



Final Report Capstone Project

Automatic Ticket Assignment

JUL 2020

Submitted by:

Shailesh Shankar Kawade
Rajesh Chaudhary
Srinivas Mahakud
Vivek Inamdar
Sowbhagya Lakshmi Behal

Mentored by:

Sumit Kumar

Table of Contents

1. Summary of Problem Statement, Data and Findings	3
Problem Statement	3
Objective	3
Dataset	4
Summary of the Approach to EDA	4
Check missing values in the dataset	5
Analysing each column of the Data	6
Visualization	9
Observations of Given Dataset	12
2. Overview of the Final Process	13
Process Overview	13
Best Practices / Strategies Leveraged	13
Internal libraries created for modularity purpose	14
Inference and Corrective Actions	14
Algorithms used	15
3. Step-by-Step Walk Through the Solution	16
Data Pre-processing	16
Feature Engineering & Model Building	19
Topic and Word Cloud Representation	23
Plot the pie chart showing the percentage of data in different topics after LDA:	23
Analysis using WordCloud	24
Word Distribution	26
Modeling	26
Representing the raw text data as numerical data by doing vectorization	27
Traditional ML Models	27
Multinomial Naive Bayes	27
K Nearest Neighbor	28
Support Vector Machine	28
Decision Tree	29
RandomForest Classifier	29
AdaBoostClassifier	30
BaggingClassifier	30
Bi-LSTM Model	31
Observations from Bi-LSTM Model	31

Confusion Matrix	32
Inference from Confusion Matrix	33
Tuning of LSTM Model	33
1. LSTM Merge Mode	33
2. Number of LSTM Cells	34
3. Regularization	35
4. Pipeline	36
Observations:	36
4. Model Evaluation	37
Other Models Tried	38
1. BERT	38
5. Comparison to Benchmark	38
6. Implications	39
7. Limitations	39
8. Closing Reflections	40
9. Appendix	41
Github Link	41
References	41
1. Topic Modeling with Gensim (Python)	41
2. Topic Modeling Industrial-Strength Natural Language Processing	41
3. Translate List and Pandas data frame using googletrans library in python	41



1. Summary of Problem Statement, Data and Findings

Problem Statement

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

The Service Desk team (L1/L2) will perform basic analysis on the user's requirement, identify the issue based on given descriptions and assign it to the respective teams.

The manual assignment of these incidents might have below disadvantages:

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

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

Objective

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

to assign the tickets to right functional groups by analysing the given description with an accuracy of at least 85 -90%.

Dataset

Details of the dataset is in the below link:

<https://drive.google.com/file/d/1OZNJm81JXucV3HmZroMq6qCT2m7ez7IJ/edit>

The dataset consists of incident tickets information which are assigned to specific groups.

This dataset has 8500 rows with 4 columns.

- Short description
- Description
- Caller
- Assignment group

The target column 'Assignment Group' has 74 values.

Summary of the Approach to EDA

The description of dataset is as below:

	Short description	Description	Caller	Assignment group
count	8492	8499	8500	8500
unique	7481	7817	2950	74
top	password reset	the bpctwhsn kzqsbmtp		GRP_0
freq	38	56	810	3976

The dataset comprises 8500 rows and 4 columns. All columns are of type object containing textual information. Password reset is one of the most occurring tickets which reflects in the Short description column. The top occurring Description in the dataset is 'the', it is meaningless and we have to deal with it. Also we can see the top caller is 'bpctwhsn kzqsbmtp' and top or most frequent assignment group is GRP_0

Check missing values in the dataset

```
#Checking if the data set has any NULL OR NAN Values
tickets_corpus.isna().sum()
```

```
Short description    8
Description          1
Caller              0
Assignment group    0
dtype: int64
```

We have very few NaN in the dataset in Short Description and Description column.

```
#displaying the data where 'Description' is null

tickets_corpus[tickets_corpus['Description'].isna()]
```

	Short description	Description	Caller	Assignment group
4395	i am locked out of skype	NaN	viyglzfo ajtfzpkb	GRP_0

```
#displaying the data where 'Short description' is null

tickets_corpus[tickets_corpus['Short description'].isna()]
```

	Short description	Description	Caller	Assignment group
2604	NaN	\r\n\r\nreceived from: ohdrnswl.rezuibdt@gmail...	ohdrnswl rezuibdt	GRP_34
3383	NaN	\r\n-connected to the user system using teamvi...	qftpazns fxpnytmk	GRP_0
3906	NaN	-user unable tologin to vpn.\r\n-connected to...	awpcmsey ctduqwe	GRP_0
3910	NaN	-user unable tologin to vpn.\r\n-connected to...	rhwsmefo tvphyura	GRP_0
3915	NaN	-user unable tologin to vpn.\r\n-connected to...	hxripljo efzounig	GRP_0
3921	NaN	-user unable tologin to vpn.\r\n-connected to...	czyadygo veiosxby	GRP_0
3924	NaN	name:vvqgbdhm fwchqjor\nlanguage:\nbrowser:mic...	vvqgbdhm fwchqjor	GRP_0
4341	NaN	\r\n\r\nreceived from: eqmuniov.ehxkcbgj@gmail...	eqmuniov ehxkcbgj	GRP_0

As per the above analysis, where we have missing values in the 'Description' column, the corresponding 'Short description' value is present. Also where the 'Short description' column has NaN values, the corresponding 'Description' column values are present.

Further processing, we are going to merge these two columns, so we no need to delete these NaN rows.

Analysing each column of the Data

'Assignment group' column

- There are a total of 74 groups , and GRP_0 has the highest number of tickets, 3968 out of 8500.
- We have around 6 assignment groups which have only one ticket sample.

'Short description' Column

- Maximum length of single record in Short Description 159
- Minimum length of single record in Short Description 1
- Average length of single record in Short Description 47.26628194558945
- Maximum Word count of single record of Short Description 28
- Minimum Word count of single record of Short Description 1
- Average Word count of single record of Short Description 6.93393004357555

'Description' Column

- Maximum length of single record of Description 13001
- Minimum length of single record of Description 1
- Average length of single record of Description 204.08173360028266
- Maximum Word count of single record of Description 1417
- Minimum Word count of single record of Description 1
- Average Word count of single record of Description 28.88635025320928

'caller' Column

Number of Unique callers in the Dataset 2948.

Lets see the top 10 callers in raising tickets

```
top_callers = tickets_corpus.groupby(['Caller']).size().nlargest(10)
print(top_callers)
```

```
Caller
bpctwhsnkzqsbtmtp    810
ZkBogxibQsEJzdZO     151
fumkcsjisarmtlhy     134
rbozivdqgmlhrtvp      87
rkupnshbgsmzfojw      71
jloygrwhacvztedi      64
spxqmiryzpwgoqju      63
oldrctiubxurpsyi      57
olckhmvxpcqobjnd      54
dkmcfreganwmfvlg      51
dtype: int64
```

Only one caller has raised 810 tickets, rest of the callers are raised only < 200 tickets

Let's check if any caller raised the tickets for multiple groups

```
top_c = tickets_corpus['Caller'].groupby(tickets_corpus['Assignment group']).value_counts()
grp_caller = pd.DataFrame(top_c.groupby(level=0).nlargest(5).reset_index(level=0, drop=True))
multy_caller = grp_caller[grp_caller.Caller.duplicated()]
grp_caller.head(20)
```


Assignment group		Caller	
GRP_0	fumkcsjisarmtlhy		132
	rbozivdqgmlhrtvp		86
	olckhmvxpcqobjnd		54
	efbwiadpdicafxhv		45
	mfeyoulindobtzpw		13
GRP_1	bpctwhsnkzqsbmp		6
	jloygrwhacvztedi		4
	jyoqwxhzclhxsoqy		3
	spxqmiryzpwgoqu		3
	kbnfxpsygehxyzayq		2
GRP_10	bpctwhsnkzqsbmp		60
	ihfk wzjderbxoyqk		6
	dizquolfhlykecx		5
	gnasmtvxcwxtsvkm		3
	hlmufzxqcdzierm		3
GRP_11	ctvaejbomjcerqwo		7
	tghrloksjbgcvlmf		2
	vlymsnejwhlqxcst		2
	dnwfhpylzbldipk		1
	fbgetcnjlsvxura		1

```

multy_caller_unique = [idx[1] for idx in multy_caller.index[multy_caller.Caller.unique()]]
multy_caller_unique

```

```

['hlrmufzxqcdzierm',
'fbgetcznjlsvxura',
'gnasmtvxcwxtsvkm',
'ihfkwxjderbxoyqk',
'tqfnalpjyoscnge',
'fmqubnvskcxpeyiv',
'tghrlksjbgcvlmf',
'jwqyxbzsadpvilqu',
'nuhfwpljojcwxser',
'oldrctiubxurpsyi',
'vlymsnejwhlqxcst',
'dkmcdfreganwmfvlg',
'bpctwhsnkzqsbmtp',
'spxqmiryzpwoqju',
'obanjrhgrnafleys']

```

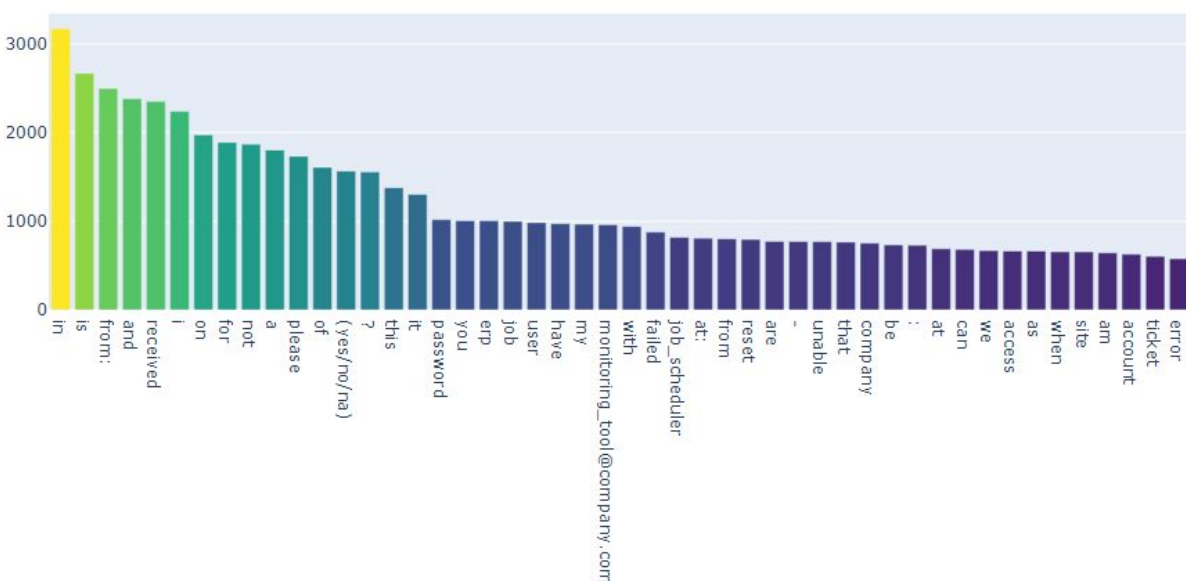
The above callers have raised tickets for multiple groups.

As per our above analysis, we don't see any significant relationship between the 'Caller' and group to which tickets are assigned. So for the model building, we may can avoid this column

Visualization

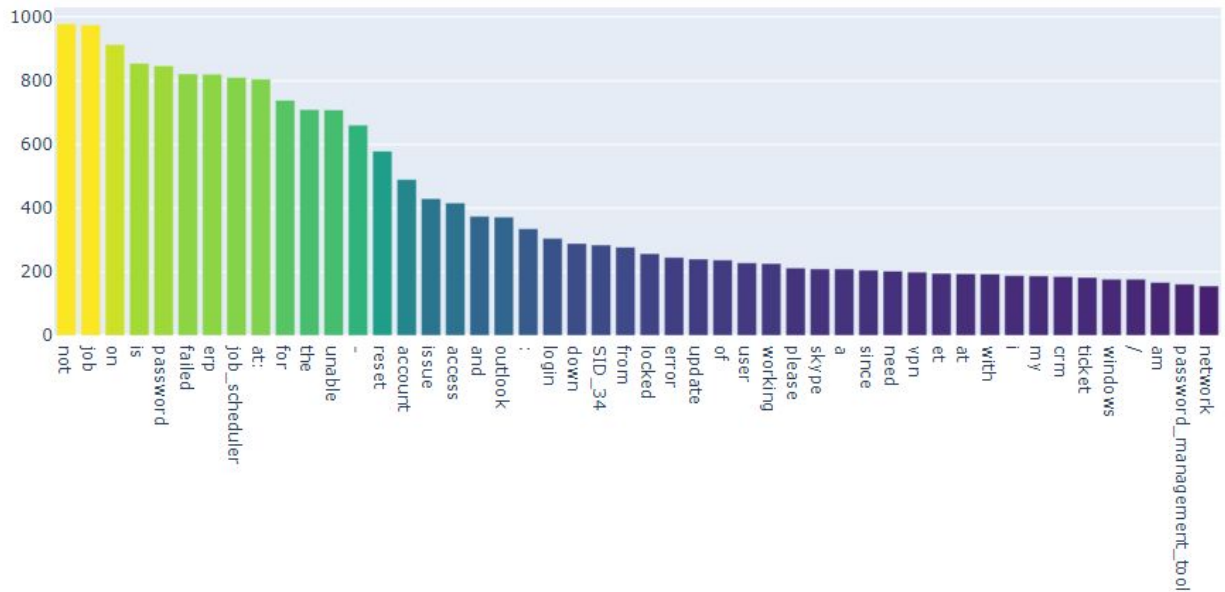
Visualizing the Frequency of words in Description:

Frequent Occuring word (unclean) in Description

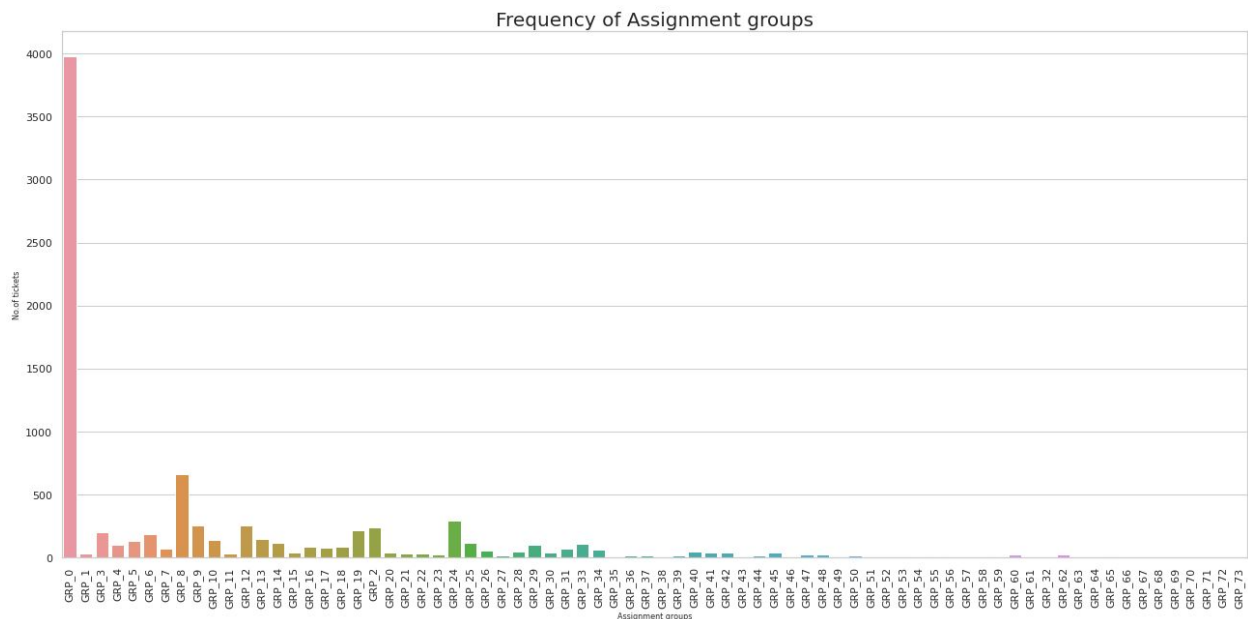


Visualizing the Frequency of words in Short description

Frequent Occuring word (unclean) in Short Description



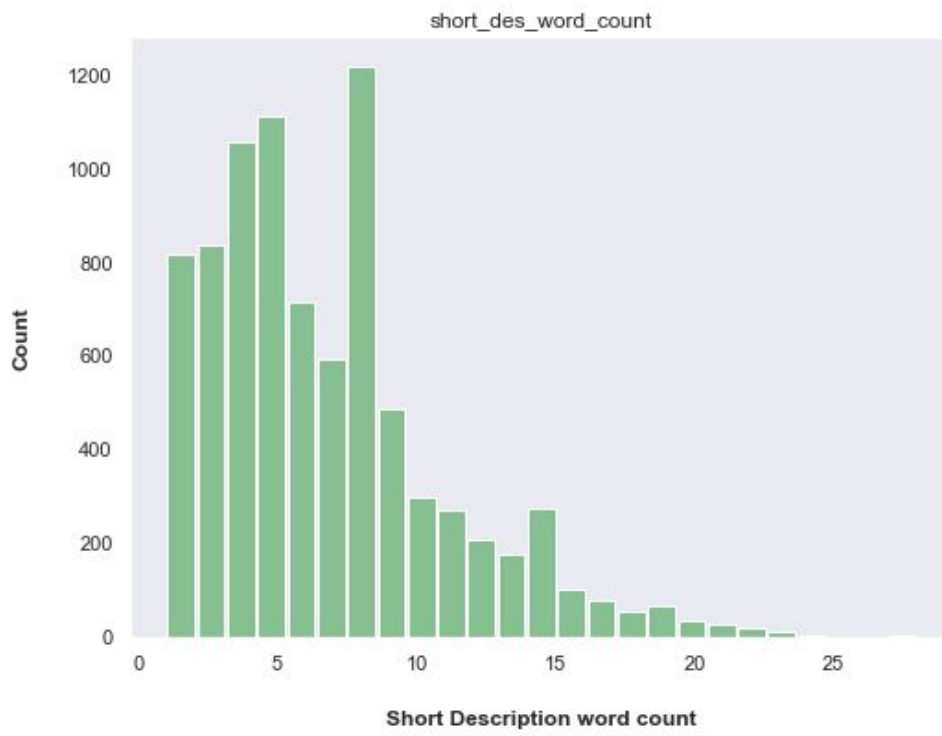
Distribution of tickets by the Group



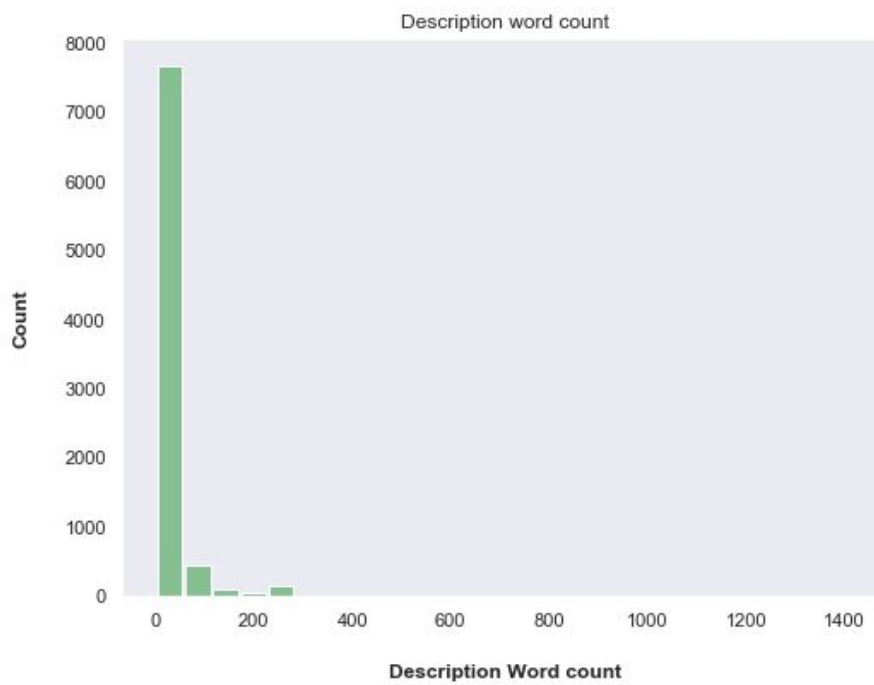
From the above plot we can see that 46.73% of the data is for GRP_0.

Some important insights from the above plot- group 0, 8, 24, 12, 9, 2, 19 have the highest number of cases tagged. Data is highly biased towards GRP_0 incidents.

Short Description Word count visualization



Short Description Word count visualization



Observations of Given Dataset

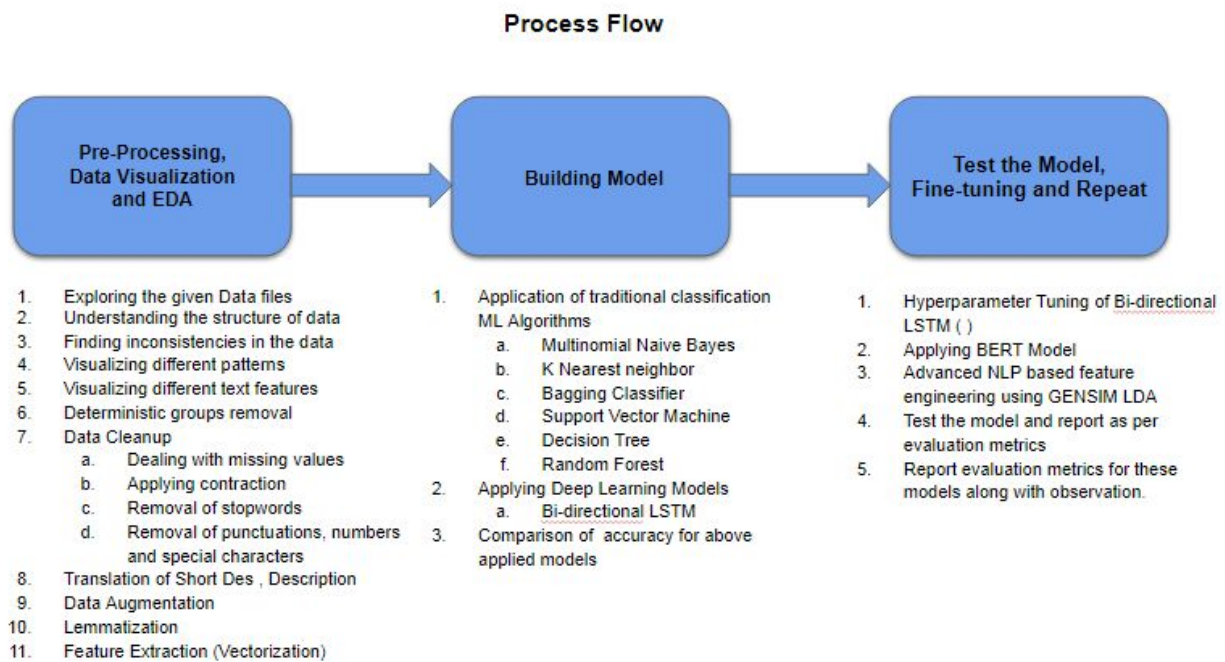
Following observations are seen in given dataset:

- ❑ Four columns – Short Description, Description, Caller and Assignment group
- ❑ 74 Assignment groups found - these will be our Target classes
- ❑ Caller names in a random fashion (may not be useful for training data)
- ❑ European non-English language also found in the data
- ❑ Email/chat format in description
- ❑ Images are attached in Description
- ❑ Symbols & other characters in the description
- ❑ Hyperlinks, URLs & few image data found in the description
- ❑ Blanks found either in the short description or description field
- ❑ Few descriptions same as the short description
- ❑ Few words were combined together
- ❑ Spelling mistakes and typo errors are found
- ❑ Missing data in case of Description & Short Description

2. Overview of the Final Process

Process Overview

The following submission guidelines & requirements apply to this Request for Proposal:



Best Practices / Strategies Leveraged

Following best practices are being used:

1. Modularization using different function files (.py) e.g. data_rule.py, data_augmentation.py.
2. Leveraging the PKL for persisting intermittent dataframes to save time across multiple iterations.
3. Avoiding smaller Assignment groups (to avoid overfitting issues) with strategies like deterministic groups.
4. Data Augmentation to take care of imbalanced dataset distribution.

Internal libraries created for modularity purpose

Following libraries (.npy) are created to reuse in main notebook:

- `import data_trans as tr` # Used for translation
- `import data_augmentation as da` # used for data augmentation
- `import data_cleaning as dc` # used for data cleaning
- `import data_rules as dr` # used for data rules

Inference and Corrective Actions

- ❑ There are Total 74 groups but there is serious **class imbalance** i.e. “GRP_0” has **almost 50% of all records**. This demands the need for data augmentation. Also there are a lot of groups which have very few tickets.
- ❑ After the manual data analysis, it seems that there can be certain tickets and groups which have certain patterns, e.g. certain combinations of short description , description and caller is always assigned to a particular group. To handle these types of ticket assignments we do not need machine learning. There can be deterministic rules which can take care of these issues. **Also having deterministic rules for groups with smaller numbers of records helps us in avoiding oversampling of the smaller groups. These smaller numbers of groups and records will not be part of Machine learning flow. Deterministic Rules assignments will run before machine learning model predictions.**
- ❑ Also there are 8 missing Short Descriptions and 1 missing Description. where we have missing values in the 'Description' column, the corresponding 'Short description' value is present. Also where the 'Short description' column has NaN values, the corresponding 'Description' column values are present. Further processing, we are going to merge these two columns, so we no need to delete these NaN rows.
- ❑ Manual Analysis of data shows that the description needs to be cleaned up for removal of stopwords (with enhanced list of stopwords which are relevant to this dataset), punctuations, special characters, numbers etc.

- ❑ There are certain tickets which are in Non-English languages. They need to be detected and translated into english before carrying out any further pre-processing.

Algorithms used

We have tried below pre-modelling and modelling techniques

1. Multinomial Naive Bayes
2. K Nearest neighbor
3. Bagging Classifier
4. Support Vector Machine (SVM)
5. Decision Tree
6. Random Forest
7. Adaboost Classifier
8. Bagging Classifier
9. Bidirectional LSTM Models using both embedding and compared
10. BERT



3. Step-by-Step Walk Through the Solution

Data Pre-processing

Summary of Pre-Processing Steps (Before feature engineering)

- ❑ Identify Deterministic Rules
- ❑ Data Cleaning up Activities.
- ❑ Data Translation for non-english tickets.
- ❑ Data Augmentation to balance the dataset for all groups

Considering the modularization and reusability of above functions separate function files (.py) have been created and imported into this main file.

Remove Deterministic Groups From Dataset

- ➔ For groups which have a lesser number of ticket assignments (in 10's max). If there are standard patterns found in ticket assignment to groups based on content of short description and description then such groups are removed from the Machine Learning process.
- ➔ When a new ticket arrives it will be first evaluated against the deterministic rules with the help of patterns matching against short description and description. If there are no deterministic groups found then ML model prediction will be run. (Evaluation method is provided in the data_rules.py file.

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

- ❑ Dropped the caller field as the data was not found to be useful for analysis
- ❑ Replaced Null values in Short description and Description by merging Short Description & Description fields for analysis
- ❑ Remove Deterministic Groups From DataSet

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

Data Cleaning

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

Translation

It has been observed that a given dataset has non-english data as well. Here we are translating the 'Short Description' and 'Description' column by using `googletrans` library to translate non-english data to English language.

Lemmatization & Stop words removal

Stop words have been removed using `nlTK` corpus modules.

Lemmatization is the process of grouping together the different inflected forms of a word so they can be analyzed as a single item. Lemmatization is similar to Stemming but it brings context to the words. So, it links words with similar meanings to one word.

Lemmatization is the process of converting a word to its base form, e.g., going to “go”. We use spaCy’s lemmatizer to obtain the lemma, or base form, of the words.

Here we have preferred Lemmatization over Stemming because lemmatization does morphological analysis of the words.

Spell Check

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

Found that performing spell check was a time consuming process. Although spell check helped in reduction of vocabulary size, it did not help in model accuracy improvement. Hence it’s not been applied in the data preprocessing.

Data Augmentation

There are Total 74 groups but there is serious class imbalance i.e. “GRP_0” has almost 50% of all records. This demands the need for data augmentation. For creating new records for the groups which have smaller numbers of records. It uses the Spacy library and Synonym based record generation technique.

After invoking the function for Removal of Deterministic groups from data_rules.py file, 54 groups are remaining.

Group_0 has divided into 5 subgroups and augmented upto 8000 each and later these 5 subgroups again have been merged back to Group_0. Other remaining 53 groups have been done for other 53 groups. In this way data is balanced and ready to do feature engineering.

Feature Engineering & Model Building

Overview of the steps:

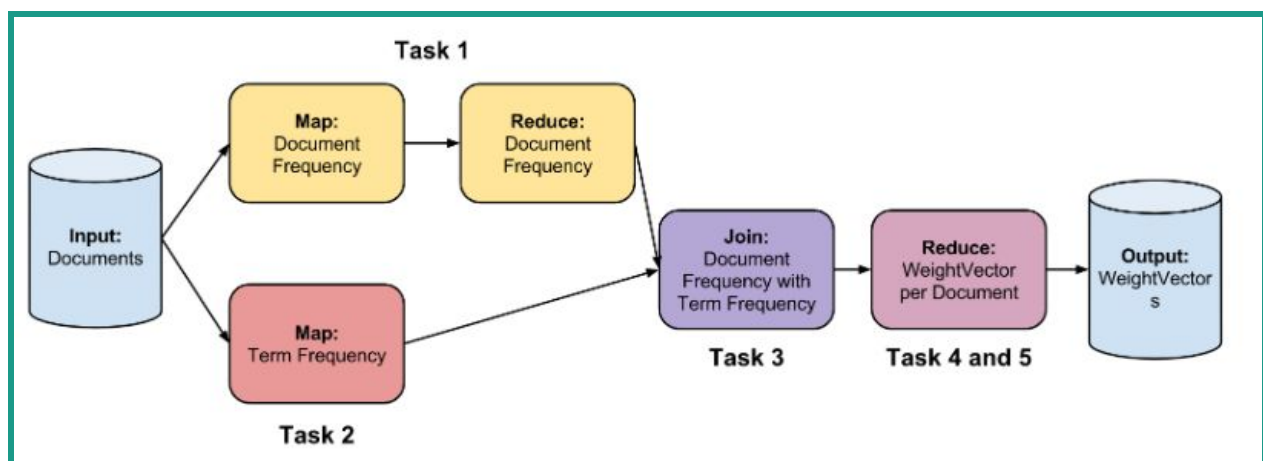
- ❑ Generating Features using TF IDF over lemmatized data.
- ❑ Traditional machine learning algorithms meant for classification problem solving will be tried against the vectorized features generated out of TF IDF.
- ❑ Deep learning models such as Bidirectional LSTM will be leveraged.
- ❑ Comparison of the model accuracy for selecting best performing model.

Two different types of feature engineering approaches have been tried:

- TF-IDF Approach
- LDA based Feature Engineering

TF-IDF Approach

Term frequency-Inverse document frequency uses all the tokens in the dataset as vocabulary. Frequency of occurrence of a token from vocabulary in each document consists of the term frequency and number of documents in which token occurs determines the Inverse document frequency. What this ensures is that, if a token occurs frequently in a document that token will have high TF but if that token occurs frequently in majority of documents then it reduces the IDF

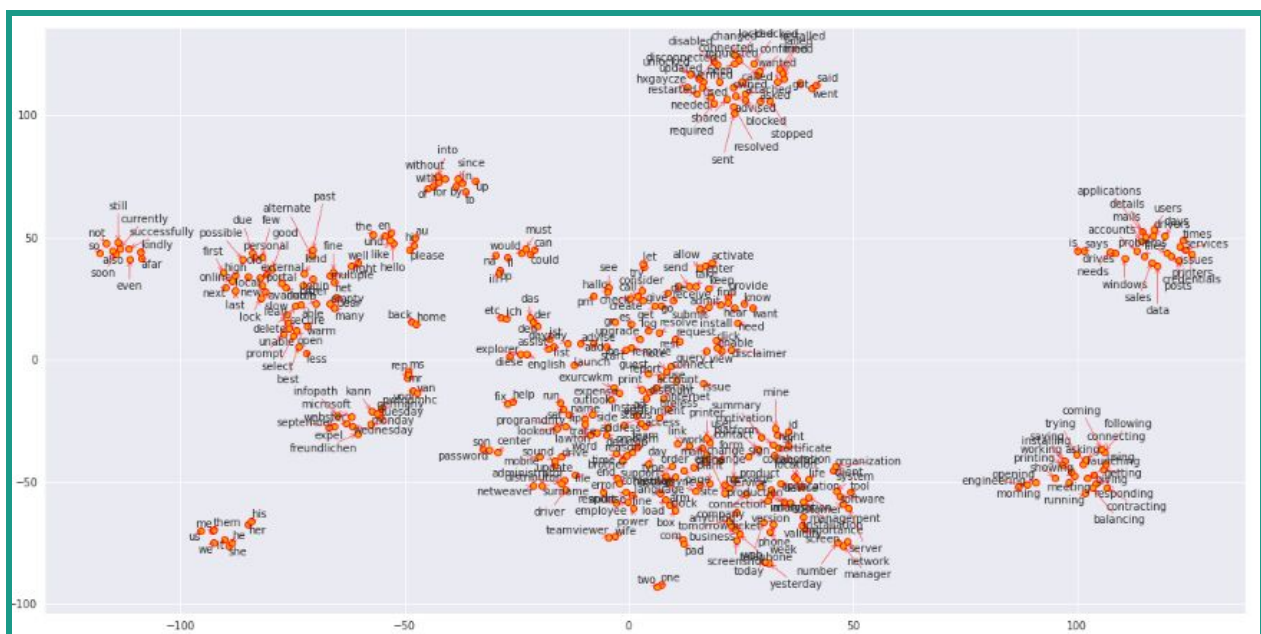


As part of the feature extraction process, we will find out the various terms being used and its frequency of occurrence. This will help us derive the following.


```
print("Terms sorted based on it's score. Last 10 ranks")
ranking.tail(10)
```

	term	rank
153	disclaimer alternate language	0.521492
151	disclaimer	0.521492
697	view disclaimer alternate	0.521492
28	admit	0.498490
103	company network	0.484563
434	password help	0.442776
122	contracting	0.402155
495	requested	0.385458
349	mail sent	0.374749
120	consider	0.342891

Word Embeddings Plotting



TF-IDF and word embeddings which helped to trim high and low-frequency words and use pre-trained word vectors for visualizing word corpus. From the above diagram, we can derive the following clusters:

Cluster1: Comprising of Notified, Locked, Failed, attached, received

Cluster2: VPN, Microsoft, Windows, CRM

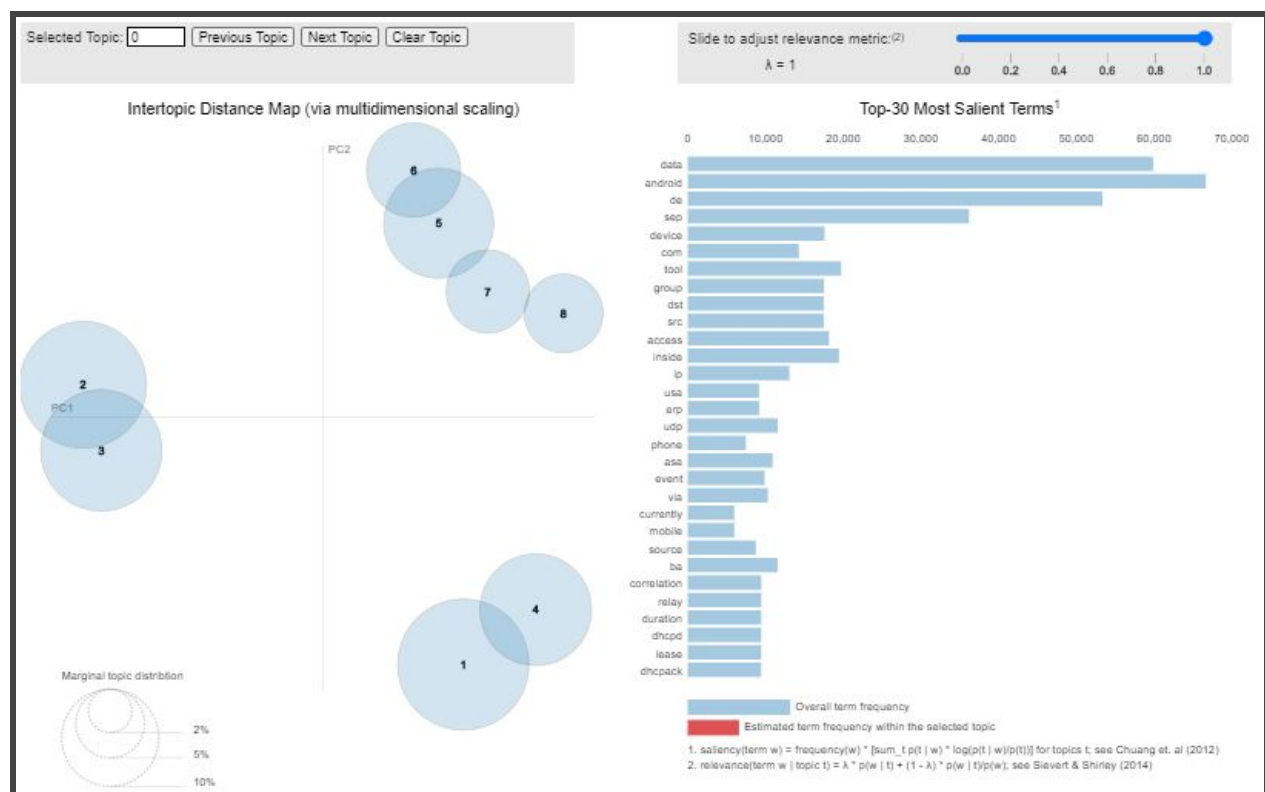
Cluster3: login,unable,backup

Cluster4: internet,email,error,power

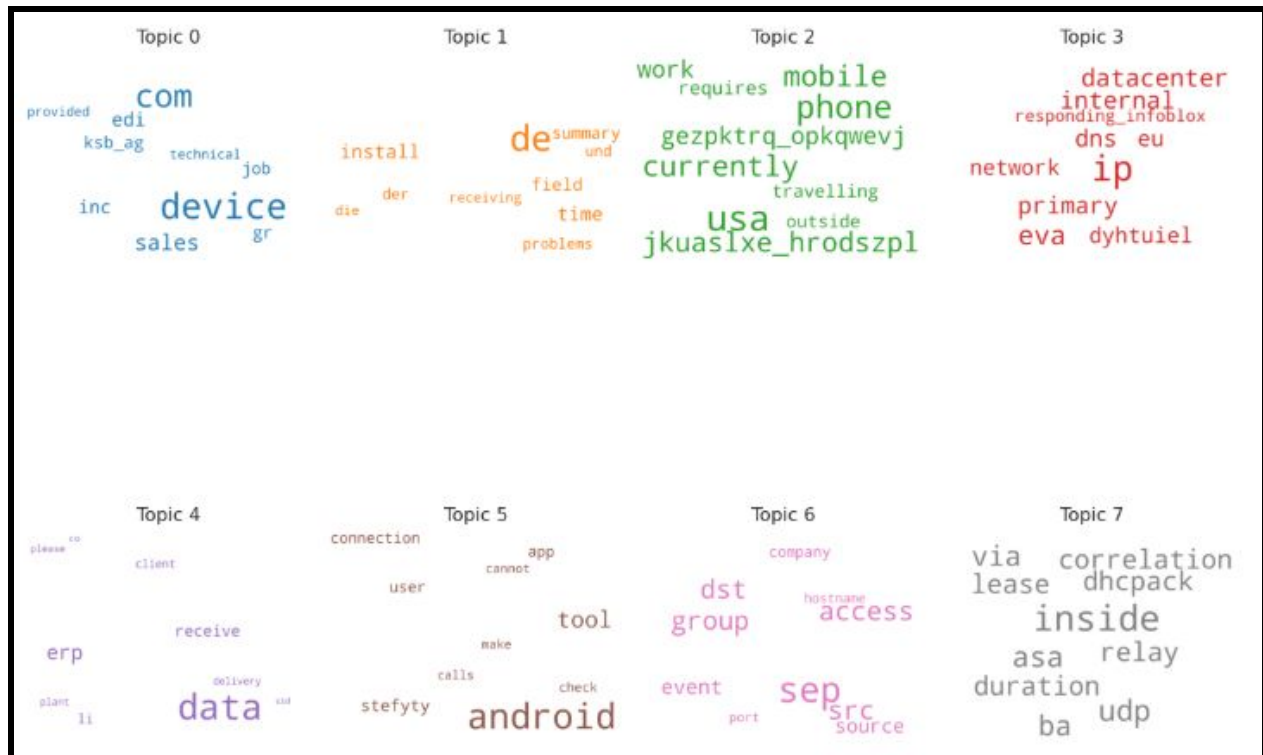
above clusters are derived based on the visualization, we can also list some more

LDA based feature engineering (Topic Representation based on LDA)

As part of the Feature Engineering Process, We used LDA Model using Gensim Library and are able to come up with 8 different topics where each topic is a combination of keywords and each keyword contributes a certain weightage to the topic. We decided to use 8 Topics as adding more Topics does not give enough details.



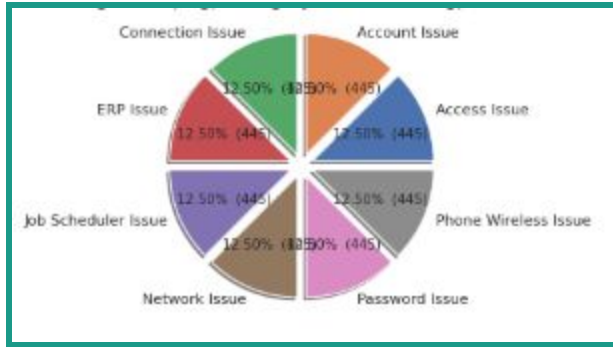
Topic and Word Cloud Representation



Topic and Number of Tickets Mapping

Network Issue	34322
Job Scheduler Issue	3685
Connection Issue	3035
Account Issue	1509
Password Issue	1287
ERP Issue	674
Access Issue	643
Phone Wireless Issue	445
Name: Topic, dtype: int64	


Plot the pie chart showing the percentage of data in different topics after LDA:



Based on the feature extraction using LDA, we were able to derive the potential Topics which fits well to the data set. Now we will apply the Modelling on the newly derived features.

Initially we ran our Models on the existing groups and found out that both SVM and LSTM Bidirectional works better. Now we will apply the same modelling technique but on the the newly generated Topics generated post LDA Feature Engineering Analysis.

```
[ ] from sklearn.svm import SVC
    clf = SVC(kernel='linear').fit(train_x, train_y)
    y_pred = clf.predict(test_x)
    acc_score = accuracy_score(test_y, y_pred)
    print("SVM-Linear Score: ", acc_score)
    f_sc = f1_score(test_y, y_pred, average='weighted', labels=np.unique(y_pred))
    print("SVM F1 Score: ", f_sc)
```

 SVM-Linear Score: 0.9892543859649123
 SVM F1 Score: 0.9891935712867673

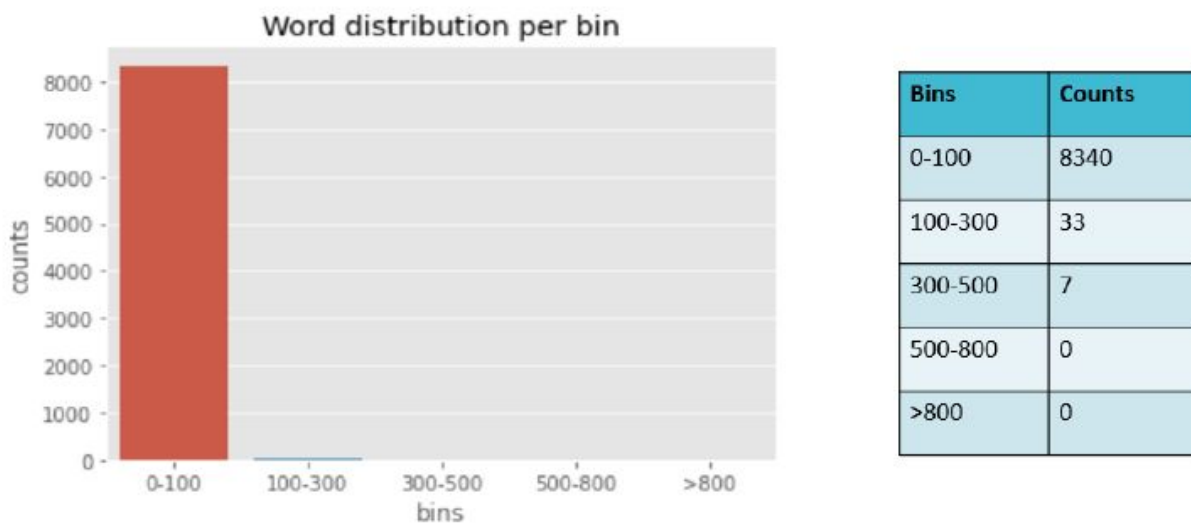
Analysis using WordCloud

WordCloud is a data visualization technique used for representing text data in which the size of each word indicates its frequency or importance. Significant textual data points can be highlighted using a word cloud. Let's visualize the frequent words in all tickets

Grp_3	boot, file, window, location, drive, run, document, computer, printer, update, dell, client, tool, replace, outlook, pc, erp, email, inside_outside, user, new, phone, print	System/OS related
Grp_6	job, fail, scheduler, abended	Job Scheduler related issues (similar to grp_48)
Grp_13	billing, sale, price, inwarehouse, delivery, material, customer, tool, block, can_not, quote, item, fix, time, unable, receive, erp, correct, note, print, send, system, check	Sales related issues

Word Distribution

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



Modeling

As the target class is completely skewed, various models have been tried with the augmented set of dataset of 54 groups to compare each performance.

Representing the raw text data as numerical data by doing vectorization

The raw data, a sequence of symbols cannot be fed directly to the algorithms themselves as most of them expect numerical feature vectors with a fixed size rather than the raw text documents with variable length. Before creating the above classifier models, let's first vectorize our input data.

Scikit-learn's CountVectorizer is used to transform a corpora of text to a vector of term / token counts. It also provides the capability to preprocess your text data prior to generating the vector representation making it a highly flexible feature representation module for text.

TF-IDF or Term Frequency(TF) — Inverse Dense Frequency(IDF) is a technique which is used to find meaning of sentences consisting of words and cancels out the incapacabilities of Bag of Words technique which is good for text classification or for helping a machine read words in numbers. Let's use this for running our base classification models. In text analysis with machine learning, TF-IDF algorithms help sort data into categories, as well as extract keywords. This means that simple, monotonous tasks, like tagging support tickets or rows of feedback and inputting data can be done in seconds.

Traditional ML Models

Multinomial Naive Bayes

Naive Bayes is a family of algorithms based on applying Bayes theorem with a strong(naive) assumption, that every feature is independent of the others, in order to predict the category of a given sample. They are probabilistic classifiers, therefore will calculate the probability of each category using Bayes theorem, and the category with the highest probability will be output. Naive Bayes classifiers have been successfully applied to many domains, particularly NLP.


```

clf = MultinomialNB().fit(X_train_tfidf, y_train)
y_train_pred_NB = clf.predict(count_vect.transform(X_train))
y_test_pred_NB = clf.predict(count_vect.transform(X_test))
print("Multinomial NaiveBayers :")
print('Training accuracy: %.2f%%' % (accuracy_score(y_train,y_train_pred_NB) * 100))
print('Testing accuracy: %.2f%%' % (accuracy_score(y_test, y_test_pred_NB) * 100))

```

```

Multinomial NaiveBayers :
Training accuracy: 81.65%
Testing accuracy: 80.16%

```

K Nearest Neighbor

KNN is a non-parametric and lazy learning algorithm. Non-parametric means there is no assumption for underlying data distribution. This will be very helpful in practice where most of the real world datasets do not follow mathematical theoretical assumptions. Lazy algorithm means it does not need any training data points for model generation. All training data used in the testing phase.

```

clf_knn = KNeighborsClassifier(n_neighbors=7,weights='uniform').fit(X_train_tfidf, y_train)
y_train_pred_knn = clf_knn.predict(count_vect.transform(X_train))
y_test_pred_knn = clf_knn.predict(count_vect.transform(X_test))
print("K Nearest Neighbours :")
print('Training accuracy: %.2f%%' % (accuracy_score(y_train,y_train_pred_knn) * 100))
print('Testing accuracy: %.2f%%' % (accuracy_score(y_test, y_test_pred_knn) * 100))

```

```

K Nearest Neighbours :
Training accuracy: 89.53%
Testing accuracy: 86.77%

```

Support Vector Machine

SVM (Support Vector Machine) classifies the data using hyperplane which acts like a decision boundary between different classes. Extreme data points from each class are called Support Vectors. SVM tries to find the best and optimal hyperplane which has maximum margin from each Support Vector. Support vectors are the data points, which are closest to the hyperplane. These points will define the separating line better by calculating margins.

The linear kernel is often recommended for text classification.

```

clf_svc = LinearSVC().fit(X_train_tfidf, y_train)
y_train_pred_svc = clf_svc.predict(count_vect.transform(X_train))
y_test_pred_svc = clf_svc.predict(count_vect.transform(X_test))
print("Support Vector Machine :")
print('Training accuracy: %.2f%%' % (accuracy_score(y_train,y_train_pred_svc) * 100))
print('Testing accuracy: %.2f%%' % (accuracy_score(y_test, y_test_pred_svc) * 100))

```

```

Support Vector Machine :
Training accuracy: 92.41%
Testing accuracy: 90.89%

```

Decision Tree

A decision tree is a flowchart-like tree structure where an internal node represents feature(or attribute), the branch represents a decision rule, and each leaf node represents the outcome.The topmost node in a decision tree is known as the root node. It learns to partition on the basis of the attribute value. It partitions the tree in a recursive manner call recursive partitioning. This flowchart-like structure helps you in decision making. It's visualization like a flowchart diagram which easily mimics the human level thinking. That is why decision trees are easy to understand and interpret.

```

clf_tree = DecisionTreeClassifier().fit(X_train_tfidf, y_train)
y_train_pred_tree = clf_tree.predict(count_vect.transform(X_train))
y_test_pred_tree = clf_tree.predict(count_vect.transform(X_test))
print("Decision Tree Classifier :")
print('Training accuracy: %.2f%%' % (accuracy_score(y_train,y_train_pred_tree) * 100))
print('Testing accuracy: %.2f%%' % (accuracy_score(y_test, y_test_pred_tree) * 100))

```

```

Decision Tree Classifier :
Training accuracy: 53.18%
Testing accuracy: 52.10%

```

RandomForest Classifier

Due to its algorithmic simplicity and prominent classification performance for high dimensional data, random forest has become a promising method for text categorization. Random forest is a popular classification method which is an ensemble of a set of classification trees.

```

clf_rand = RandomForestClassifier(n_estimators=100).fit(X_train_tfidf, y_train)
y_train_pred_rand = clf_rand.predict(count_vect.transform(X_train))
y_test_pred_rand = clf_rand.predict(count_vect.transform(X_test))
print("RandomForest Classifier:")
print('Training accuracy: %.2f%%' % (accuracy_score(y_train,y_train_pred_rand) * 100))
print('Testing accuracy: %.2f%%' % (accuracy_score(y_test, y_test_pred_rand) * 100))

```

RandomForest Classifier:
 Training accuracy: 84.49%
 Testing accuracy: 82.48%

AdaBoostClassifier

```

from sklearn.ensemble import AdaBoostClassifier
adaclassifier = AdaBoostClassifier( n_estimators= 100, learning_rate=0.1, random_state=22)
adaclassifier = adaclassifier.fit(X_train_tfidf, y_train)
y_train_pred_ad = adaclassifier.predict(count_vect.transform(X_train))
y_test_pred_ad = adaclassifier.predict(count_vect.transform(X_test))
print("Adaboost Classifier:")
print('Training accuracy: %.2f%%' % (accuracy_score(y_train,y_train_pred_ad) * 100))
print('Testing accuracy: %.2f%%' % (accuracy_score(y_test, y_test_pred_ad) * 100))

```

Adaboost Classifier:
 Training accuracy: 11.99%
 Testing accuracy: 12.20%

BaggingClassifier

```

from sklearn.ensemble import BaggingClassifier
bgclassifier = BaggingClassifier(n_estimators=150, max_samples= .7, bootstrap=True, oob_score=True, random_state=22)
bgclassifier = bgclassifier.fit(X_train_tfidf, y_train)
y_train_pred_bgc = bgclassifier.predict(count_vect.transform(X_train))
y_test_pred_bgc = bgclassifier.predict(count_vect.transform(X_test))
print("Bagging Classifier:")
print('Training accuracy: %.2f%%' % (accuracy_score(y_train,y_train_pred_bgc) * 100))
print('Testing accuracy: %.2f%%' % (accuracy_score(y_test, y_test_pred_bgc) * 100))

```

Bagging Classifier:
 Training accuracy: 60.59%
 Testing accuracy: 58.88%

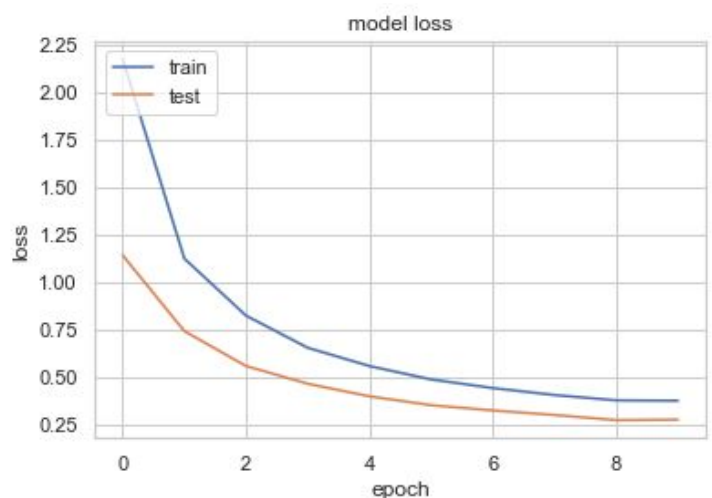
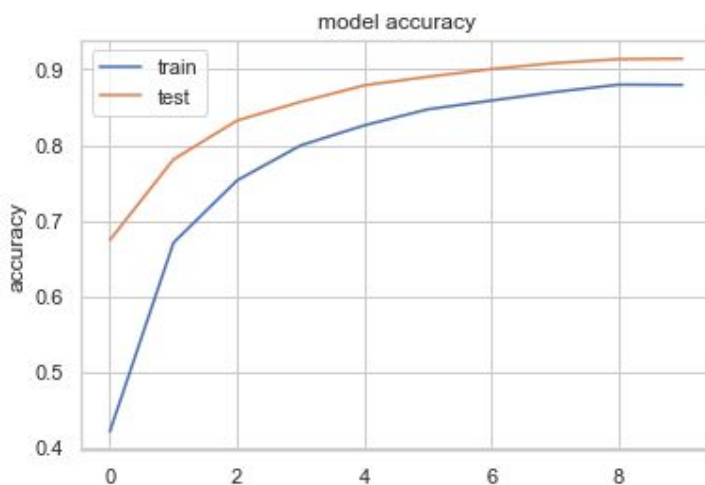
Bi-LSTM Model

LSTM stands for Long short-term memory. An LSTM module (or cell) has 5 essential components which allows it to model both long-term and short-term data. LSTM is a special type of RNN that preserves long term dependency in a more effective way compared to the basic RNNs. This is particularly useful to overcome the vanishing gradient problem as LSTM uses multiple gates to carefully regulate the amount of information that will be allowed into each node state. LSTM in its core, preserves information from inputs that have already passed through it using the hidden state. Unidirectional LSTM only preserves information of the past because the only inputs it has seen are from the past.

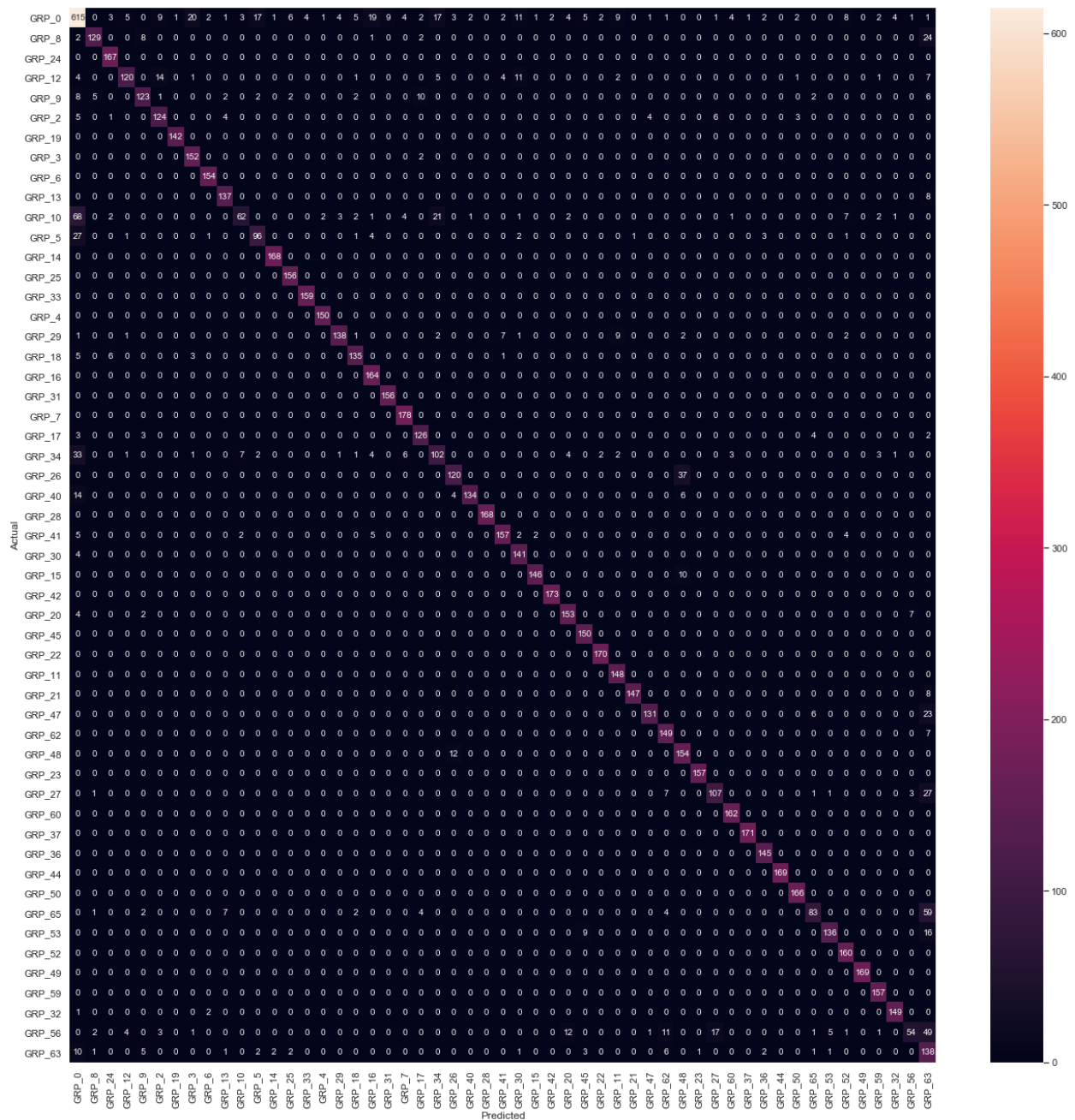
Bidirectional LSTMs are an extension of traditional LSTMs that can improve model performance on classification problems.

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 from Bi-LSTM Model



Confusion Matrix



Inference from Confusion Matrix

The diagonal elements represent the number of points for which the predicted label is equal to the true label, while off-diagonal elements are those that are mislabeled by the classifier. The higher the diagonal values of the confusion matrix the better, indicating many correct predictions.

Many Assignment groups are not present in the test data. The diagonal element value for GRP_0 is high

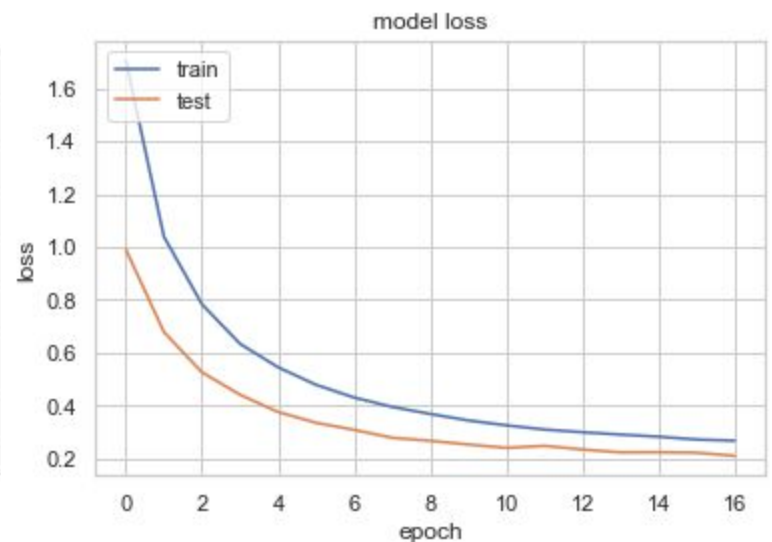
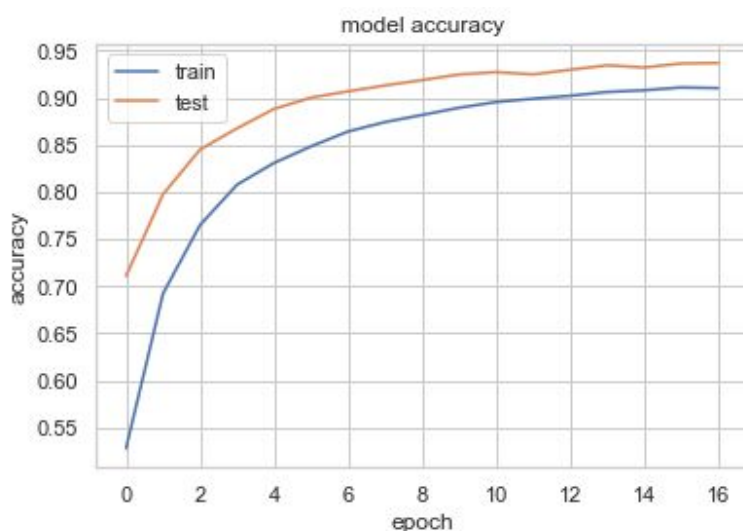
Tuning of LSTM Model

1. LSTM Merge Mode

The Bidirectional wrapper layer also allows to specify the merge mode, that is how the forward and backward outputs should be combined before being passed on to the next layer. The options are: 'sum': The outputs are added together. 'mul': The outputs are multiplied together. 'concat': The outputs are concatenated together (the default), providing double the number of outputs to the next layer. 'ave': The average of the outputs is taken.

'concat' is the default merge mode. Merge mode 'mul' and 'ave' didn't show any improvements in F1 score. However, merge mode of 'sum' showed improved F1 score.

Look at the following results with 17 epochs. Fit an LSTM model with `merge_mode="sum"`

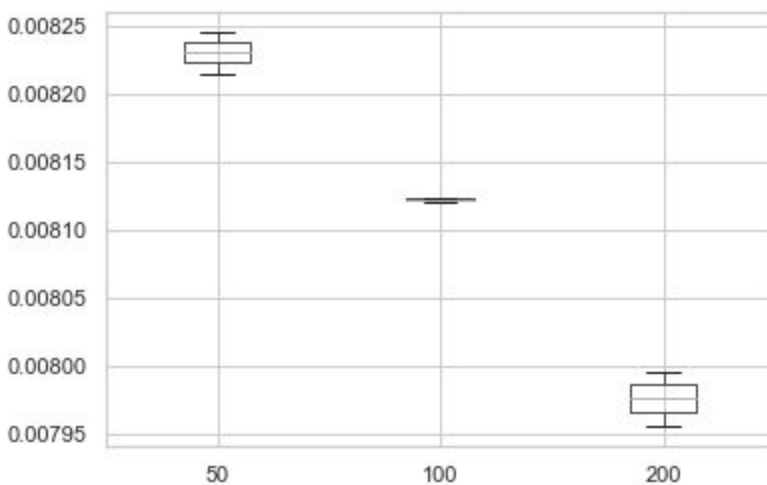


Observations (When compared to the original model created in Milestone 2):

With LSTM merge mode SUM, Test accuracy has improved to 93.67%, however training accuracy improved to 94.89%. The average F1 Score of the model is 0.57. We can go ahead with SUM.

2.Number of LSTM Cells

We cannot know the best number of memory cells for a given LSTM architecture. We must test a suite of different memory cells in LSTM hidden layers to see what works best. Let's take 3 different numbers of LSTM cells, 50, 100 and 200.

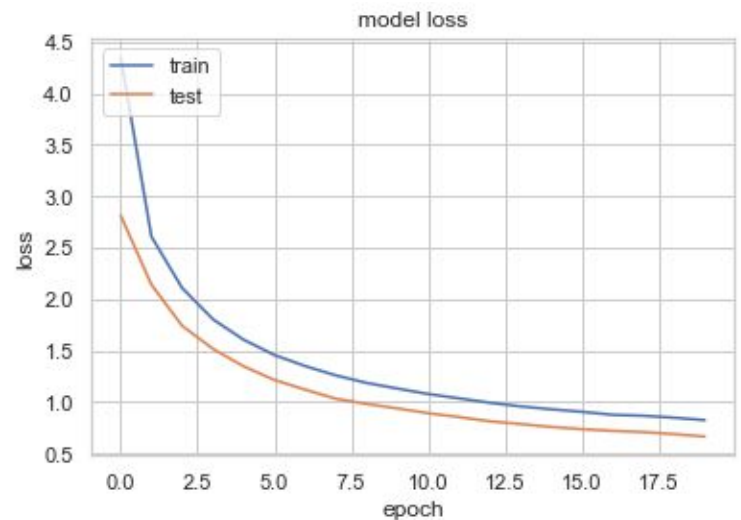
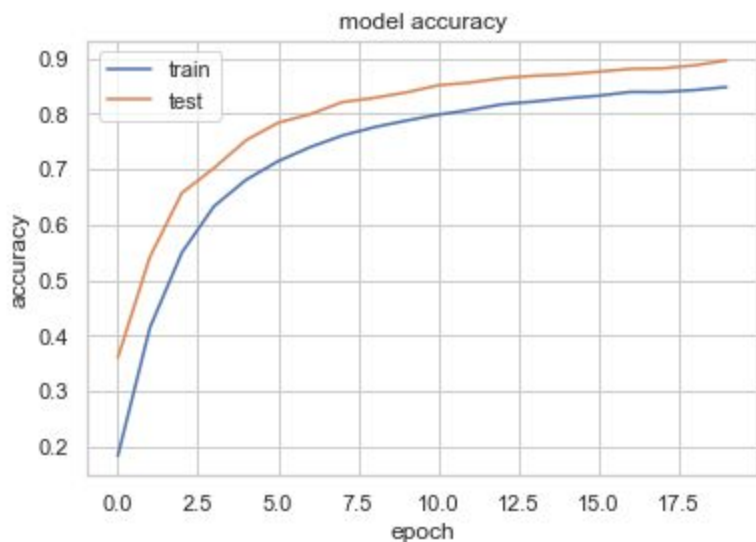


Observations (When compared to the original model created in Milestone 2):

By Increasing the number of LSTM cells from 100 to 200 we can see the reduction in overall loss.

3. Regularization

LSTMs can quickly converge and even overfit on some sequence prediction problems. To counter this, regularization methods can be used. LSTMs support regularization such as weight regularization that imposes pressure to decrease the size of network weights. Again, these can be set on the LSTM layer with the arguments.



Observations:

By adding regularization on the Dense layer using `kernel_regularizer` and `activity_regularizer`, no improvement is seen on train and validation data. F1 score dropped from 0.57 to 0.51 may be because of less data related to other categories.

4. Pipeline

Pipelines work by allowing for a linear sequence of data transforms to be chained together culminating in a modeling process that can be evaluated. Python scikit-learn provides a Pipeline utility to help automate machine learning workflows. The goal is to ensure that all of the steps in the pipeline are constrained to the data available for the evaluation, such as the training dataset or each fold of the cross validation procedure.

Here we got 92% accuracy.

Observations:

Pipeline and Feature Union as such does not improve performance of the models. Its adds more value by combining different rules and models, we can define our own transformers that will improve the performance. Here we have done the basic pipeline model.

Pipelines help in optimizing the entire workflow, preventing data leakage and code simplicity.



4. Model Evaluation

Training Accuracy for each model

Out of all the models we've tried, in traditional ML algorithms, Support Vector Machine and RandomForestClassifier are performing better than all others. But these models are highly overfitted.

LSTM is efficient for dealing with textual data. Bidirectional LSTMs are an extension of traditional LSTMs that can improve model performance on classification problems. Using bidirectional will run your inputs in two ways, one from past to future and one from future to past and what differs this approach from unidirectional is that in the LSTM that runs backwards you preserve information from the future and using the two hidden states combined you are able in any point in time to preserve information from both past and future.

Model	Train_Accuracy	Test_Accuracy
Multinomial NB	81.61	80.17
K Nearest Neighbours	88.99	86
Support Vector Machine	92.38	90.68
Decision Tree Classifier	51.47	49.92
RandomForest Classifier	83.68	82.1
Adaboost Classifier	11.99	12.2
Bagging Classifier	60.59	58.88
Bidirectional LSTM(epochs=10)	89.69	87.9
Bidirectional LSTM (merge_mode="sum", epochs=17)	94.89	93.67
Bidirectional LSTM (Regularization on Dense layer)	91.01	89.56
LDA GENSIM - SVM Classifier	94.05	92.56

Other Models Tried

1. BERT

BERT stands for **Bidirectional Encoder Representations from Transformers**. It is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of NLP tasks.

1. BERT is based on the Transformer architecture.
2. 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).
3. 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.

We got the F1 score 94% in case of BERT Model.

2. LDA-Mallet

In addition to this we used an advanced version of Topic Modelling and considered LDA Modeling using Mallet Distribution. In this case also SVM gives a better accuracy. For more details please refer to github repository.



5. Comparison to Benchmark

From the given problem description, we could see that the existing system is able to assign 75% of the tickets correctly.

So our objective here is to build an AI-based classifier model to assign the tickets to right functional groups by analysing the given description with an accuracy of at least 85%.

From the prediction results we see that the BLSTM, LDA GENSIM - SVM Classifier & BERT model based on the augmented data are able to achieve an accuracy more than 91% which is above our benchmark.



6. Implications

Although the above model can classify the IT tickets with good accuracy, to achieve better accuracy in the real world it would be good if the business can collect additional data around 300 records for each group.



7. Limitations

During data pre-processing, we had removed deterministic groups with which have lesser number of ticket assignments (in 10's max). If there are standard patterns found in ticket assignment to groups based on content of short description and description then such groups are removed from the Machine Learning process. It results in a reduction of Target class from 74 to 54 groups.

When a new ticket arrives it will be first evaluated against the deterministic rules with the help of patterns matching against short description and description. If there are no deterministic groups found then ML model prediction will be run. (Evaluation method is provided in the data_rules.py file.)

By above approach, we have eliminated the limitation of having probability to miss the classification of any new ticket even Machine Learning done on 54 groups.



8. Closing Reflections

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

Final Conclusion

In this project, a model based on supervised machine learning algorithms is proposed to assign tickets automatically. Preprocessed dataset consisting of previously categorized tickets are used to train classification algorithms. We have implemented different classification algorithms to evaluate performances comparatively. We tried tuning the model using different hyper parameters for better performance and we have achieved the planned target.



9. Appendix

Github Link

[https://github.com/kawadeshailesh1981/Capstone-NLP-Automatic Ticket Assignment](https://github.com/kawadeshailesh1981/Capstone-NLP-Automatic_Ticket_Assignment)