ignored
DROP_THRESHOLD = 10000
model_wv = Sequential()
model_wv.add(Embedding(input_dim=vocab_size,
                       output_dim=embedding_size,
                       weights=[w2v_weights],
                       input_length=maxlen,
                       mask_zero=True,
                       trainable=False))
model_wv.add(SpatialDropout1D(0.2))
model_wv.add(Bidirectional(LSTM(128, return_sequences=True)))
model_wv.add(Bidirectional(LSTM(64)))
model_wv.add(Dropout(0.25))
#model.add(Dense(units=2, activation='sigmoid')) # found softmax to work better
model_wv.add(Dense(units=36, activation='softmax'))
model_wv.compile(loss = 'categorical_crossentropy', optimizer='adam',metrics = ['accuracy'])
print(model_wv.summary())
```

```
WARNING:tensorflow:From C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\keras\backend.py:3794:
add_dispatch_support.<locals>.wrapper (from tensorflow.python.ops.array_ops) is deprecated and will be removed in a future
version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_4 (Embedding)      (None, 100, 8000)         25080000
_________________________________________________________________
spatial_dropout1d_3 (Spatial (None, 100, 8000)         0
_________________________________________________________________
bidirectional_4 (Bidirection (None, 100, 256)          8324096
_________________________________________________________________
bidirectional_5 (Bidirection (None, 128)               164352
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 36)                4644
=================================================================
Total params: 33,573,092
Trainable params: 8,493,092
Non-trainable params: 25,080,000
_________________________________________________________________
None
```

```
batch_size = 100
epochs = 20
history = model_wv.fit(X_train, y_train, epochs=epochs, batch_size=batch_size, verbose=1, validation_spli
```

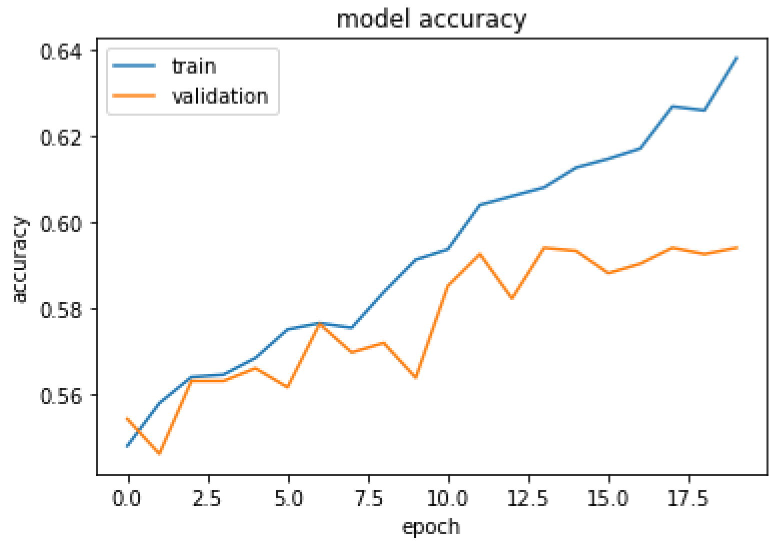
```
batch_size=batch_size, verbose=1, validation_split=0.2)
```

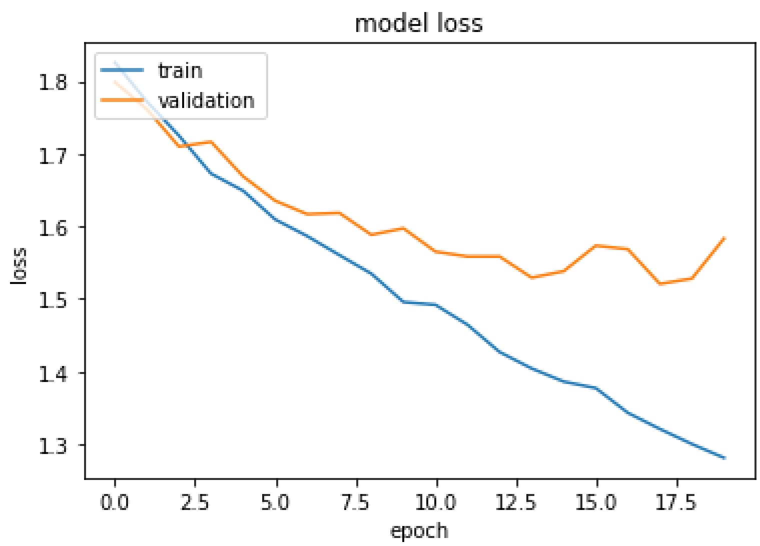
```
WARNING:tensorflow:From C:\ProgramData\Anaconda3\lib\site-packages\keras\backend\tensorflow_backend.py:422: The name tf.global_variables is deprecated. Please use
tf.compat.v1.global_variables instead.

Train on 5440 samples, validate on 1360 samples
Epoch 1/20
5440/5440 [==============================] - 7s 1ms/step - loss: 2.4032 - accuracy: 0.4906 - val_loss: 2.0145 - val_accuracy: 0.5390
Epoch 2/20
5440/5440 [==============================] - 4s 746us/step - loss: 1.8557 - accuracy: 0.5511 - val_loss: 1.7470 - val_accuracy: 0.5676
Epoch 3/20
5440/5440 [==============================] - 5s 942us/step - loss: 1.6060 - accuracy: 0.5908 - val_loss: 1.6015 - val_accuracy: 0.5875
Epoch 4/20
5440/5440 [==============================] - 9s 2ms/step - loss: 1.4235 - accuracy: 0.6241 - val_loss: 1.4852 - val_accuracy: 0.6029
Epoch 5/20
5440/5440 [==============================] - 14s 3ms/step - loss: 1.2848 - accuracy: 0.6548 - val_loss: 1.4204 - val_accuracy: 0.6118
Epoch 6/20
5440/5440 [==============================] - 20s 4ms/step - loss: 1.1681 - accuracy: 0.6722 - val_loss: 1.4323 - val_accuracy: 0.6272
Epoch 7/20
5440/5440 [==============================] - 21s 4ms/step - loss: 1.0614 - accuracy: 0.6982 - val_loss: 1.3695 - val_accuracy: 0.6331
Epoch 8/20
5440/5440 [==============================] - 21s 4ms/step - loss: 0.9648 - accuracy: 0.7250 - val_loss: 1.3758 - val_accuracy: 0.6412
Epoch 9/20
5440/5440 [==============================] - 20s 4ms/step - loss: 0.8719 - accuracy: 0.7502 - val_loss: 1.3969 - val_accuracy: 0.6441
Epoch 10/20
5440/5440 [==============================] - 20s 4ms/step - loss: 0.7849 - accuracy: 0.7785 - val_loss: 1.3331 - val_accuracy: 0.6434
Epoch 11/20
5440/5440 [==============================] - 20s 4ms/step - loss: 0.7080 - accuracy: 0.7980 - val_loss: 1.3705 - val_accuracy: 0.6493
Epoch 12/20
5440/5440 [==============================] - 20s 4ms/step - loss: 0.6224 - accuracy: 0.8182 - val_loss: 1.3947 - val_accuracy: 0.6529
Epoch 13/20
5440/5440 [==============================] - 20s 4ms/step - loss: 0.5783 - accuracy: 0.8357 - val_loss: 1.3726 - val_accuracy: 0.6529
Epoch 14/20
5440/5440 [==============================] - 20s 4ms/step - loss: 0.5144 - accuracy: 0.8487 - val_loss: 1.4673 - val_accuracy: 0.6515
Epoch 15/20
5440/5440 [==============================] - 20s 4ms/step - loss: 0.4667 - accuracy: 0.8647 - val_loss: 1.4241 - val_accuracy: 0.6537
Epoch 16/20
5440/5440 [==============================] - 20s 4ms/step - loss: 0.4210 - accuracy: 0.8768 - val_loss: 1.4460 - val_accuracy: 0.6485
Epoch 17/20
5440/5440 [==============================] - 20s 4ms/step - loss: 0.4107 - accuracy: 0.8756 - val_loss: 1.4442 - val_accuracy: 0.6551
Epoch 18/20
5440/5440 [==============================] - 20s 4ms/step - loss: 0.3650 - accuracy: 0.8939 - val_loss: 1.4409 - val_accuracy: 0.6625
Epoch 19/20
5440/5440 [==============================] - 20s 4ms/step - loss: 0.3302 - accuracy: 0.9064 - val_loss: 1.5367 - val_accuracy: 0.6559
Epoch 20/20
5440/5440 [==============================] - 20s 4ms/step - loss: 0.3193 - accuracy: 0.9022 - val_loss: 1.5055 - val_accuracy: 0.6596
```

```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()
```





Model clearly overfits, on validation data it has an accuracy of only ~ 66% whereas train accuracy is 90%.

ANALYZE CLASSIFICATION SUMMARY

```
# Analyze Classification Summary
y_proba = model_wv.predict([X_test])
y_pred = (y_proba > 0.5).astype('int32')
classification_summary(y_test, y_pred, y_proba)
```

```
Model accuracy: 48.24%

Confusion matrix:
 [[782    0    0 ...    0    0    0]
 [  3    0    0 ...    3    0    0]
 [ 19    0    7 ...    2    0    0]
 ...
 [ 49    0    1 ...   76    0    0]
 [ 50    0    0 ...    0    1    0]
 [ 67    1    0 ...    3    0    0]]

Classification report:
              precision    recall  f1-score   support

           0       0.74      0.85      0.79       795
           1       0.00      0.00      0.00         6
           2       0.88      0.25      0.39        28
           3       0.00      0.00      0.00         6
           4       0.75      0.12      0.20        51
           5       0.00      0.00      0.00        29
           6       0.50      0.08      0.14        24
           7       0.00      0.00      0.00         8
           8       0.00      0.00      0.00        17
           9       0.92      0.75      0.83        16
          10       0.00      0.00      0.00        18
          11       0.00      0.00      0.00        43
          12       0.80      0.08      0.15        48
          13       0.00      0.00      0.00         7
          14       0.00      0.00      0.00         6
          15       0.75      0.57      0.65        58
          16       1.00      0.04      0.08        23
          17       0.00      0.00      0.00        11
```

18	0.00	0.00	0.00	9
19	0.00	0.00	0.00	20
20	0.00	0.00	0.00	40
21	0.00	0.00	0.00	8
22	0.00	0.00	0.00	14
23	0.50	0.05	0.09	21
24	0.00	0.00	0.00	12
25	0.12	0.05	0.07	20
26	0.00	0.00	0.00	9
27	0.00	0.00	0.00	8
28	0.00	0.00	0.00	7
29	0.00	0.00	0.00	7
30	0.00	0.00	0.00	26
31	0.00	0.00	0.00	37
32	0.00	0.00	0.00	14
33	0.64	0.58	0.61	132
34	1.00	0.02	0.04	51
35	0.00	0.00	0.00	71
micro avg	0.73	0.48	0.58	1700
macro avg	0.24	0.10	0.11	1700
weighted avg	0.55	0.48	0.47	1700
samples avg	0.48	0.48	0.48	1700

SUMMARY OF MODEL FITS

The same bi-directional LSTM model (with all parameters the same) has been applied to both the Glove as well as the WOrd2Vec Embedding.

The Glove based LSTM gives a higher validation accuracy of ~ 66% and the Word2Vec based one which has a validation accuracy of 59%