



AIML ONLINE CAPSTONE

FINAL REPORT DOCUMENT

[Abstract](#)

Build NLP based AIML classifier that can classify the tickets by analyzing text

Project Team

Jayesh Salunke

Sunil A.K.

Sunil Dara

Suresh Kumar A.P.

Mentor

Nishant Parikshit

Table of Contents

Contents

1. Project Overview	3
1.1. Problem Overview	3
1.2. AS/IS Process	3
1.3. Problem Statement	3
1.4. Project Objective	4
1.5. Assumptions	4
1.6. Proposed solution	4
1.7. Evaluation Metrics	5
2. Exploratory data analysis	6
2.1. Data File exploration	6
2.2. Language	6
2.3. Data inconsistencies	7
2.4. Missing points in data	7
2.5. Dealing with missing data	7
2.6. Description Field Properties	8
2.7. Word Analysis	8
2.8. Class Labels	8
3. Exploratory visualization	9
3.1. Language Distribution:	9
3.2. Class Distribution:	9
3.3. Word cloud	10
3.3.1. Word cloud for Description feature	11
3.3.2. Word cloud for Short Description feature	11
3.4. N-gram Exploration	13
3.4.1. Bi-gram	13
3.4.2. Tri-gram	13
3.5. LDA – Topic modelling exploration with pyLDAvis	14
3.6. Word Vectors with tSNE	17
4. Text pre-processing:	18
4.1. Data cleansing / Data inconsistencies	18
4.2. Impute missing data	18
4.3. Data Pre-processing:	19
5. Word Vocabulary / Word Embedding:	19

6.	Feature Engineering	20
7.	Model Building	21
7.1.	Base model:	21
7.2.	Hyper Parameter Tuning using GridSearchCV	23
7.3.	Deep Learning Models –Basic NN & LSTM	24
7.3.1.	Neural Network Model	25
7.3.2.	LSTM(RNN) with GloVe Embeddings	25
7.4.	Transfer learning using ULMFit	26
8.	Model evaluation	28
9.	Benchmark Comparison	30
10.	UI App Model deployment	32
11.	Implications	36
12.	Limitations	36
13.	Closing Reflections	36
14.	References	37

1. Project Overview

1.1. Problem Overview

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 the Incident Management process to achieve the above Objective.

An incident is something that is an 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 the 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.

1.2. AS/IS Process

Currently the incidents are created by various stakeholders (Business Users, IT Users and Monitoring Tools) within the 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

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 the right functional groups. During this process, some of the incidents are in queue and not addressed timely resulting in poor customer service.

1.3. Problem Statement

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. Currently the L1/L2 team is facing issues in the assignment of tickets to correct L3 functional groups as well as spending ~ 1 FTE effort for incident assignment to the L3 team.

1.4. Project Objective

Using NLP based AI techniques build a classifier that can automatically classify incidents to right functional groups, in turn can help organizations to reduce the resolving time of the issue and can focus on more productive tasks.

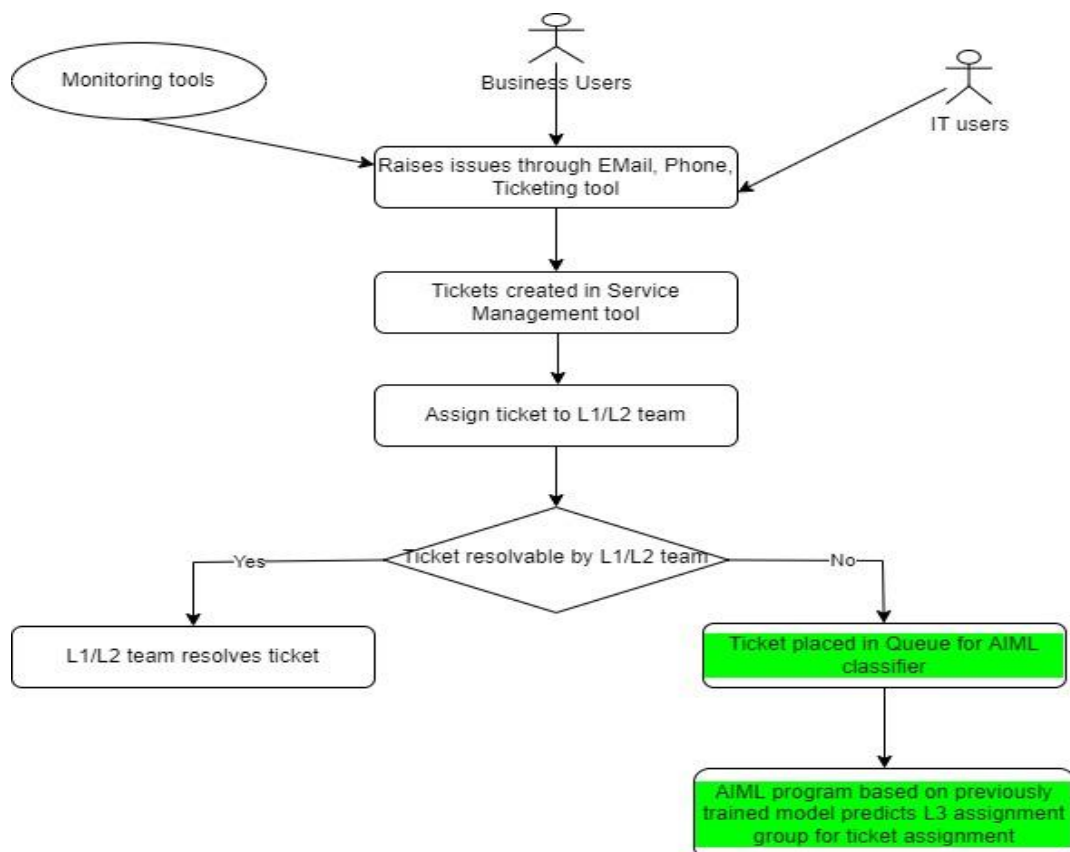
1.5. Assumptions

As per the problem statement, only L3 tickets need automation for assignment of tickets.

There is no specific guideline given for which tickets are L1/L2 or L3. Though tickets like password reset management, drive folder access which primarily assigned to Group 0 and Group 8 can be considered as L1/L2 tickets, but our solution would assume to do classification of all tickets in the data

For class imbalance, we would go with merging groups with less than 10 tickets to Group others, we would also try addressing smaller number of groups in 1 or 2 models (number of classes restricted to 10 classes)

1.6. Proposed solution



Following steps would be followed in the proposed solution.

- Create ticket through Ticket management tools
- Tool to assign tickets to L1/L2 team

- L1/L2 team does Initial diagnosis and 54% of the ticket they resolve and remaining tickets to be assigned to L3 team
- Remaining L3 tickets to be placed in AIML job queue for classification
- Using Natural Language Processing techniques, parse the ticket description and assign it to correct L3 group for further resolution

1.7. Evaluation Metrics

Since it is a multi-class classification, NLP model metrics are evaluated using the below metrics:

- Weighted F1 score
- Weighted precision
- Weighted recall
- Weighted accuracy.

2. Exploratory data analysis

2.1. Data File exploration

We had a minimal dataset with 8500 records. Each record had details below.

Column Name	Description	Feature/Target
Short Description	Short description about ticket	Feature
Description	Problem explained in detail by user	Feature
Caller	User for whom ticket is created	Feature
Assignment group	IT group to which ticket needs to be assigned	Target

2.2. Language

We found multi-language data in both Short Description and Long description. Majority were English and other major languages were German, Afrikaans, Italy and French.

We followed the following approach for Language detection and Translation.

Language detection was done using **LangDetect**.

Translation was tried using below option,

- **GoogleTrans**
 - **Pyclid2**
 - **goslate**
 - **spacy language detection**
-
- Pyclid2 had issues when a specific description had multi languages like “probleme mit laufwerk z: \laeusvjo fvaihgp” had German and Lang detect for z: was different language (sl instead of de).
 - Spacy needed more learning curve (adding some custom code).
 - goslate uses internally GoogleTrans
 - Deciding to go with Google trans version **4.0.0-rc1**, Google Trans had frequent 429 error codes (too many requests); added a sleep of 1 sec to overcome this. Since the whole process is time consuming, we did a one-time translation and stored it in a csv file format and reused it for further analysis and development.

2.3. Data inconsistencies

Input data had below issues

- Newline characters
- Carriage-return characters and
- Accented characters.

	Short description	Description	Caller	Assignment group
0	login issue	-verified user details.(employee# & manager na...	spxjnwir pjlcoqds	GRP_0
1	outlook	_x000D_\n_x000D_\nreceived from: hmjdrvpb.komu...	hmjdrvpb komuaywn	GRP_0
2	cant log in to vpn	_x000D_\n_x000D_\nreceived from: eylqgodm.ybqk...	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
222	support fÃ¼r fa.gstry \arexjftu ohxdwngl	support fÃ¼r fa.konnica \arexjftu ohxdwngl	arexjftu ohxdwngl	GRP_24

2.4. Missing points in data

Data also had null values for Description (1) and Short Description (8)

#	Column	Non-Null Count	Dtype
0	Short description	8492 non-null	object
1	Description	8499 non-null	object
2	Caller	8500 non-null	object
3	Assignment group	8500 non-null	object

2.5. Dealing with missing data

Wherever description was missing, consider short description and vice versa. If both are null, we have dropped the rows

2.6. Description Field Properties

- Max length of ticket description ~ 7600
- Min length of ticket description 3
- Mean length of ticket description ~ 205

- Max words in a ticket description ~ 1500
- Min words in a ticket description 2
- Average words in a ticket description 35
- Mostly we have a range from 5 to 50 words in a ticket

2.7. Word Analysis

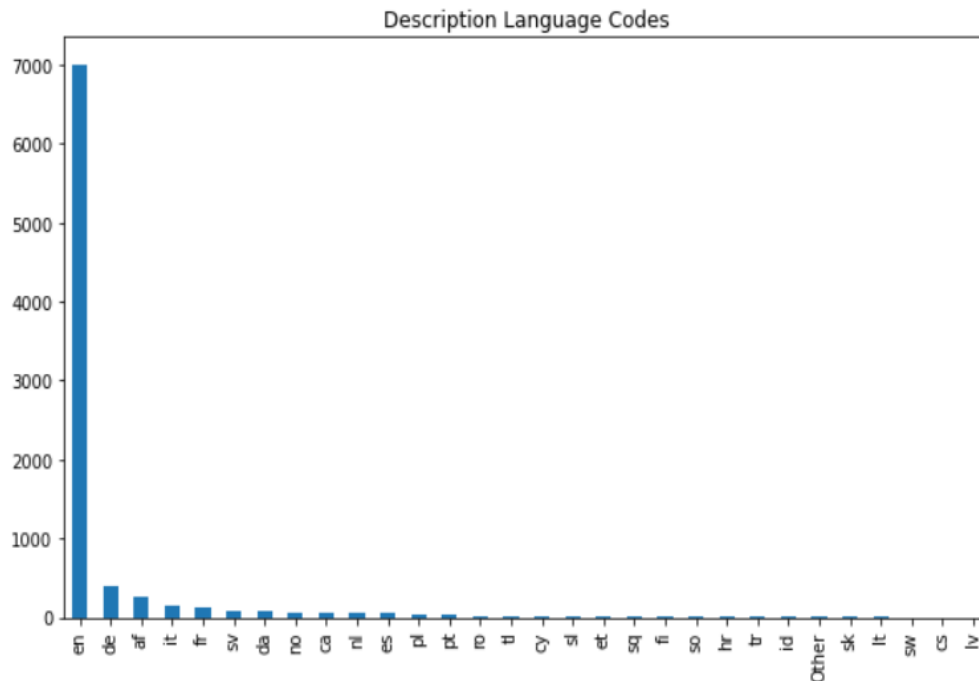
Corpus for word analysis was built using local data set and we used WordCloud, N-gram exploration, LDA, Word2Vec with t-SNE for Word Analysis

2.8. Class Labels

- Total Number of Classes (assignment group) – 74;
- Data is highly imbalanced Group 0 has close to 4000 records (50% of the data) while there are 25 groups with less than 10 rows.
- All the Group (25 groups) with less than 10 rows are moved to Group others as part of reducing Class Imbalance to some extent

3. Exploratory visualization

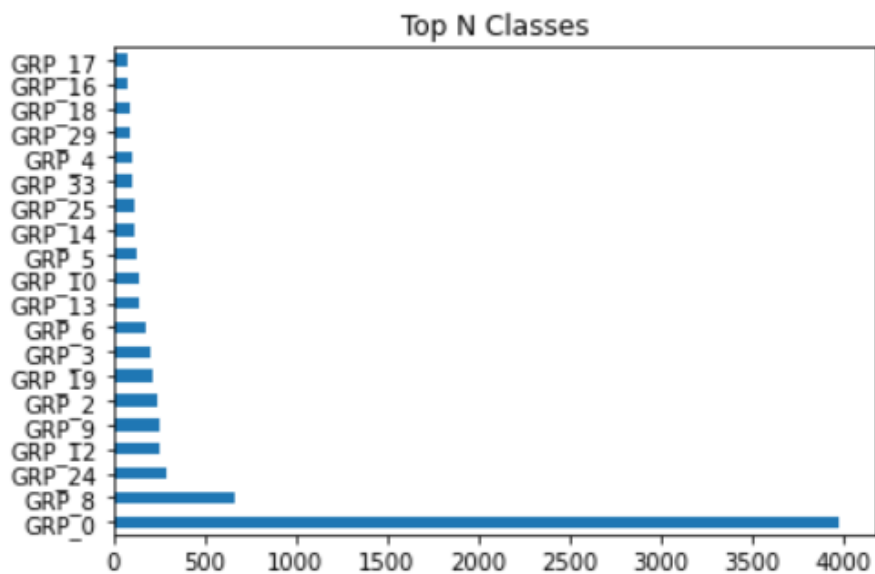
3.1. Language Distribution:



We found multi-language data in both Short Description and long description. Majority were English and other major languages were German, Afrikaans, Italy and French

3.2. Class Distribution:

- Plot below shows Class distribution for Top 20 assignment groups.

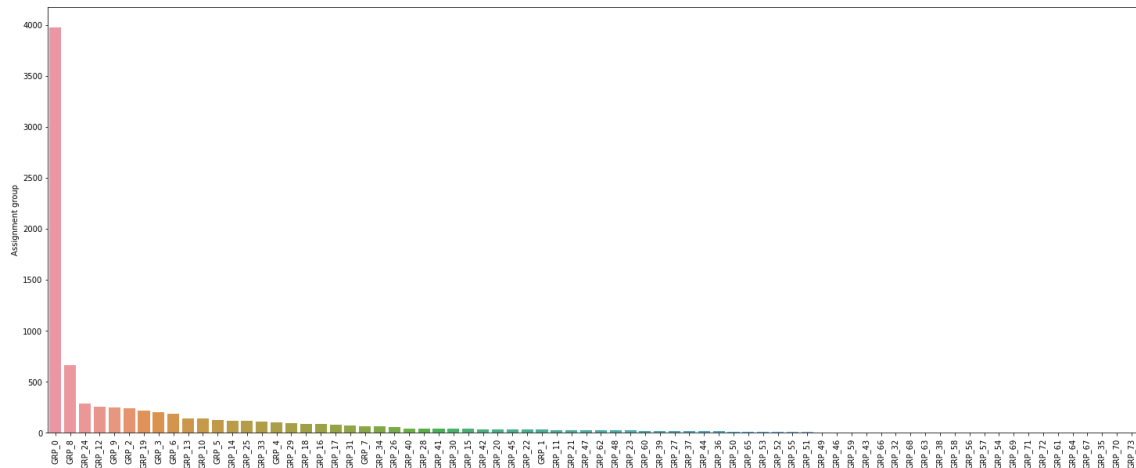


Group 0 and 8 accounts for 50% of the data

- Plot below shows class distribution for all given assignment groups in the data

Total Number of assignment group – 74;

We notice class imbalance. Group 0 has close to 4000 records (50% of the data) while there are 25 groups with less than 10 rows



3.3. Word cloud

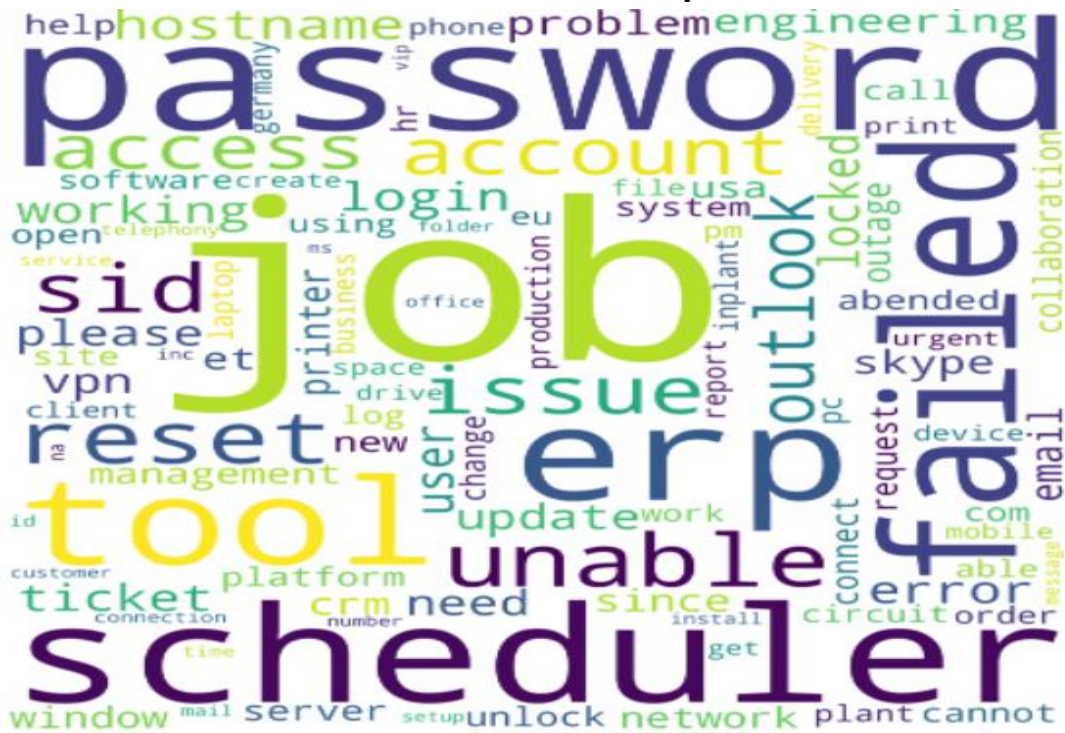
A word cloud (also known as a tag cloud) is a visual representation of words. Cloud creators are used to highlight popular words and phrases based on frequency and relevance. They provide you with quick and simple visual insights that can lead to more in-depth analyses.

3.3.1. Word cloud for Description feature



- Top words in Word cloud for description clearly indicate various issue area like tool, job, password, scheduler, unable to access, account reset

3.3.2. Word cloud for Short Description feature



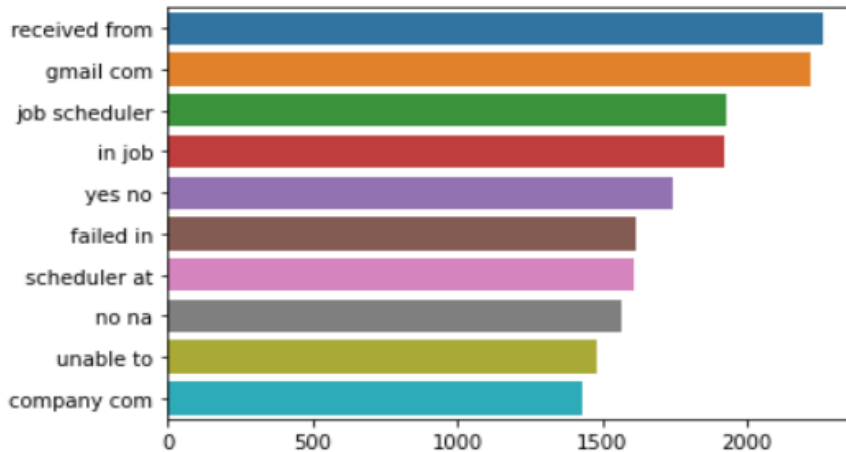
- Top words in Word cloud for short description also indicate various issue area like tool, job, password, scheduler, unable to access, account reset

Both Short Description and Description has similar words, but there are bunch of words addition in both fields, it would be good idea to use both features for ticket classification and assignment

3.4. N-gram Exploration

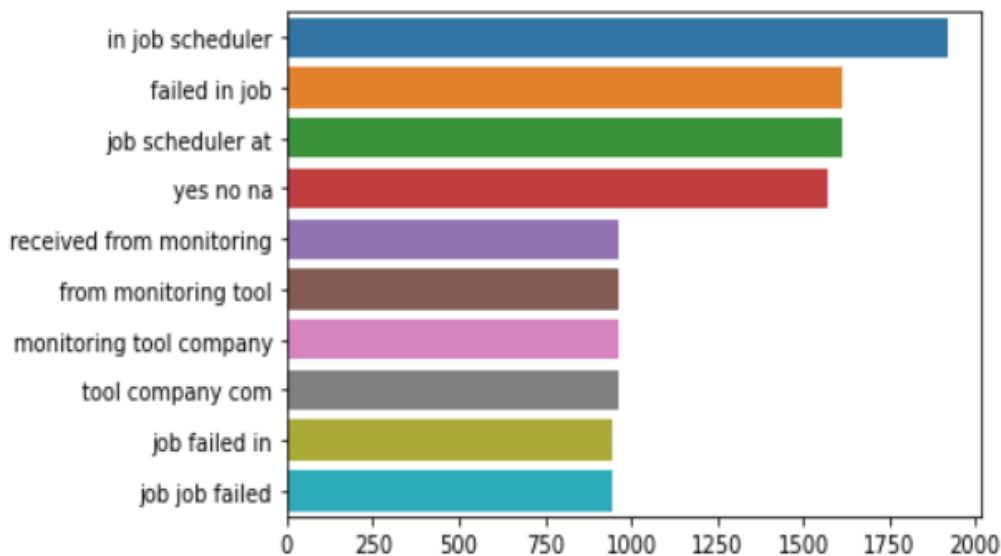
N-grams are simply contiguous sequences of n words. If the number of words is two, it is called bigram. For 3 words it is called a trigram and so on. Looking at most frequent n-grams can give you a better understanding of the context in which the word was used

3.4.1. Bi-gram



- Context of the word - 'job scheduler', 'failed in', 'unable to', 'gmail com' indicating issues in Jobs and also unable to access gmail com

3.4.2. Tri-gram



- Repeated combinations for groups of words 'Job scheduler at', 'from monitoring tool', 'job failed in', indicates Stemming/Lemmatization needs to be done in the word corpus.

3.5. LDA – Topic modelling exploration with pyLDAvis

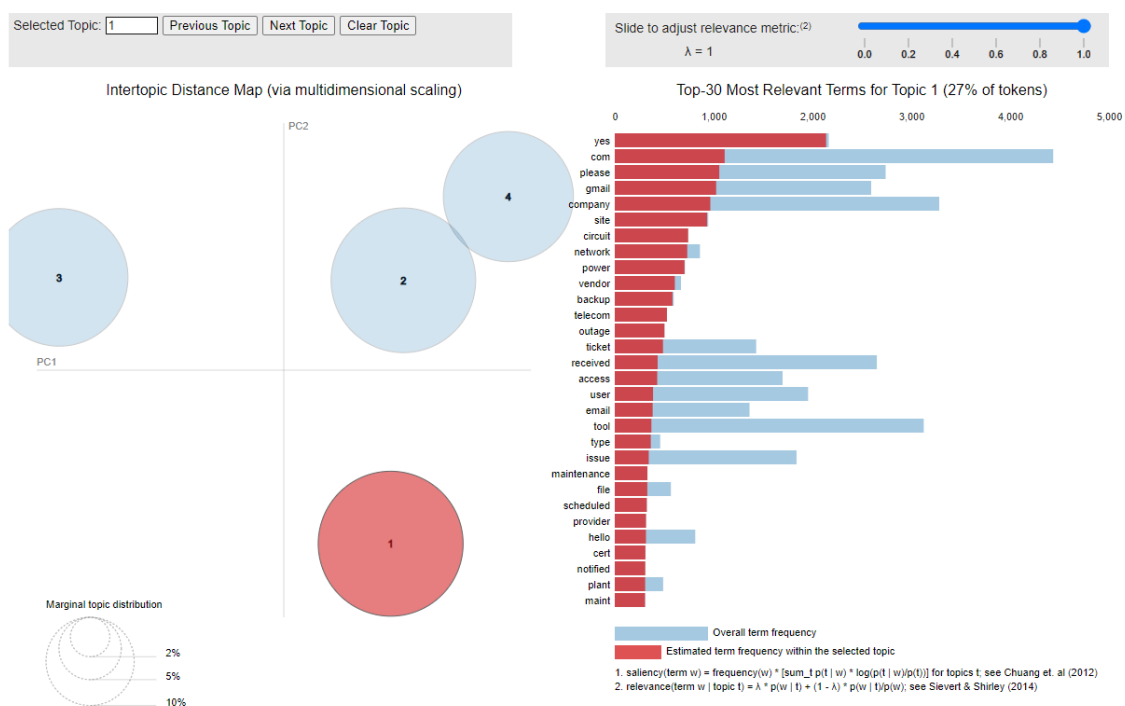
Topic modelling is the process of using unsupervised learning techniques to extract the main topics that occur in a collection of documents.

Latent Dirichlet Allocation (LDA) is an easy to use and efficient model for topic modelling. Each document is represented by the distribution of topics and each topic is represented by the distribution of words.

Figure below shows Topic modelling for the entire dataset with Topic 1 highlighted.

LDA on entire data set show 4 important topics

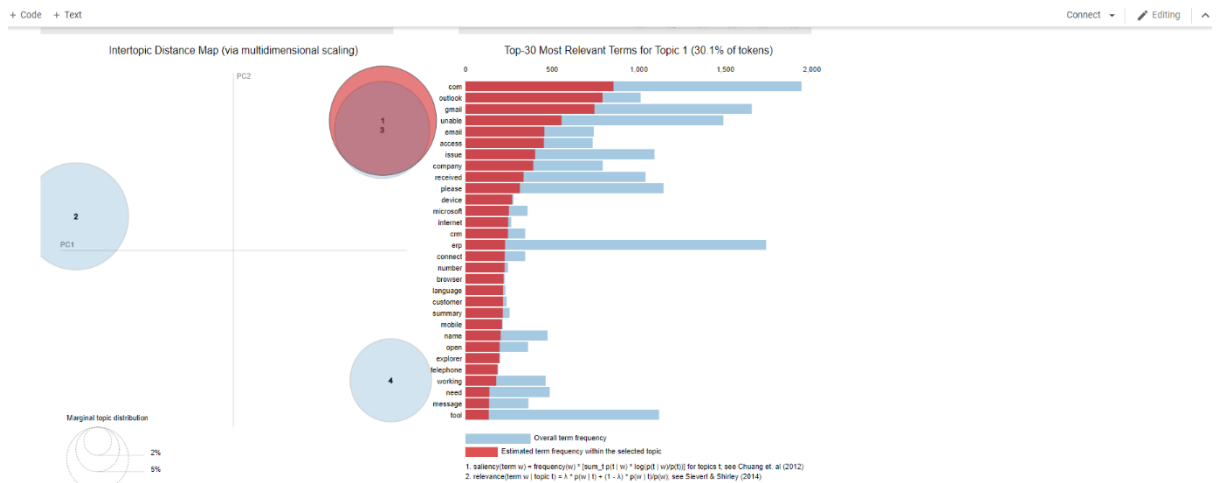
1. Password reset; job scheduler / tool failed
2. Outlook issues, account locked, error all related to erp
3. Email issues, event, unable to access, hostname for Gmail and Microsoft
4. Site, circuit, network, vendor, power



Similarly, LDA Analysis was done for Top 3 Assignment groups to find out specific topics for which respective assignment group deals with

- GRP_0
- GRP_8
- GRP_24

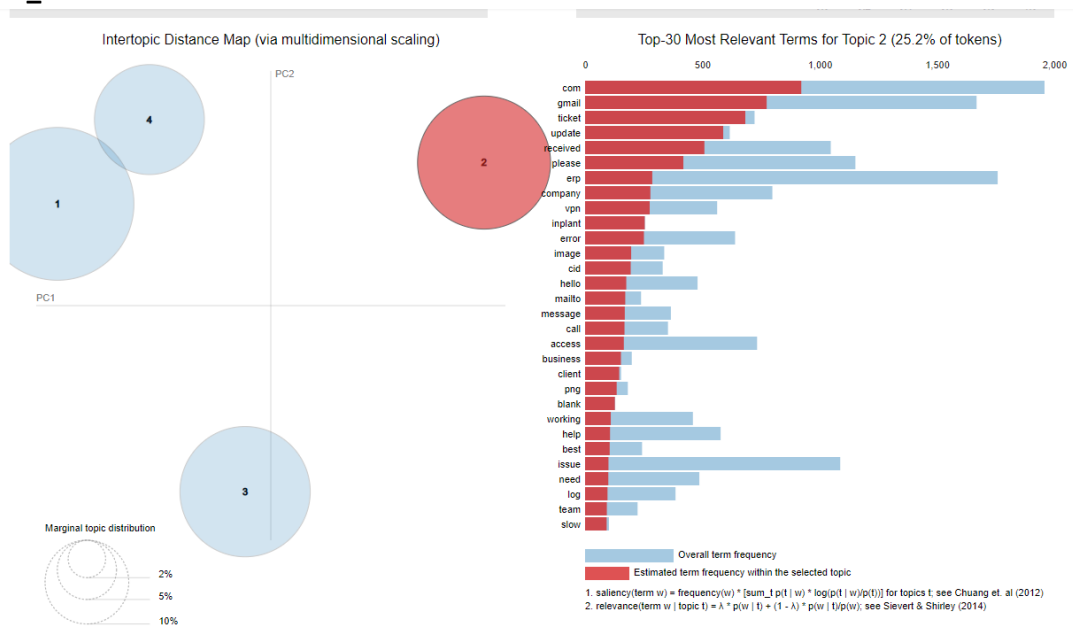
GRP_0



Top 4 topics in GRP_0

- (0, '0.028*"ticket" + 0.026*"update" + 0.025*"com" + 0.022*"gmail" + 0.019*"received" + 0.014*"vpn" + 0.013*"please" + 0.013*"image" + 0.012*"tool" + 0.012*"cid"')
- (1, '0.026*"unable" + 0.025*"skype" + 0.018*"call" + 0.016*"error" + 0.015*"user" + 0.015*"printer" + 0.014*"erp" + 0.014*"issue" + 0.013*"tool" + 0.011*"engineering"')
- (2, '0.098*"password" + 0.051*"reset" + 0.044*"account" + 0.044*"erp" + 0.034*"user" + 0.030*"sid" + 0.025*"locked" + 0.025*"login" + 0.018*"com" + 0.018*"tool"')
- (3, '0.030*"com" + 0.027*"outlook" + 0.026*"gmail" + 0.019*"unable" + 0.016*"email" + 0.016*"access" + 0.014*"issue" + 0.014*"company" + 0.012*"received" + 0.011*"please"')

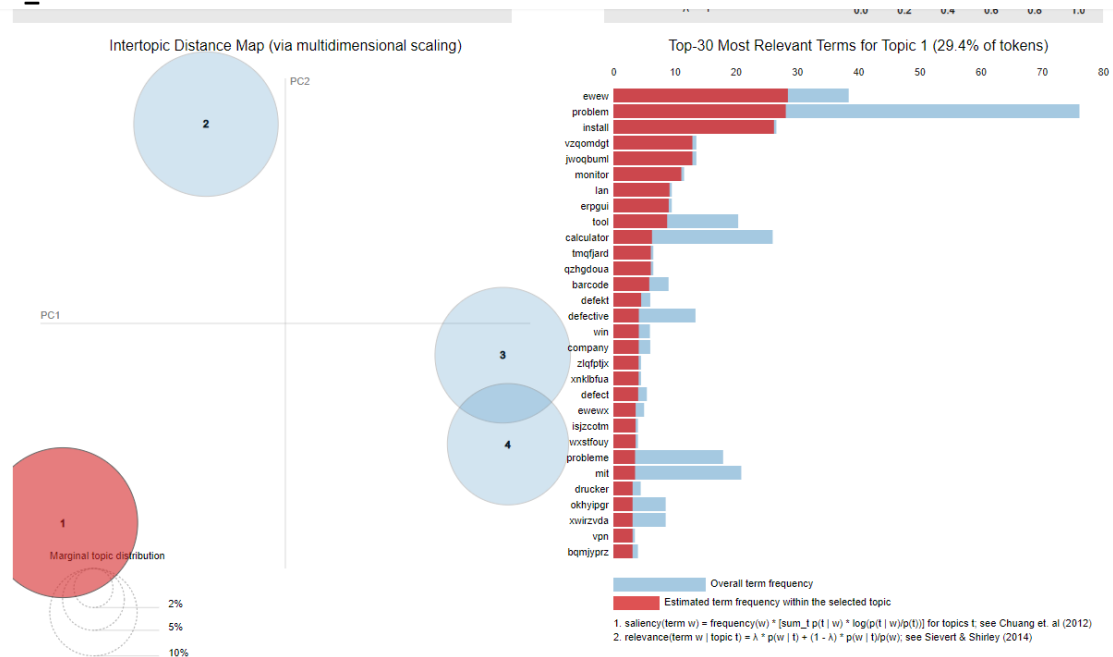
GRP_8



Top 4 topics in GRP_8

```
(0, '0.032*"unable" + 0.027*"account" + 0.023*"user" +
0.021*"outlook" + 0.021*"issue" + 0.019*"login" + 0.019*"email" +
0.015*"locked" + 0.013*"skype" + 0.012*"access"')
(1, '0.074*"password" + 0.058*"reset" + 0.052*"erp" + 0.035*"sid"
+ 0.019*"please" + 0.019*"com" + 0.017*"gmail" + 0.017*"account" +
0.013*"unlock" + 0.012*"printer"')
(2, '0.038*"com" + 0.032*"gmail" + 0.028*"ticket" + 0.024*"update"
+ 0.021*"received" + 0.017*"please" + 0.012*"erp" + 0.012*"company"
+ 0.011*"vpn" + 0.011*"inplant"')
(3, '0.058*"tool" + 0.056*"password" + 0.024*"unable" +
0.020*"engineering" + 0.016*"com" + 0.014*"management" +
0.014*"help" + 0.014*"gmail" + 0.014*"login" + 0.012*"received"')
```

● GRP_24



```
(0, '0.041*"new" + 0.040*"setup" + 0.039*"problem" +
0.025*"jionmpsf" + 0.025*"wnkpzcmv" + 0.017*"mit" + 0.017*"tool" +
0.015*"reinstall" + 0.013*"probleme" + 0.011*"ljtzbdqg"')
(1, '0.067*"setup" + 0.053*"new" + 0.030*"calculator" +
0.023*"wzrgyung" + 0.023*"wrcktgbd" + 0.014*"problem" +
0.013*"ewew" + 0.012*"work" + 0.010*"wnkpzcmv" + 0.010*"jionmpsf"')
(2, '0.044*"problem" + 0.020*"csenjruz" + 0.019*"niptbwdq" +
0.017*"mit" + 0.015*"probleme" + 0.015*"printer" + 0.014*"access" +
0.014*"calculator" + 0.012*"support" + 0.010*"portal"')
(3, '0.049*"ewew" + 0.049*"problem" + 0.045*"install" +
0.022*"vzqomdgt" + 0.022*"jwoqbuml" + 0.019*"monitor" + 0.016*"lan"
+ 0.016*"erpgui" + 0.015*"tool" + 0.011*"calculator"')
```

3.6. Word Vectors with tSNE

t-distributed stochastic neighbor embedding (t-SNE) is a statistical method for visualizing high-dimensional data by giving each datapoint a location in a two or three-dimensional map.

Word2Vec focuses on predicting words via their linguistic context, thereby the model identifies underlying similarities among groups of words which are unlikely to all appear in the same text.



t-SNE model has picked up some interesting word clusters.

- There is a cluster of words that appears to indicate countries and places. Words in this cluster include: usa, germany, india, amsterdam etc.

- A second cluster of words appears to indicate machines, and includes words such as: laptop, wifi, homecomputer, monitor, wireless, machine etc.
- A few other clusters of words. Group of words appear to indicate issue: password reset management, drive folder access etc.

- **Top 10 Similar/related words**

```
# Top 10 Similar/related words from Word2Vec for Login
('caller', 0.91),
('confirmed', 0.87),
('resolved', 0.86),
('hub', 0.84),
('es', 0.84),
('logging', 0.83),
('teamviewer', 0.81),
('advised', 0.8),
('sync', 0.79),
('able', 0.79)]

# Top 10 Similar/related words from Word2Vec for laptop
('dell', 0.81),
('computer', 0.8),
('office', 0.78),
('old', 0.75),
('wifi', 0.74),
('setup', 0.73),
('room', 0.72),
('machine', 0.71),
('work', 0.7),
('setting', 0.69)
```

4. Text pre-processing:

4.1. Data cleansing / Data inconsistencies

Data had Newline characters, carriage return characters and accented characters.

- Newline characters and carriage returns are removed using regular expression.
- Accented characters are removed using `unicodedata.normalize` libraries

4.2. Impute missing data

Dataset had missing data – 8 for short description and 1 for Description.

- Wherever description was missing, update with short description if available and vice versa.
- If both are null, we have dropped the rows as the count is very less for this condition, also text is required for analysis, imputing the data will bias / pollute the data

4.3. Data Pre-processing:

Following data cleaning process was done

- Translation
- Remove unwanted characters
- Convert to lower case
- Removal of stop words
- Lemmatization

5. Word Vocabulary / Word Embedding:

Corpus / Word Vocabulary - All the texts / words used in the data is stored in a list of lists known as Corpus. In other words, we will have one giant list which contains all the texts. Within this giant list, each individual text will be represented in a (sub) list, which contains the words for that text. Word embedding will use this Corpus

Word Embedding or Word Vector is a numeric vector input that represents a word in a lower-dimensional space. The reason for this transformation is so that machine learning algorithms can perform linear algebra operations on numbers (in vectors) instead of words. Below Techniques are used in our project.

- **BOW**

A bag-of-words model, or BoW for short, is a way of extracting features from text for use in modelling, such as with machine learning algorithms. The approach is very simple and flexible, and can be used in a many of ways for extracting features from documents

In BoW we construct a dictionary that contains a set of all unique words from our text review dataset. The frequency of the word is counted here. if there are **d** unique words in our dictionary then for every sentence or review the vector will be of length **d** and count of word from review is stored at its particular location in vector. The vector will be highly sparse in such cases. In binary BoW, we don't count the frequency of word, we just place **1** if the word appears in the review or else **0**. In CountVectorizer there is a parameter **binary = true** this makes our BoW to binary BoW

- **TF-IDF**

TF-IDF (term frequency-inverse document frequency) is a statistical measure that evaluates how relevant a word is to a document in a collection of documents.

This is done by multiplying two metrics: how many times a word appears in a document, and the inverse document frequency of the word across a set of documents.

- **Word2Vec**

Word embedding is one of the most popular representations of document vocabulary. It is capable of capturing context of a word in a document, semantic and syntactic similarity, relation with other words, etc. In Word2Vec every word is assigned a vector.

Word2Vec is one of the most popular techniques to learn word embeddings using shallow neural network

- **GloVe**

GloVe is an unsupervised learning algorithm for obtaining vector representations for words. In this method, we take the corpus and iterate through it and get the co-occurrence of each word with other words in the corpus. We get a co-occurrence matrix through this. The words which occur next to each other get a value of 1, if they are one word apart then $1/2$, if two words apart then $1/3$ and so on.

- **ULMFit**

Universal Language Model Fine-Tuning (ULMFIT) is a transfer learning technique which can help in various NLP tasks.

ULMFIT creates its own word embeddings. Rather, the difference between Word2Vec and Glove, on one side, and ULMFIT, on the other is how such word embeddings are built. While training the pre-trained language model (meaning the first stage) ULMFIT tries to predict the next character, whereas Word2Vec tries to predict the word in the middle of each sentence. The idea behind it being to understand the context of other words/tokens a word is surrounded by.

6. Feature Engineering

- Both Short Description and Description has similar words, but there are bunch of words in addition in both fields, it would be good idea to use both features for ticket classification and assignment, hence created a new field "Desc_All" by combining "Short description" and "Description" field and model trained with new field performed better than with either one of the features used.
- We also tried including the Caller field as part of this new field, but it didn't improve model performance, hence the Caller feature was ignored.

7. Model Building

7.1. Base model:

Following base model were considered for benchmark

1. Logistic regression,
2. SVM,
3. Naïve Bayes,
4. Random Forest.

Since it is a multi-classification problem, One-vs-the-rest (OvR) multiclass strategy was used. For each classifier, the class is fitted against all the other classes. In addition to its computational efficiency (only $n_classes$ classifiers are needed), one advantage of this approach is its interpretability. Since each class is represented by one and one classifier only, it is possible to gain knowledge about the class by inspecting its corresponding classifier. This is the most commonly used strategy for multiclass classification and is a fair default choice.

Based model were tried for different combination of features like

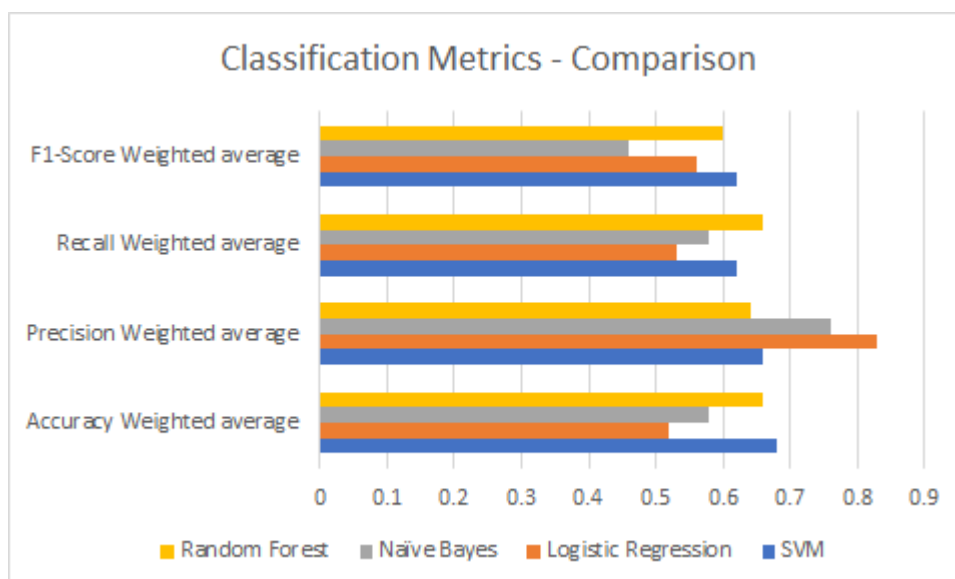
- Classification based on Description - Prior to Data Cleansing
- Classification based on Description - Post Data Cleansing
- Classification based on Short description
- Classification based on both Short description and Description [Feature Engineering, created new field by combining these 2 features]

Note: Also, Data features extracted were tried on TF-IDF and BOW vectorizers

Results from various models are given below.

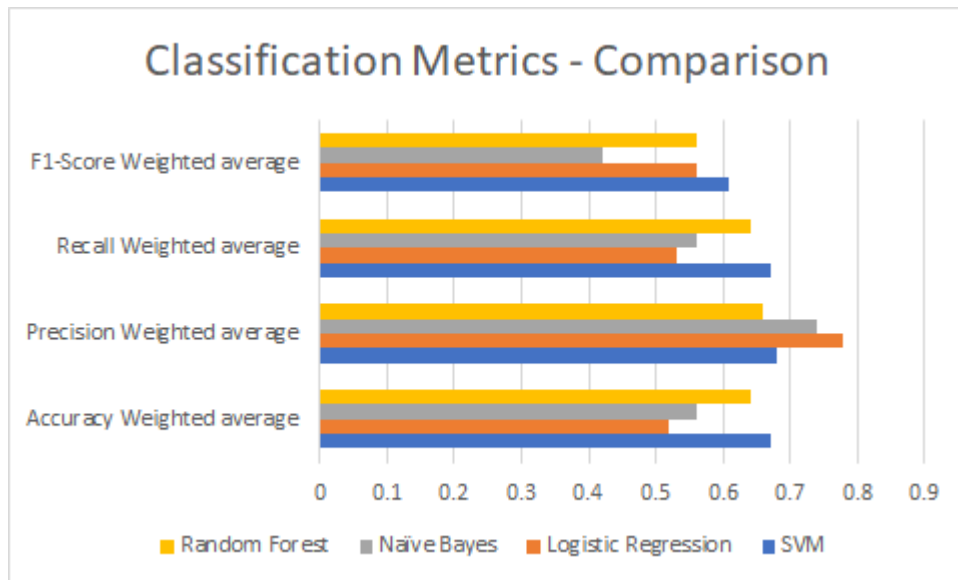
Model trained on Short Description

Classification metrics for models trained on Short Description:



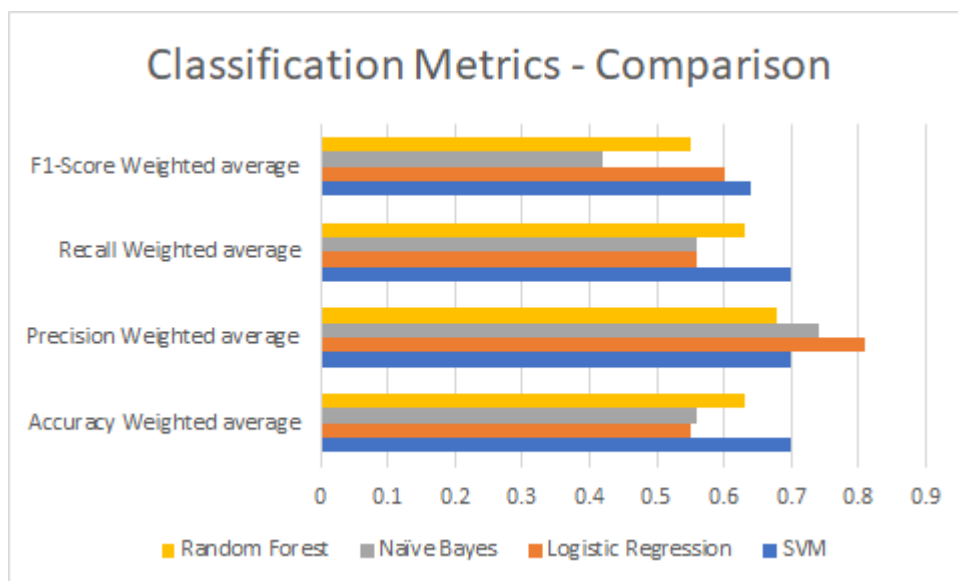
Model trained on Long Description

Classification metrics for models trained on Long Description:



Model trained on combining Short Description and Long Description

Classification metrics for models trained by combining both Long Description and Short Description



- TF-IDF seem to work better than BoW vectorizer,
- While using TF-IDF vectorizer, SVM and logistic regression model performed better among the above models.

7.2. Hyper Parameter Tuning using GridSearchCV

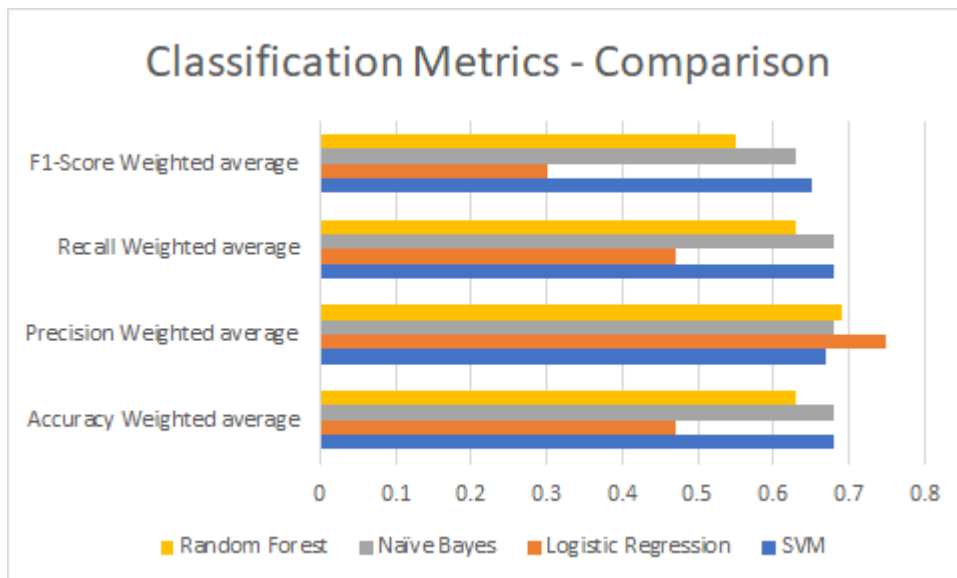
Next step, we tried Hyper parameter tuning for various base models, GridSearchCV was used for parameter tuning.

GridSearchCV is a meta-estimator. It takes an estimator like SVC and creates a new estimator that behaves exactly the same – in this case, like a classifier. You should add `refit=True` and choose `verbose` to whatever number you want, the higher the number, the more verbose (verbose just means the text output describing the process).

What `fit` does is a bit more involved than usual. First, it runs the same loop with cross-validation, to find the best parameter combination. Once it has the best combination, it runs `fit` again on all data passed to `fit` (without cross-validation), to build a single new model using the best parameter setting.

You can inspect the best parameters found by GridSearchCV in the `best_params_` attribute, and the best estimator in the `best_estimator_` attribute

Results for hyper parameter tuning for various models given in the below plot and table.



Below table shows best parameters & Metrics as part of hyper parameter tuning for each of the models

Model Name	Accuracy Weighted average	Precision Weighted average	Recall Weighted average	F1-Score Weighted average	Trials
SVM	0.68	0.67	0.68	0.65	Embedding - TF-IDF Vectorizer Hyper Parameter Tuning - Best Model {'clf_svm__estimator__C': 1000, 'clf_svm__estimator__gamma': 0.001, 'clf_svm__estimator__kernel': 'rbf', 'vectorizer__lowercase': True,

					'vectorizer__ngram_range': (1, 1), 'vectorizer__strip_accents': 'unicode'}
Logistic Regression	0.67	0.65	0.67	0.63	Embedding : BOW Vectorizer HYPER PARAMETER Tuning - Best Model {'clf_LR__estimator__C': 10, 'clf_LR__estimator__max_iter': 250, 'clf_LR__estimator__penalty': 'l2', 'clf_LR__estimator__solver': 'liblinear', 'vectorizer__lowercase': True, 'vectorizer__ngram_range': (1, 1), 'vectorizer__strip_accents': 'unicode'}
Logistic Regression	0.68	0.67	0.68	0.64	Embedding: TF-IDF Vectorizer Hyper Parameter Tuning - Best Model {'clf_LR__estimator__C': 100, 'clf_LR__estimator__max_iter': 250, 'clf_LR__estimator__penalty': 'l2', 'clf_LR__estimator__solver': 'newton-cg', 'vectorizer__lowercase': True, 'vectorizer__ngram_range': (1, 1), 'vectorizer__strip_accents': 'unicode'}
Naïve Bayes	0.68	0.68	0.68	0.63	Embedding: TF-IDF Vectorizer Hyper Parameter Tuning - Best Model {'clf_MNB__estimator__alpha': 0.01, 'vectorizer__lowercase': True, 'vectorizer__ngram_range': (1, 2), 'vectorizer__strip_accents': 'unicode'}
Random Forest	0.63	0.69	0.63	0.55	Embedding: TF-IDF Vectorizer Hyper Parameter Tuning - Best Model {'clf_RF__bootstrap': False, 'clf_RF__max_depth': None, 'clf_RF__max_features': 'auto', 'clf_RF__min_samples_leaf': 1, 'clf_RF__min_samples_split': 5, 'clf_RF__n_estimators': 1000, 'vectorizer__lowercase': True, 'vectorizer__ngram_range': (1, 1), 'vectorizer__strip_accents': 'unicode'}

Post Hyper Parameter tuning, SVM continued to be performing better than the rest of the models.

Word2Vec

- We tried the Word2Vec model for Logistic regression, but we could not find any embeddings trained, and the scores were below the benchmark models.

7.3. Deep Learning Models –Basic NN & LSTM

For model improvement as next step we tried for Neural Network & Deep learning LSTM(RNN) models

7.3.1. Neural Network Model

Neural Network model was built with following details

Vectorizer : Before training, we need to map strings to a numerical representation , using a tokenizer. Used Keras tokenizer , convert the texts into sequences of indices representing the 20000 most frequent words. Create two lookup tables: one mapping words to numbers, and another for numbers to words. Sequences have different lengths, so we pad them (add 0 vectors at the end until the sequence is of length 100 words)

Embedding Layer : input layer. A trainable lookup table that will map the numbers of each character to a vector with `embedding_dim` dimensions;

Neural Network model details :

- **Number of Dense layers** : 5
- **Activation Function** : relu, and sigmoid for the last dense layer
- **Dropout**: 0.3
- **Optimizer**: Adam
- **Loss**: Binary Cross Entropy
- **Metrics**: Accuracy
- **Batch size**=128
- **Epochs**=20
- **Callback** – with Early stopping , monitor was `val_accuracy` and patience = 5 and delta = 0.01

This model didn't had improvement in the metrics and it had `f1_weighted` score of 0.55

7.3.2. LSTM(RNN) with GloVe Embeddings

Tried with deep learning model LSTM using GloVe Embeddings

Vectorizer : Before training, we need to map strings to a numerical representation , using a tokenizer. Used Keras tokenizer , convert the texts into sequences of indices representing the 20000 most frequent words. Create two lookup tables: one mapping words to numbers, and another for numbers to words. Sequences have different lengths, so we pad them (add 0 vectors at the end until the sequence is of length 100 words)

Glove Embedding:

Pre-trained word vectors. For our project GloVe database is taken from Wikipedia 2014 + Gigaword 5 (6B tokens, 400K vocab, uncased, 50): `glove.6B.zip` [DataSet - `glove.6B.50d.txt`]

Read all the words from the `glove2vec` database and create an index of words and corresponding word vector embedding. Create corresponding word vector embedding for our corpus and pass it as Weight input for Embedding Layer

Embedding Layer: Input layer. A trainable lookup table that will map the numbers of each character to a vector with `embedding_dim` dimensions;

LSTM RNN model details:

Use `keras.Sequential` to define the model. Three layers are used to define our model:

- `keras.layers.Embedding`: The input layer. A trainable lookup table that will map the numbers of each character to a vector with `embedding_dim` dimensions;
- `keras.layers.Bidirectional LSTM`: A type of RNN with size `units=rnn_units`

- `keras.layers.Dense`: The output layer, with number of classes as outputs.
- Optimizer: Adam
- Loss: `categorical_crossentropy`
- Metrics: Accuracy
- Batch size=128
 - Epochs=20
 - Callback – with Early stopping, monitor was `val_accuracy` and `patience = 2` and `delta = 0.01`

This model didn't have a better `f1_weighted` score of 0.62, but still less than the SVM model.

7.4. Transfer learning using ULMFit

We can use pre-trained state-of-the-art deep learning models and tweak them to work for our classification use case. This is known as transfer learning. It is not as resource intensive as training a deep learning model from scratch and produces decent results even on small amounts of training data.

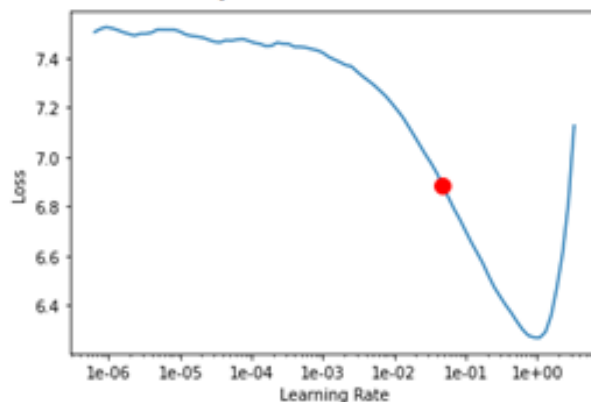
As a final step, we tried to use Transfer Learning based on ULMFit. Below was the approach

- Pre-Trained Model: AWD_LSTM
- Number of Classes: All Classes

AWD-LSTM (ASGD Weight-Dropped LSTM) is one of the most popular language models. It uses DropConnect and the average random gradient descent method, and several other regularization strategies.

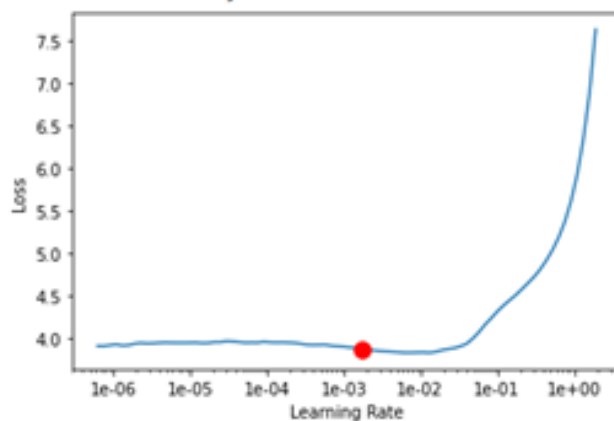
- Transfer Learning Approach (ULMFIT)
- Pre-process the data
- Split the data in test and train datasets
- Invoke the `TextLMDataBunch` to call the method `from_df` to create a language model data bunch
- Train the language model using the method `fit_one_cycle()`
 - Below plot shows the change in loss function (cross entropy) for different learning rates.

LR Finder is complete, type {learner_name}.recorder.plot() to see the graph.
 Min numerical gradient: 4.79E-02
 Min loss divided by 10: 1.10E-01



- Tried to tune the drop_mult parameter value. Started off with the value 0.3. While we increased the value to 0.5 and 0.7, there was a dip in the performance, possibly to regularization. Hence retained the value 0.3 for the final model.
- As we are interested in classification of the tickets, we can save just the encoder part alone.
- Create a text classifier bunch followed by the encoder saved in the previous step.
- Train the model by invoking the method fit_one_cycle().
 - Below plot shows the change in loss function (cross entropy) for different learning rates.

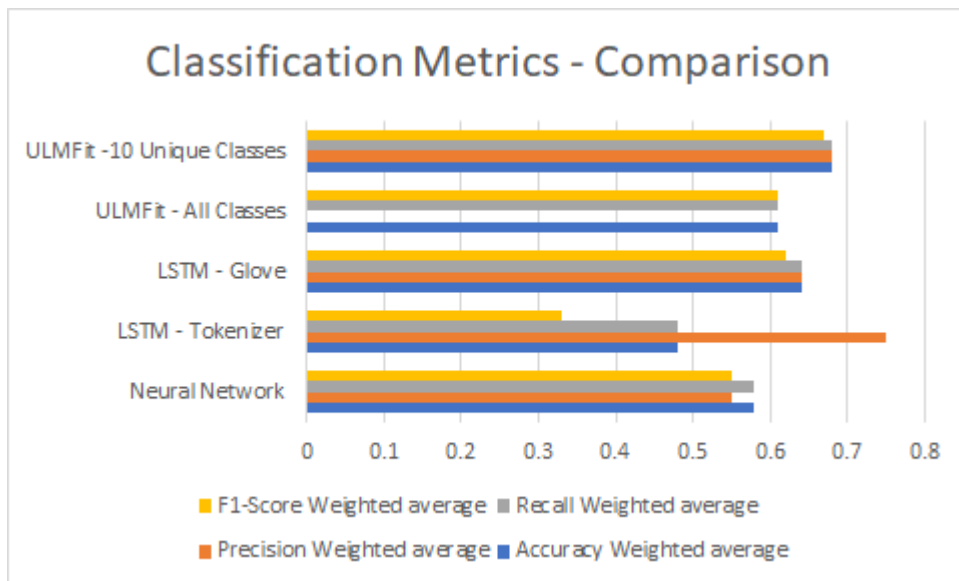
LR Finder is complete, type {learner_name}.recorder.plot() to see the graph.
 Min numerical gradient: 1.74E-03
 Min loss divided by 10: 1.32E-03



- Once trained, save the model for future predictions

As we did not notice a big improvement in the classification metrics, we tried ULMFIT once again, but this time we grouped the classes into 10 unique classes alone. The approach was to see the behaviour of the model with a lesser number of unique classes. Ideally, we should be able to see an improvement, given the fact that we do not have sufficient data for all the classes.

Overall performance, for the Neural Network, Deep learning and Transfer learning model is given in the below chart.



8. Model evaluation

Since it is a multi-class classifier problem, Weighted F1 Score was used to select the best model.

SVM Classifier after hyper parameter tuning turned out to be the best model.

Model details:

TF-IDF Vectorizer

Features: Short Description and Description

Prominent Parameters - Best Model {'clf_svm__estimator__C': 1000, 'clf_svm__estimator__gamma': 0.001, 'clf_svm__estimator__kernel': 'rbf', 'vectorizer__lowercase': True, 'vectorizer__ngram_range': (1, 1), 'vectorizer__strip_accents': 'unicode'}

Evaluation Metrics is based on **Classification Metrics**.

1. **Precision** - What percent of Model predictions were correct - Accuracy of positive predictions.
Precision = $TP / (TP + FP)$
2. **Recall** - What percent of the positive cases did Model predict; $TP / (TP + FN)$
3. **f1-score** gives you the harmonic mean of precision and recall. $(2 * Precision * Recall) / (Precision + Recall)$
4. **Accuracy** - Number of correct predictions to total number of predictions; $TP + TN / (TP + TN + FP + FN)$
5. **Support** Support is the number of actual occurrences of the class in the specified dataset

- **True Positive (TP)** — A true positive is an outcome where the model correctly predicts the positive class.
- **True Negative (TN)** — A true negative is an outcome where the model correctly predicts the negative class.
- **False Positive (FP)** — A false positive is an outcome where the model incorrectly predicts the positive class.
- **False Negative (FN)** — A false negative is an outcome where the model incorrectly predicts the negative class

One drawback is that both precision and recall are given equal importance due to which according to our application we may need one higher than the other and F1 score may not be the exact metric for it. Therefore, either weighted-F1 score or PR or ROC curve can be used.

In weighted-average F1-score, or weighted-F1, we weight the F1-score of each class by the number of samples from that class

Similarly, we can compute weighted precision and weighted recall:

Metrics for best model is given below

```
Accuracy score: 0.68
precision_weighted: 0.67
recall_weighted: 0.68
f1_weighted: 0.64
```

Classification Report:

	precision	recall	f1-score	support
0	0.73	0.93	0.82	994
1	0.25	0.25	0.25	8
2	0.54	0.40	0.46	35
3	0.67	0.25	0.36	8
4	0.64	0.61	0.62	64
5	0.55	0.58	0.57	36
6	0.53	0.33	0.41	30
7	0.33	0.10	0.15	10
8	0.60	0.14	0.23	21
9	0.95	0.95	0.95	20
10	0.55	0.50	0.52	22
11	0.43	0.37	0.40	54
12	0.51	0.47	0.49	60
13	0.50	0.11	0.18	9
14	1.00	0.14	0.25	7
15	0.33	0.12	0.18	8
16	0.60	0.50	0.55	6
17	0.89	0.76	0.82	72
18	0.77	0.34	0.48	29
19	0.14	0.07	0.10	14
20	1.00	0.00	0.00	4
21	0.50	0.09	0.15	11
22	0.59	0.42	0.49	24
23	0.40	0.28	0.33	50
24	0.50	0.17	0.25	6
25	0.38	0.35	0.36	17
26	0.67	0.37	0.48	27
27	0.67	0.25	0.36	16
28	1.00	0.25	0.40	4
29	1.00	0.25	0.40	4
30	1.00	0.60	0.75	5

31	0.47	0.32	0.38	25
32	0.40	0.18	0.25	11
33	0.86	0.60	0.71	10
34	0.80	0.44	0.57	9
35	1.00	0.25	0.40	4
36	0.00	0.00	0.00	9
37	0.67	0.29	0.40	7
38	1.00	0.00	0.00	2
39	0.65	0.53	0.59	32
40	0.00	0.00	0.00	3
41	1.00	0.00	0.00	3
42	0.77	0.37	0.50	46
43	1.00	0.20	0.33	5
44	1.00	0.17	0.29	6
45	1.00	0.00	0.00	3
46	0.45	0.29	0.36	17
47	0.58	0.84	0.69	165
48	0.73	0.17	0.28	63
49	0.45	0.23	0.30	22
accuracy			0.68	2117
macro avg			0.64	2117
weighted avg			0.67	2117

9. Benchmark Comparison

As a part of interim submission, we set the benchmark at Weighted F1 Score of 0.64 (best Model using SVM).

Model Name	Accuracy Weighted average	Precision Weighted average	Recall Weighted average	F1-Score Weighted average	Trials
SVM	0.7	0.7	0.7	0.64	TF-IDF Vectorizer New feature created using Short Description and Long Description Default hyper parameter

Tried with hyper parameter tuning, Deep learning models, Transfer learning using ULMFit, table below shows score for these trials.

Model Name	Accuracy Weighted average	Precision Weighted average	Recall Weighted average	F1-Score Weighted average	Trials
SVM	0.68	0.67	0.68	0.65	TF-IDF Vectorizer Hyper Parameter Tuning - Best Model {'clf_svm__estimator__C': 1000, 'clf_svm__estimator__gamma': 0.001, 'clf_svm__estimator__kernel': 'rbf', 'vectorizer__lowercase': True, 'vectorizer__ngram_range': (1, 1), 'vectorizer__strip_accents': 'unicode'}
Neural Network	0.58	0.55	0.58	0.55	Embedding -Tokenizer Classification based on both short description and Description;
LSTM	0.64	0.64	0.64	0.62	GLoVe Embedding Classification based on both short description and Description;
ULMFit - With All Classes	0.61	0.61	0.61	0.61	ULMFit - With All Classes. Pre-Trained Model: AWD_LSTM drop_mult = 0.3 Model: https://s3.amazonaws.com/fast-ai-modelzoo/wt103-fwd.tgz Learning Rate: 2e-03 Optimizer: Adam Loss Function: Cross Entropy
Transfer Learning AWD LSTM - ULMFit	0.68	0.68	0.68	0.67	Grouped to 10 unique classes Pre-Trained Model: AWD_LSTM drop_mult = 0.3 Model: https://s3.amazonaws.com/fast-ai-modelzoo/wt103-fwd.tgz Learning Rate: 2e-03 Optimizer: Adam Loss Function: Cross Entropy

Hyper parameter and ULMFit models had marginal improvements in the weighted F1 Scores, but deep learning model & LSTM didn't had improvement due to following reasons

- Dataset volume
- Class imbalance
- Too many classes

10. UI App Model deployment

Followed below high-level steps for deploying best performing SVM model in Streamlit Cloud

Step 1: Create a new virtual environment using IDE like Spyder, Pycharm

Step 2: Install necessary libraries.

Step 3: Build the best machine learning model and save it. – In our project SVM classifier performed well and model is saved using pickle package

Step 4: Test the loaded model.

Step 5: Create an App using Web application frameworks like Streamlit, Flask, Django. We have chosen Streamlit for our model deployment

Step 6: Create Main.py file which does following

- Loads serialized pickle model (best model: SVM).
- Create GUI which receives inputs from user
- Performs Pre-processing and uses the trained model to make predictions and predicted results are displayed to the user.

Step 7: Upload local project to GitHub

Step 8: Create a New App in Streamlit, by providing GitHub URL

App is hosted on Streamlit Cloud and can be accessed using below URL

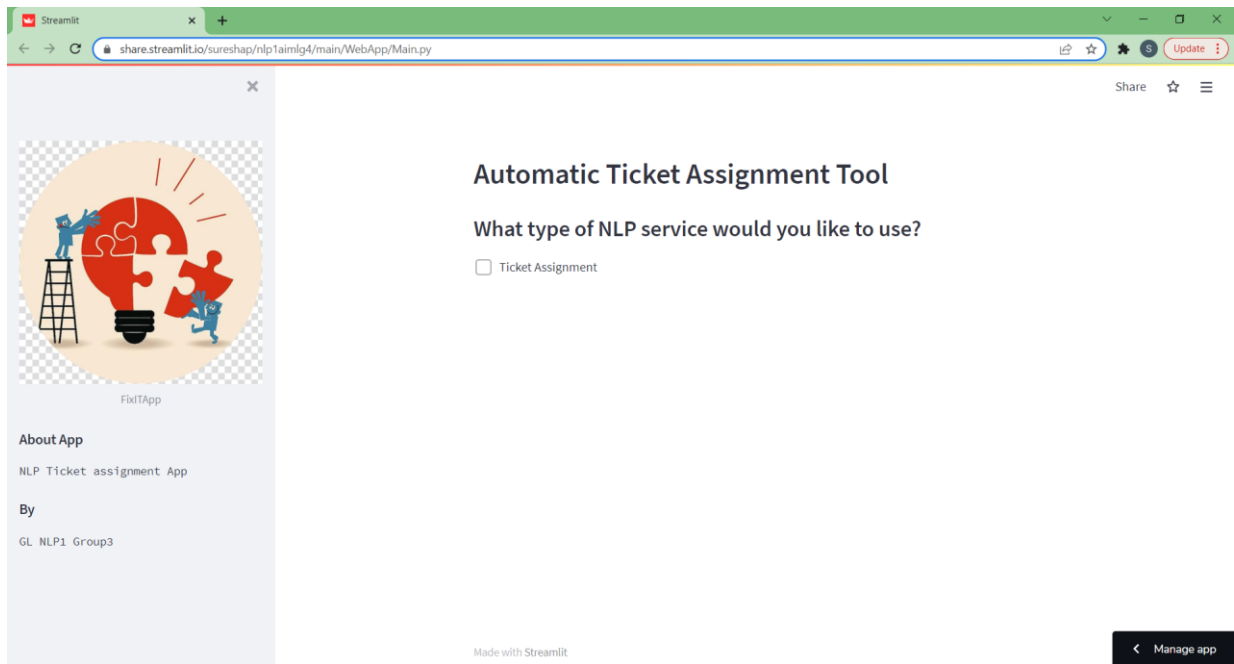
<https://share.streamlit.io/sureshap/nlp1aimlg4/main/WebApp/Main.py>

GitHub URL for source code:

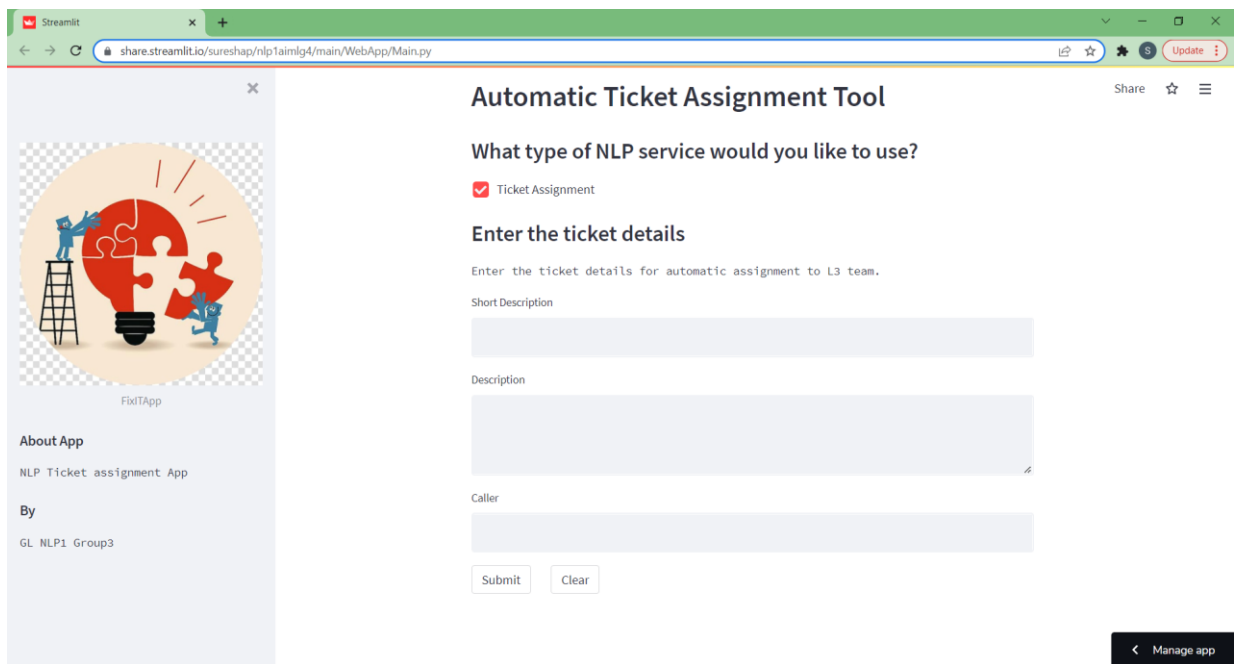
<https://github.com/sureshap/nlp1aimlg4/tree/main/WebApp>

Below are snapshots of UI App deployed in Streamlit Cloud

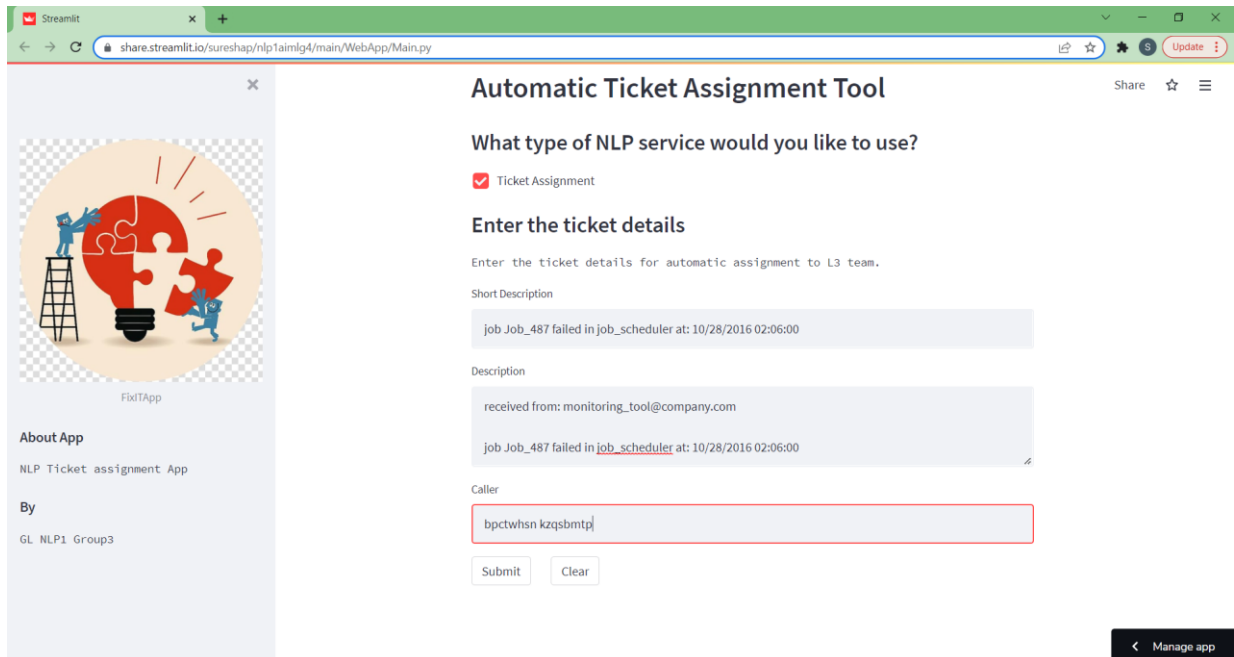
1. Main screen



2. Select Ticket Assignment option



3. Enter ticket details



The screenshot shows a web application titled "Automatic Ticket Assignment Tool" running on Streamlit. The interface includes a sidebar with an app icon (a lightbulb with puzzle pieces) and the text "FixITApp", "About App", "NLP Ticket assignment App", and "By GL NLP1 Group3". The main content area has a header "Automatic Ticket Assignment Tool" and a sub-header "What type of NLP service would you like to use?". Below this is a checkbox labeled "Ticket Assignment" which is checked. The section "Enter the ticket details" prompts the user to "Enter the ticket details for automatic assignment to L3 team.". It contains three text input fields: "Short Description" with the value "job Job_487 failed in job_scheduler at: 10/28/2016 02:06:00", "Description" with the value "received from: monitoring_tool@company.com" and "job Job_487 failed in job_scheduler at: 10/28/2016 02:06:00", and "Caller" with the value "bpctwhsn kzqsbmtpl". There are "Submit" and "Clear" buttons at the bottom of the form. A "Manage app" button is visible in the bottom right corner.

Automatic Ticket Assignment Tool

What type of NLP service would you like to use?

☒ Ticket Assignment

Enter the ticket details

Enter the ticket details for automatic assignment to L3 team.

Short Description

job Job_487 failed in job_scheduler at: 10/28/2016 02:06:00

Description

received from: monitoring_tool@company.com

job Job_487 failed in job_scheduler at: 10/28/2016 02:06:00

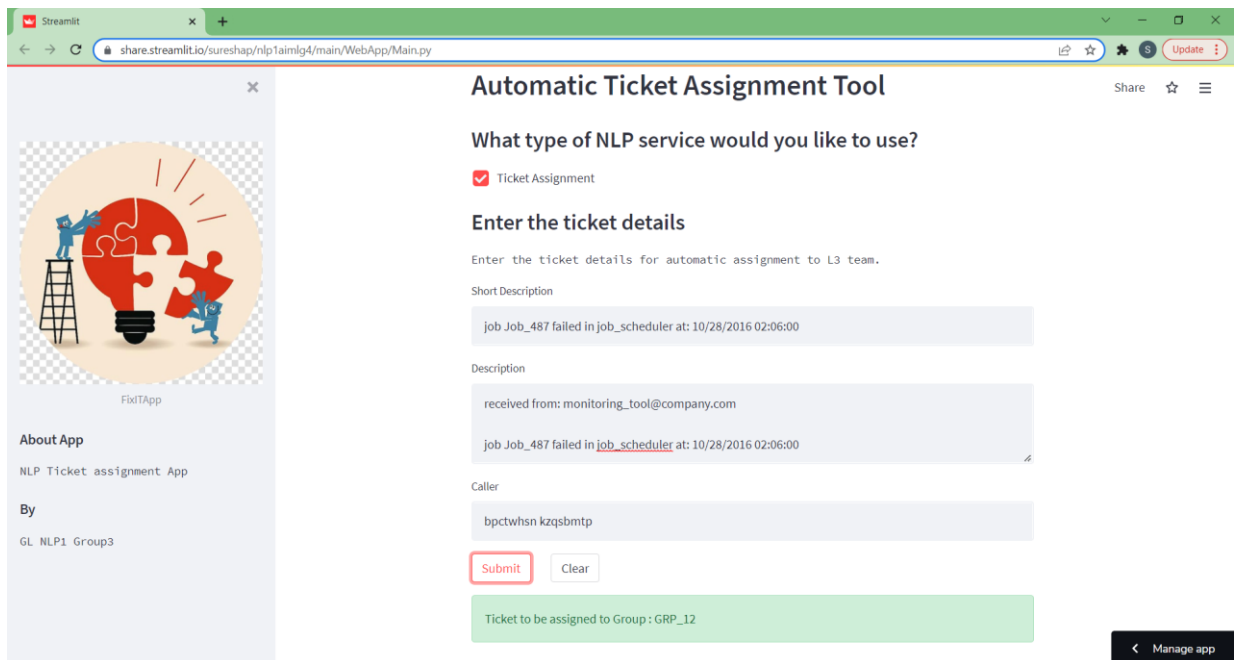
Caller

bpctwhsn kzqsbmtpl

Submit Clear

Manage app

4. Model predicts and display output (assignment group) in screen



The screenshot shows the same web application as before, but now with the output of the model prediction displayed. The "Submit" button is highlighted in red. Below the form, a green box displays the text "Ticket to be assigned to Group : GRP_12". The "Manage app" button is still visible in the bottom right corner.

Automatic Ticket Assignment Tool

What type of NLP service would you like to use?

☒ Ticket Assignment

Enter the ticket details

Enter the ticket details for automatic assignment to L3 team.

Short Description

job Job_487 failed in job_scheduler at: 10/28/2016 02:06:00

Description

received from: monitoring_tool@company.com

job Job_487 failed in job_scheduler at: 10/28/2016 02:06:00

Caller

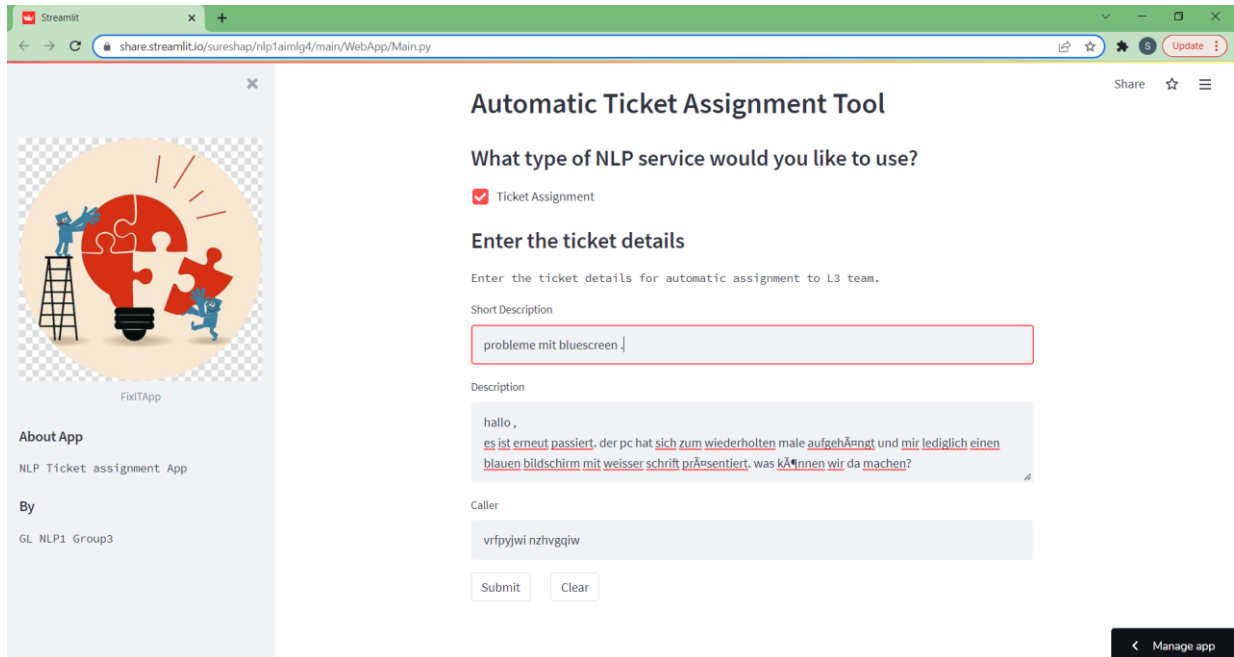
bpctwhsn kzqsbmtpl

Submit Clear

Ticket to be assigned to Group : GRP_12

Manage app

5. Try entering new ticket details in Non-English language (German)



The screenshot shows a web application titled "Automatic Ticket Assignment Tool". On the left is a sidebar with a logo of two figures working on a lightbulb puzzle, labeled "FixITApp". The sidebar also contains "About App" (NLP Ticket assignment App) and "By" (GL NLP1 Group3). The main content area has a heading "Automatic Ticket Assignment Tool" and a sub-heading "What type of NLP service would you like to use?". A checkbox labeled "Ticket Assignment" is checked. Below this is a section "Enter the ticket details" with a prompt "Enter the ticket details for automatic assignment to L3 team.". There are two input fields: "Short Description" containing "probleme mit bluescreen ." and "Description" containing a German text about a PC screen issue. A "Caller" field contains "vrfpyjwi nzhvgqiw". At the bottom of the form are "Submit" and "Clear" buttons. A "Manage app" button is in the bottom right corner.

Automatic Ticket Assignment Tool

What type of NLP service would you like to use?

☒ Ticket Assignment

Enter the ticket details

Enter the ticket details for automatic assignment to L3 team.

Short Description

probleme mit bluescreen .

Description

hallo ,
es ist erneut passiert, der pc hat sich zum wiederholten male aufgehängt und mir lediglich einen blauen bildschirm mit weisser schrift präsentiert. was können wir da machen?

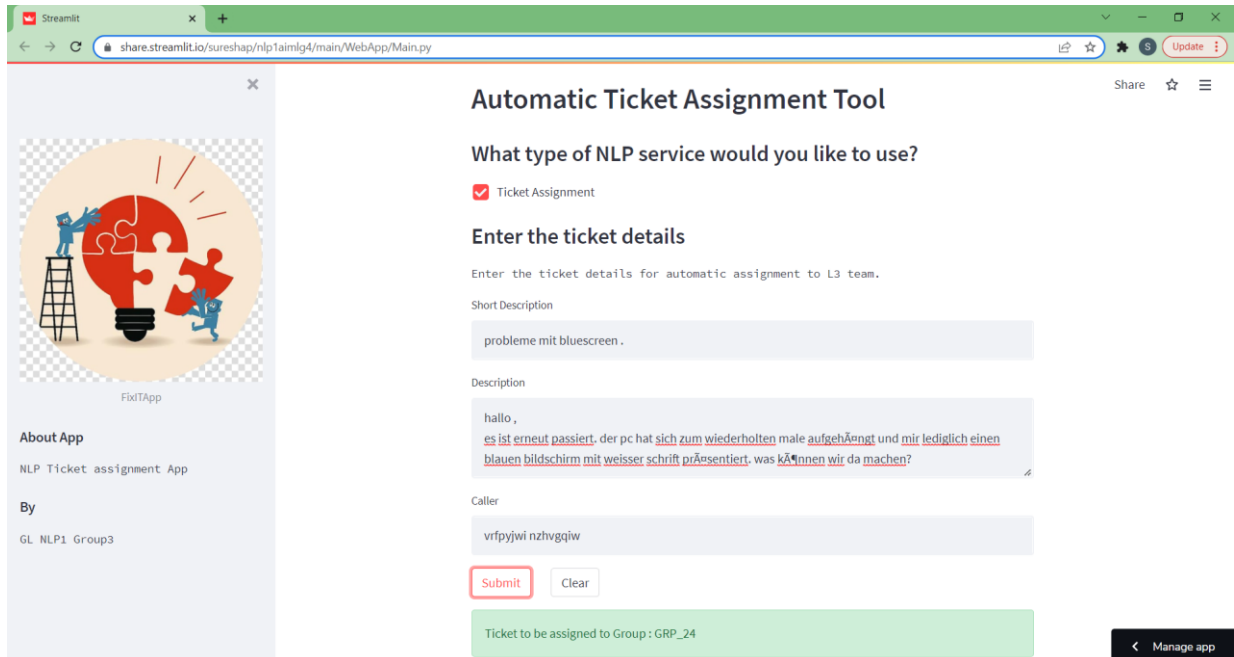
Caller

vrfpyjwi nzhvgqiw

Submit Clear

Manage app

6. Model predicts and display output (assignment group) in screen



This screenshot is identical to the previous one, but with an additional green box at the bottom of the form area. This box contains the text "Ticket to be assigned to Group : GRP_24", indicating the model's prediction based on the input German text.

Automatic Ticket Assignment Tool

What type of NLP service would you like to use?

☒ Ticket Assignment

Enter the ticket details

Enter the ticket details for automatic assignment to L3 team.

Short Description

probleme mit bluescreen .

Description

hallo ,
es ist erneut passiert, der pc hat sich zum wiederholten male aufgehängt und mir lediglich einen blauen bildschirm mit weisser schrift präsentiert. was können wir da machen?

Caller

vrfpyjwi nzhvgqiw

Submit Clear

Ticket to be assigned to Group : GRP_24

Manage app

11. Implications

Using NLP based AIML we could build a classifier that can automatically classify tickets (ITSM) to right functional groups with an accuracy of 68%

This model when placed in production using batch job or RPA BOT, in turn can automate classification process, done currently by L1/L2 team and in turn help organizations to reduce the resolving time of the issue and can focus on more productive tasks.

12. Limitations

1. Translation works currently for Latin script languages, non-Latin scripts like Chinese, Japanese don't work. Need a translator that can handle non-Latin scripts.
2. For data imbalance, we can try data augmentation techniques like paraphrasing using Paraphrasing Tool like QuillBot and create sample data for lesser data in classes
3. Grouping of appropriate assignment groups based on the domain knowledge would help to clean the Target variable and create multiple groups first at L1/L2 and L3 split and then subdivide into individual assignment group within the respective team
4. Our model is trained on very limited dataset, build a mechanism to train the model on the future data on a periodic basis using MLOPs
5. Instead of using generic Glove embedding, could have used pre-trained word embeddings specific to ITSM tickets
6. Implementation of model – currently deployed in Streamlit which can pick one ticket at a time to predict the assignment group. In real time, it could be batch job or RPA bot which can run at a predefined time which pick the tickets assigned to AIML queue and AIML model program runs to predict the ticket assignment to respective assignment group

13. Closing Reflections

1. Class Imbalance
 - 50% of data falls into GRP_0 out of available 74 groups, there were many Classes with only 1 row indicating class imbalance in the dataset.
 - For data imbalance, we can try data augmentation techniques like paraphrasing using Paraphrasing Tool like QuillBot and create sample data for lesser data in classes
2. Feature parsing:
 - ✓ Looked out for Accent characters; carriage return and new line characters as a part of data pre-processing.
 - Data quality in the description field for some tickets were vague, more detailed description for tickets will help.
 - To avoid missing data, make description field mandatory on the source system

3. Translation of multilingual characters

- ✓ We found translation is important step in the pre-processing as this could improve the model performance
- ✓ Tried multiple language translators to choose the most optimal translator

4. Model Performance tracker

- ✓ Tried to traditional machine learning approaches such as SVM, Naive Bayes, Random Forest.
- ✓ Tried the Deep Learning options including LSTM
- ✓ Tried the Transfer learning approach using ULMFit
- ✓ Model performance was tracked in excel sheet
 - We can look at using MLOPs tool like MLflow, Wandb

5. Model Performance confidence

- ✓ Currently evaluation metrics were used to show confidence of how model works
 - Explain NLP Models by using feature engineering tools like LIME & SHAP to help users understand and interact meaningfully with machine learning

14. References

ULMFIT

1. <https://www.fast.ai/>
2. <https://medium.com/technonerds/using-fastais-ulmfit-to-make-a-state-of-the-art-multi-label-text-classifier-bf54e2943e83#:~:text=Overview,of%20the%20WikiText%20103%20corpus.>
3. <https://www.kaggle.com/neoyipeng2018/tweet-classification-using-awd-lstm>

Language Translators

4. <https://pypi.org/project/googletrans/>

Removing Accented Characters

5. <https://stackoverflow.com/questions/517923/what-is-the-best-way-to-remove-accents-normalize-in-a-python-unicode-string>

Streamlit Application

6. <https://docs.streamlit.io/library/get-started/create-an-app>
7. <https://towardsdatascience.com/building-web-applications-with-streamlit-for-nlp-projects-cdc1cf0b38db>

EDA for NLP

8. <https://towardsdatascience.com/nlp-text-data-visualization-5d5c64c86019>
9. <https://neptune.ai/blog/exploratory-data-analysis-natural-language-processing-tools>

t-SNE

10. <https://www.scikit-yb.org/en/latest/api/text/tsne.html>
11. <https://methodmatters.github.io/using-word2vec-to-analyze-word/>

WordCloud

12. <https://www.analyticsvidhya.com/blog/2020/10/word-cloud-or-tag-cloud-in-python/>

Data Augmentation

13. <https://www.kdnuggets.com/2019/04/text-preprocessing-nlp-machine-learning.html#:~:text=Text%20normalization%20is%20the%20process,%E2%80%9D%20to%20just%20%E2%80%9Cstopwords%E2%80%9D.>
14. https://quillbot.com/?utm_source=Google&utm_medium=Search&utm_campaign=Paraphrase_Developing&gclid=Cj0KCQjw0PWRBhDKARIsAPKHFGiuwp4OB_45GmP7MJyufagI6MU58on6zPuvsAcFNOaf1YEY9sTkpAcaAgUaEALw_wcB

Model working demonstration

15. <https://towardsdatascience.com/explain-nlp-models-with-lime-shap-5c5a9f84d59b>

Hyper Parameter Tuning

16. <https://analyticsindiamag.com/guide-to-hyperparameters-tuning-using-gridsearchcv-and-randomizedsearchcv/>

Modelling Pipeline

17. <https://machinelearningmastery.com/modeling-pipeline-optimization-with-scikit-learn/>