

Capstone Project - INTERIM REPORT

AIML Online May19 Group7A

26 April, 2020

Contents

1. **Summary ……………………..…………………3**
2. **Exploratory Data Analysis ………………………………………..5**
   1. Data Exploration ………………………………………..5
   2. Data Cleaning ………………………………………..7
   3. N-gram Analysis ………………………………………..14
3. **Modelling ………………………………………..16**
   1. Traditional Methods ………………………………………..17
      1. Naive Bayes ………………………………………..17
      2. Logistic Regression ………………………………………..17
      3. SVM classifier ………………………………………..18
      4. Random Forest ………………………………………..18
      5. XG Boost ………………………………………..19
      6. KNN ………………………………………..19
   2. Deep learning models ………………………………………..20
      1. Simple NN ………………………………………..20
   3. Hyperparameter tuning ………………………………………..21
      1. Random Forest ………………………………………..21
      2. Deeplearning model ………………………………………..22
4. **Embeddings ………………………………………..24**
   1. Fast Text ………………………………………..24
   2. Word2Vec ………………………………………..25
   3. GloVe ………………………………………..27
   4. BiLSTM ………………………………………..28
   5. Future improvements ………………………………………….28
5. **Next Steps ………………………………………..29**
6. **Summary**

A ticketing system is a customer service tool that helps companies manage their service and support cases. The system or app creates a "ticket" which documents customer requests and interactions over time, making it easier for customer service reps to resolve complicated issues. It is a tool that allows IT support to be organized, focused, efficient. This directly impacts costs and revenues, customer retention, and public brand image.

An incident is something that is unplanned interruption to an IT service or reduction in the quality of an IT service that affects the Users and the Business. The main goal of Incident Management process is to provide a quick fix / workarounds or solutions that resolves the interruption and restores the service to its full capacity to ensure no business impact. It goes through a lifecycle of Identifying and recording the issue to management and resolution of the same. 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.

An incident management system guided by powerful AI techniques that can classify incidents to right functional groups can help organizations to reduce the resolving time of the issue and yield more productive tasks. In this capstone project, the goal is to build a classifier that can classify the tickets by analyzing text.

Understanding and visualization of data

Data Cleaning and feature extraction

Model fitting and hyperparameter tuning

Effect of word embeddings on performance of model

Our approach consisted of 4 broad steps:

1. Exploratory data analysis: Data was explored and various visualizations were created using word cloud to get a sense of terminology and to identify stopwords and special characters that might need cleaning
2. Data Cleaning and feature vector generation: Upon identification of stopwords, punctuations, special characters and missing fields, data was appropriately cleaned and word frequency and TF-IDF was determined to produce feature vector for model fitting.
3. Model fitting and hyperparameter tuning: input vector was fit into traditional models like Naive Bayes, logistic Regression, SVM, Random Forest, XG boost, KNN and simple Neural Network. Random forest and Neural Network gave better accuracy than others.Complex network consisting of Dense, Batch Normalization and Dropout layer was constructed for modelfitting. The ones that performed better than others were further tuned through their hyperparameters.
4. Word embeddings boost to model performance: Different Embeddings namely ELMO, BERT, FastText, Word2Vec and Glove were explored to further enhance the model performance. ELMO and BERT are not content sensitive embeddings. And therefore, our main focus was on Fast Text, Word2Vec and Glove.

A ticket after being raised will go through the flow as shown in flowchart below:

A ticket is raised. Data will consist of Short Description, Description and Caller details

Description is converted to string and is taken up for further processing

If Size of Description is greater than that of Short description?

Description is replaced by Short Description

Description is converted to lower case

Following is removed form Description:

Line breaks, ‘\n’,Emails, Underscore, Punctuation, Additional Whitespaces,

Digits other than with format Job\_<jobnumber>,

Stopwords, Caller names

Extract bigrams and trigrams

Determine TF-IDF for unigrams, bigrams, trigrams

Based on TF-IDF values, transform unigrams, bigrams and trigrams to feature vectors

Pass generated vector through pre-trained model

Load word embeddings

Create embedding matrix

Output

N

Y

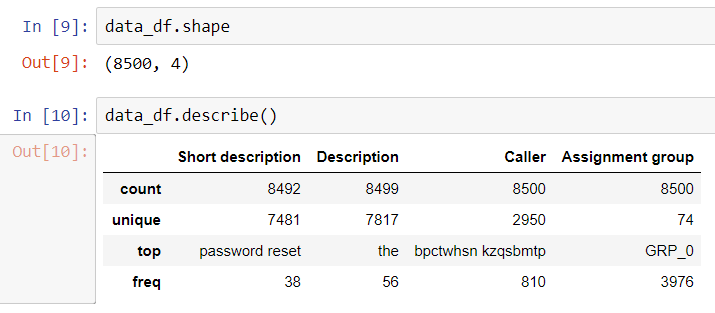
1. **Exploratory Data Analysis**

**2.1 Data Exploration**

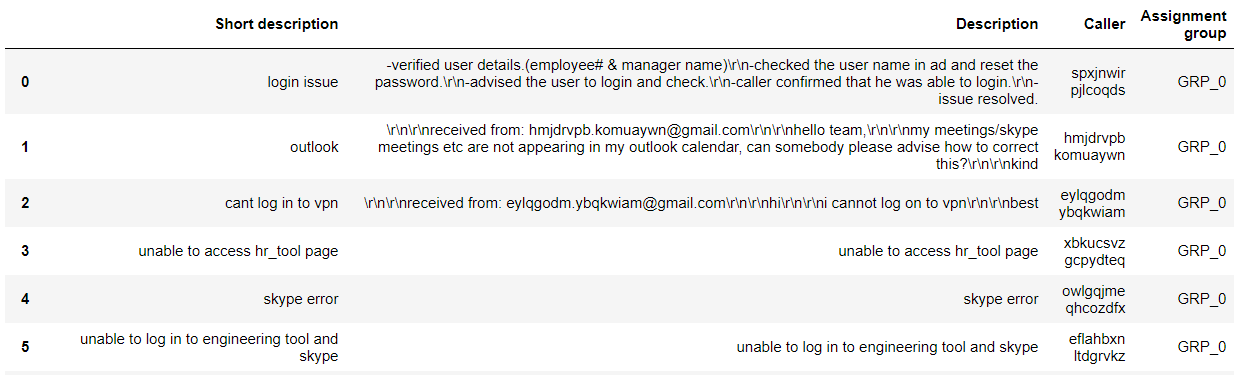
Data for building model for ticketing system consists of 8500 entries with 4 columns. Following is the list these 4 columns.

1. ‘Short description’: Short summary of ticket
2. ‘Description’: Description of the issue faced for which the ticket is raised. Since, this column has more information that ‘Short description’column, this will be our input variable. There are 7817 unique values on the description. This signifies there might be some duplicates in the column. There are cases where length of Description is smaller than that of ‘Short Description’column. In such cases, Short description will be considered instead of Description.
3. ‘Caller’: Name of the person who raised the ticket. There are 2950 unique callers. This column does not provide any special information in differentiating the Assignment group allotment. Hence, this column can be removed for dataset.
4. ‘Assignment\_group’: Group name to which the ticket is assigned to.There are total of 74 groups out of which maximum entries belong to GRP\_0. out of 8500, 3976 entries belong to GRP\_0. This column will be our target variable.

Following is an image summarizing the data for building classifier:

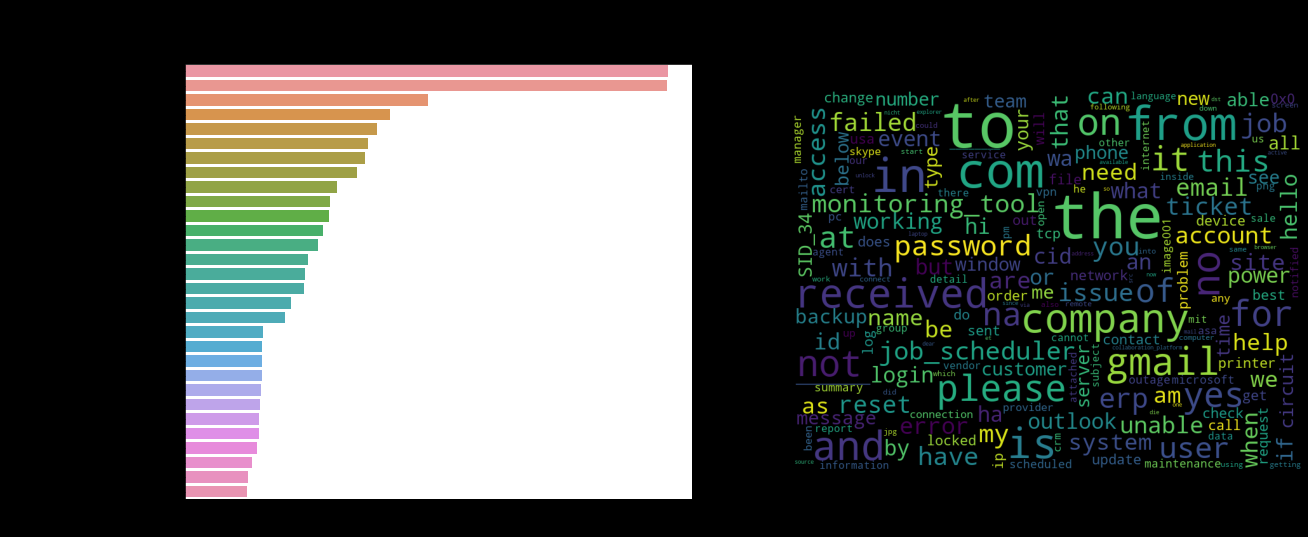


First 5 entries of data can be seen in image below:



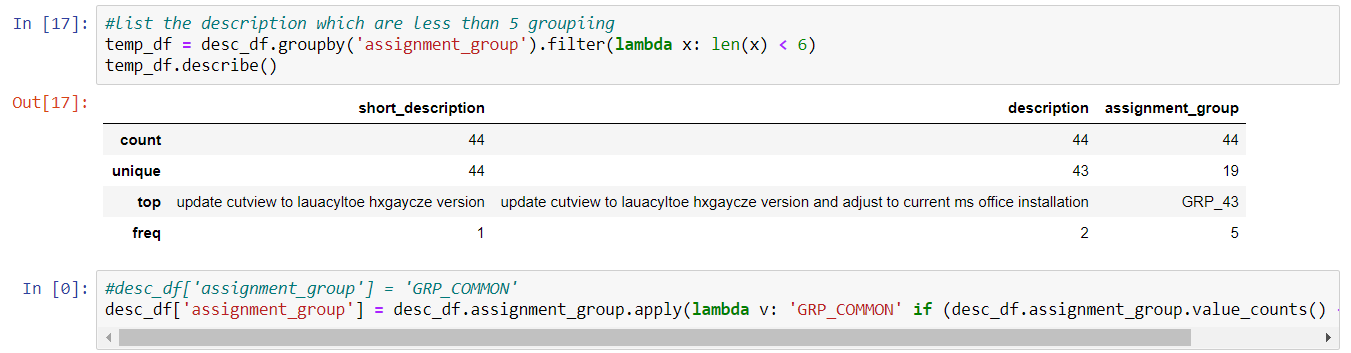
To get a visual understanding of data, Word Cloud was constructed for ‘Description’ column.

First look of Description column:



Upon data exploration, following observations were recorded:

1. There are os related new line and line termination tags.
2. Few description have header - received from : - which doesn't provide much information on classification. Few description also have footer note - Thanks/regards followed by name - which doesn't provide much information on classification.
3. There are few encoded words in description which could be name of the persons that are encrypted owing to PII governance which needs to be handled in pre-processing.
4. There are also few system drive path, software versioning number and ip addresses.
5. Some description might have attached with evidence photos which resulted in cid: tags in the description footer.
6. ‘\_’ was used to join main functions like job\_scheduler, hr\_payroll. Therefore, upon removal of ‘\_’,bigrams and trigrams will be required to improve model performance.
7. Some of the entries mention job-numbers in their description that are important in classificaiton of group. Therefore, we have to be careful not to remove these numbers while cleaning digits.
8. ‘no’, ‘not’ and similar negative words are not to be removed from data
9. There are email references that do not add value to classification
10. Column ‘caller’ does not provide any important information towards classification. Also, they need to be added to stopwords as well.
11. Punctuation and special characters needs to be removed from data.
12. There are cases where Description is either not present or Short description is more informative. In such cases, short description will have to be considered
13. There are few entries without any english words
14. NA will have to be dealt with.
15. There are few groups that have just one entry. It will be difficult to train a model in presence of such low entries. Hence, groups less than 6 entries are clubbed together to form just one group. There were 44 such entries. And all these are clubbed together under one group GRP\_COMMON. After this step, we have a total of 54 groups.

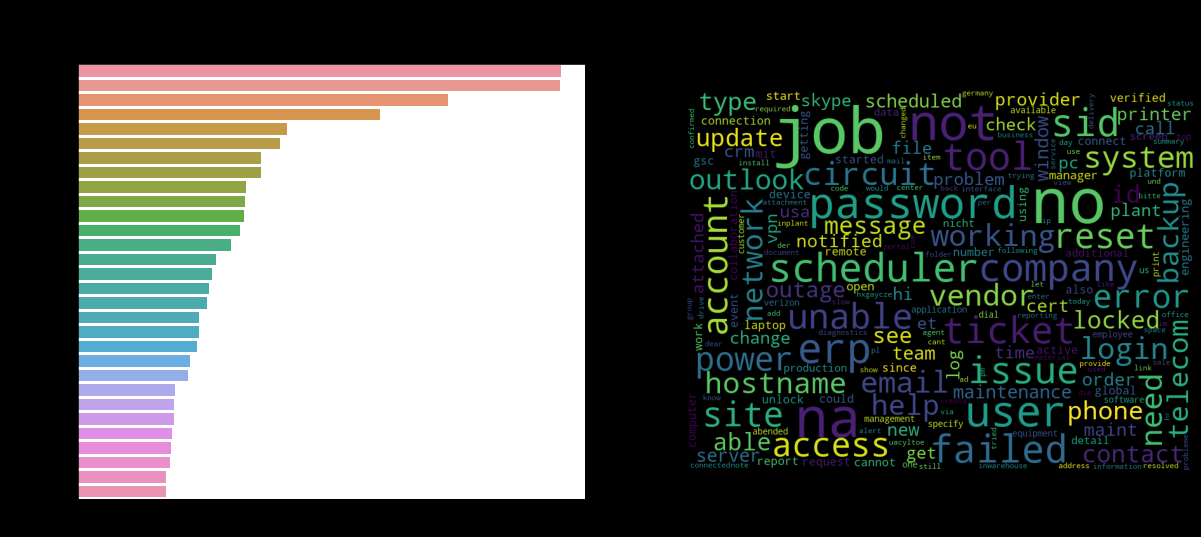


#### **2.2 Data Cleaning**

#### Following are the preprocessing steps taken on data:

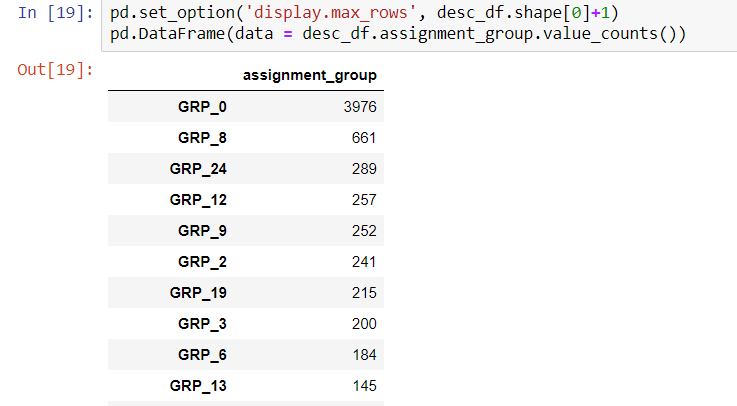
1. Wherever length of Short description was smaller than that of Description, Description was modified to include content from Short Description.
2. All Characters were converted to lower case.
3. Escape characters like new line were removed.
4. Stopwords were updated to include received from header & footer tags and image attachment reference in footer.
5. email id references were removed.
6. Punctuation characters were removed.
7. Stop words were removed.
8. Digits other than Job\_<job number> were removed.
9. Caller names were removed.

Word cloud for Description column upon cleaning:



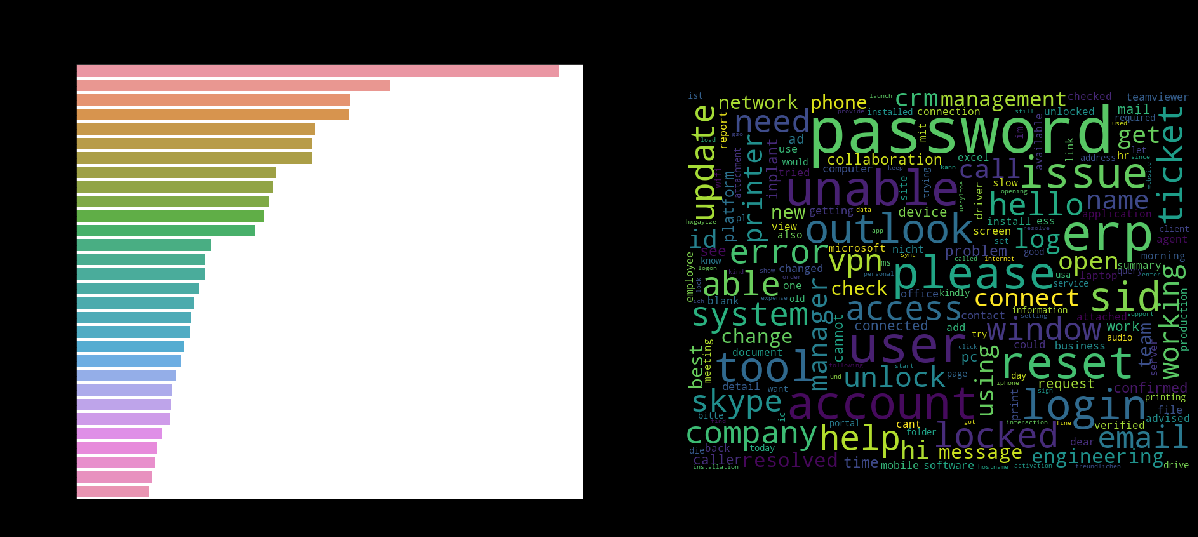
This visual representation provides with better understanding of vocabulary were are dealing with. We also know that out of these 54 groups, GRP\_0 has maximum entries. Hence, this representation will have maximum words from GRP\_0. Therefore, it is important to see word cloud for individual groups as well.

Following is the list of top 10 Assignment groups:

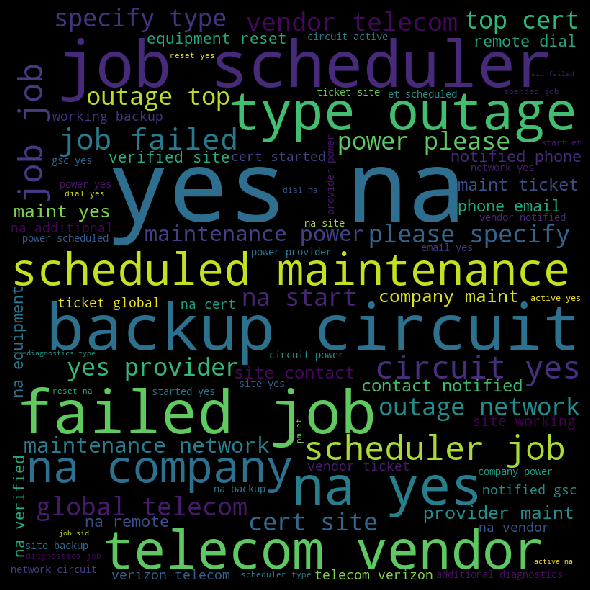


Wordclouds for top 10 groups: To have further understanding of vocabulary, word clouds for these 10 assignment groups were generated.

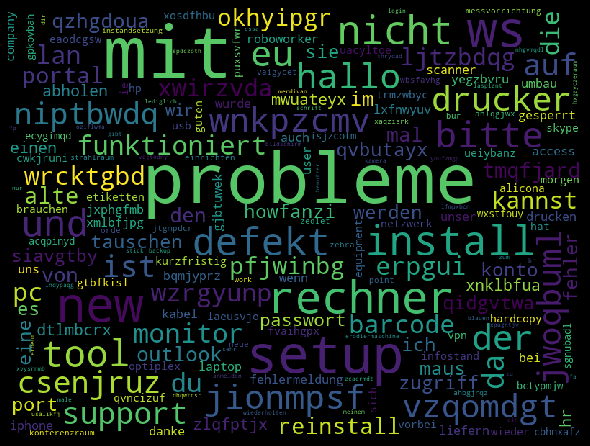
Word Cloud of GRP\_0: We can see that this looks lot similar to the word cloud of Description. Most of the issues look like they arise from inability to access an account.



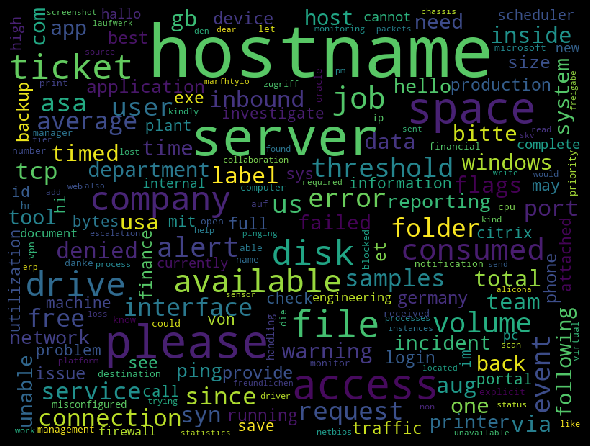
Word Cloud for GRP\_8: most of issues are concentrated around job scheduler and failed job.



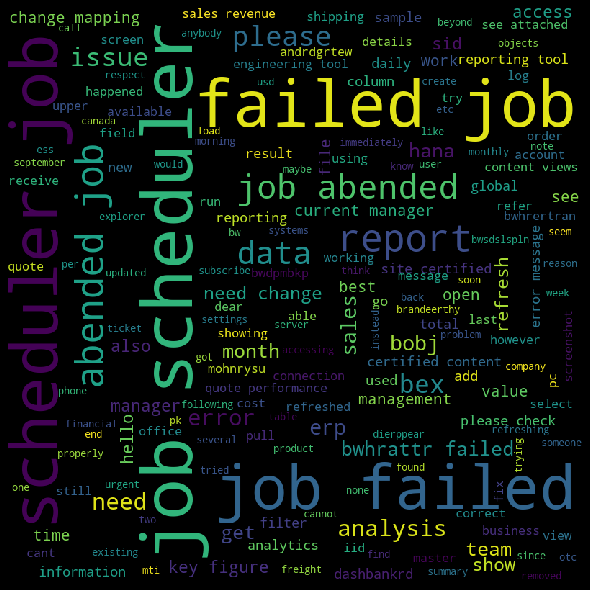
Word cloud for GRP\_24: Issues consist of installation and setup related problems



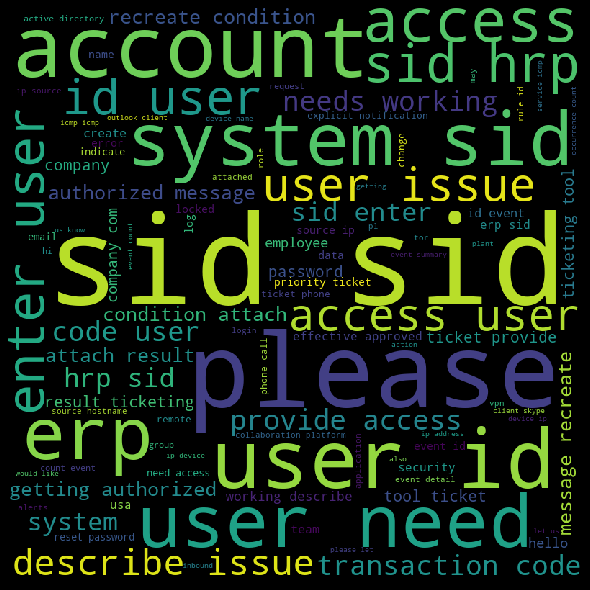
Word cloud for GRP\_12: Maximum tickets describe server access and disk space to be the issue.



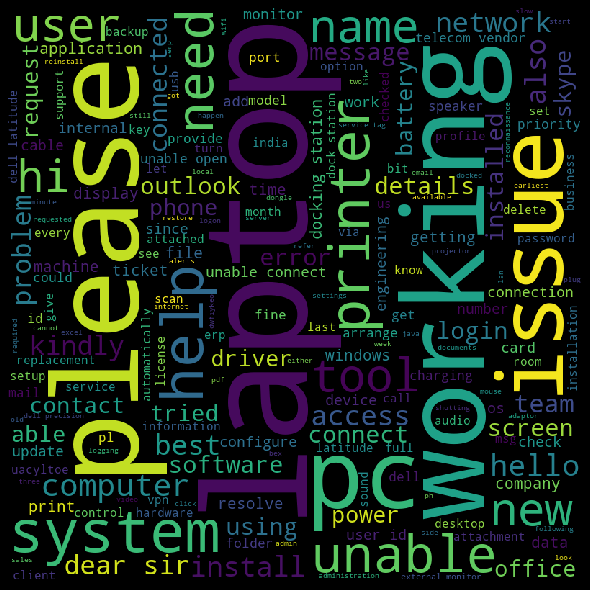
Word Cloud for GRP\_9: This looks similar to group 8 with key words being job scheduler and failed job. Job numbers become important here. Also, most of the issues apear to be sales related.



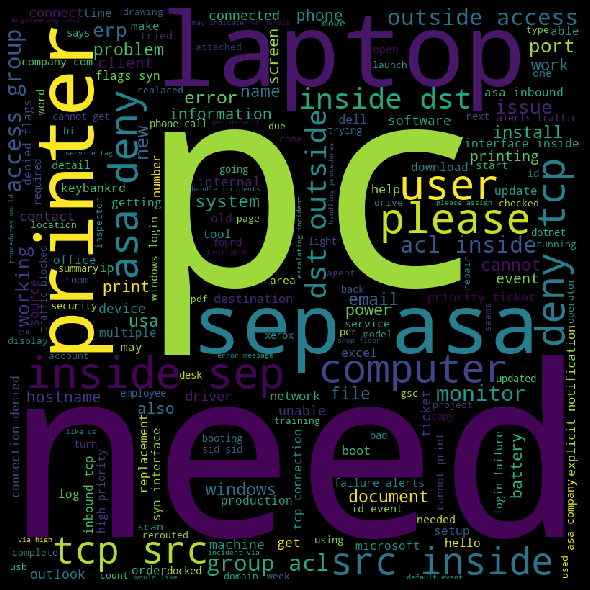
GRP\_2: Keywords here are ‘sid’, ‘hrp’, ‘erp’. Therefore , we have to be careful with removal of non-english words. Also, embeddings to be selected needs to have this vocabulary.



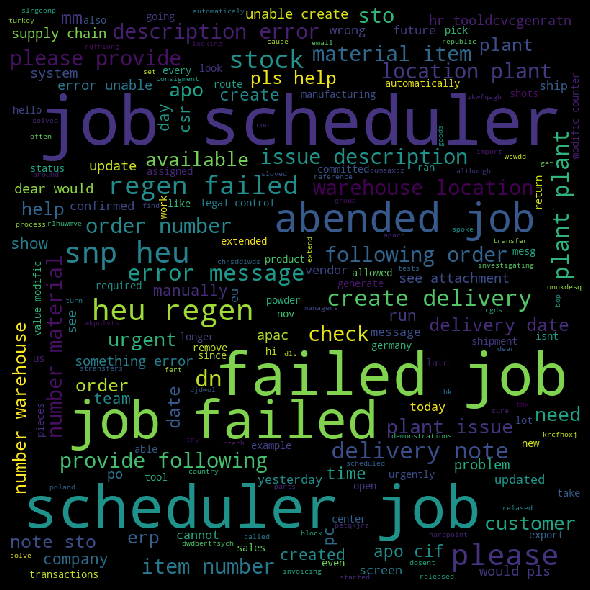
Word cloud for GRP\_19: Most tickets are regarding laptop or printer not working.



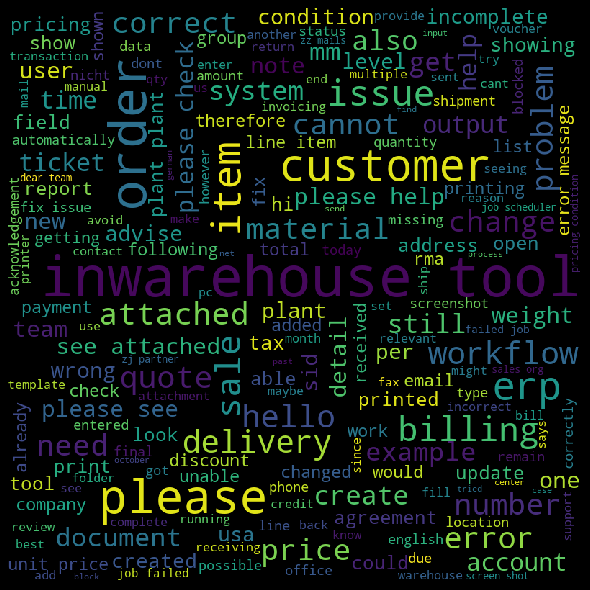
Word cloud for GRP\_3: We can see some non-english but frequently used acronyms like tcp, sep, src and dst. We have to be careful with them while data cleaning as well as closing an embedding source.



Word cloud for GRP\_6: This also is similar to GRP\_8 and GRP\_9. Job numbers become important here as well. The tickets here appear to be raised by supply chain team.



Word cloud for GRP\_13: Main keywords are order, billing, and shipment. This group looks to catering to order and billing related issues.



**2.3 N gram analysis**

Bigrams and trigramsprovides more sensible keywords that would help us in enhancing overall performance of model.

**Word cloud for unigram:**

Total no of uni-grams from corpus without stopwords : 14520

Size of uni-gram Vocabulary upon cleaning : 3049

Size of updated uni-gram Tokens based on TF-IDF frequency : 2659



**Word Cloud for bigrams:**

Total no of bi-grams from corpus without stopwords : 61580

Size of bi-gram Vocabulary upon cleaning : 30368

Size of updated bi-gram Tokens based on TF-IDF frequency: 26692



**Word Cloud for trigrams:**

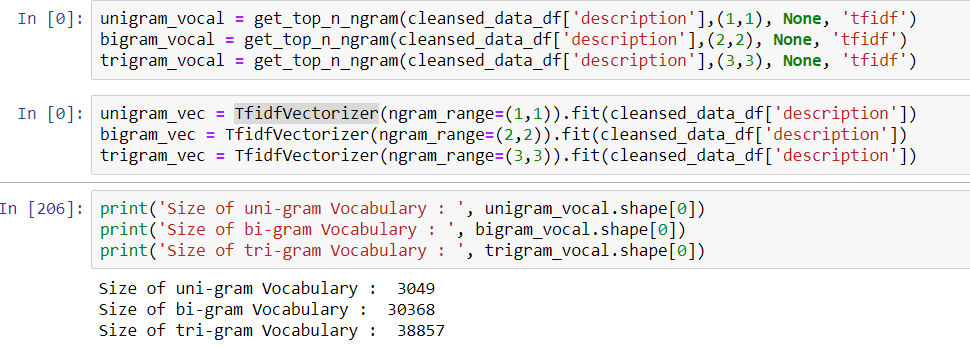
Total no of tri-grams from corpus without stopwords: 75257

Size of tri-gram Vocabulary upon cleaning : 38857

Size of updated tri-gram Tokens based on TF-IDF frequency : 36155



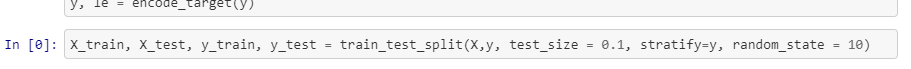
After Generation of Ngrams, TF-IDF was calculated and on its basis, high and low frequency words were removed. Data was further vectorized and input matrix consisting of unigrams, bigrams and trigrams was constructed. This was provided as input to models.



1. **Modelling**

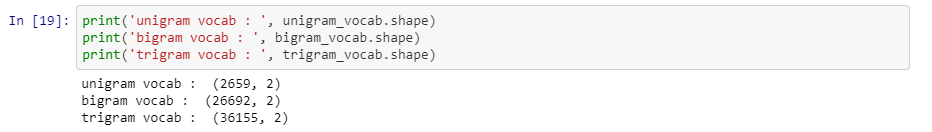
After cleansing the data,before moving to Complex models, we first started with traditional machine learning models namely, Naïve Bayes, Logistic Regression, SVM Classifier, Random Forest, XGBoost and KNN.

For modeling we have taken 90% train and 10% test data for modeling.

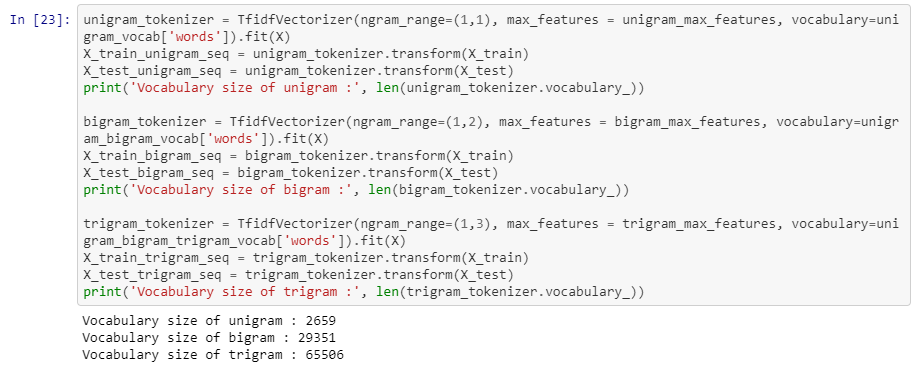


Since it is a classification problem we have used the aforementioned algorithms.

Modeling was done on uni-gram, bi-gram and tri-grams. Below is the vocab size for all the three.



Then we calculated the Tf-Idf Vectorizer for Uni-gram, bi-gram and tri-gram.



We then modeled the uni-gram, bi-grams and tri-grams vectors.

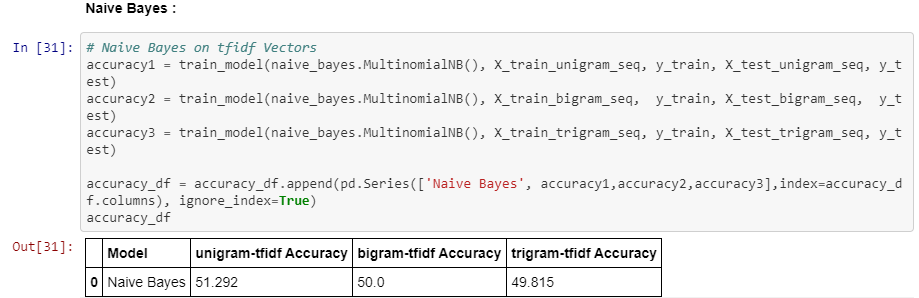
Below are the models used and their corresponding accuracy scores.

**3.1 Traditional Models**

**3.1.1 Naïve Bayes**

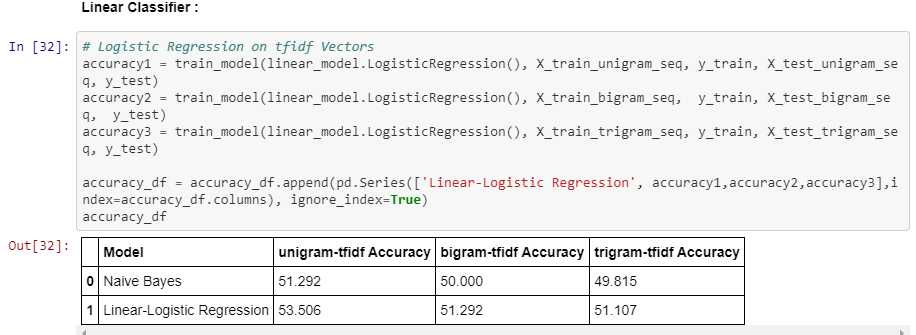
Naïve Bayes is a fast algorithm for classification problem. This algorithm is a good fit for real-time prediction, multi-class prediction, recommendation system, text classification, and sentiment analysis use cases. Naive Bayes Algorithm can be built using Gaussian, Multinomial and Bernoulli distribution. This algorithm is scalable and easy to implement for the large data set.

Due to its better performance with multi-class problems and its independence rule, Naive Bayes algorithm perform better or have a higher success rate in text classification, Therefore, it is used in [Sentiment Analysis](https://www.educba.com/sentiment-analysis-social-media/) and Spam filtering.



**3.1.2 Logistic Regression**

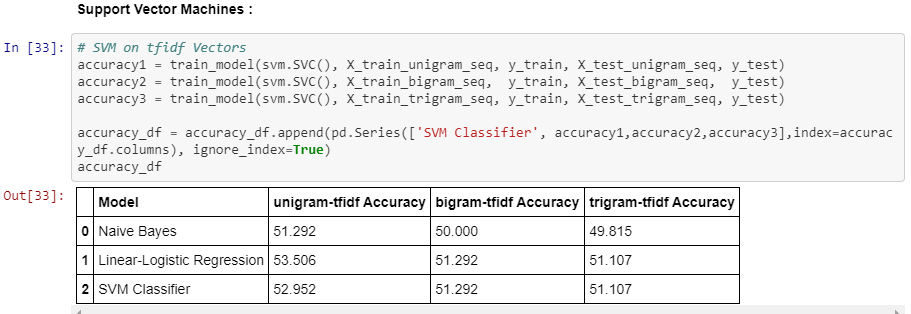
Logistic Regression is widely used technique because it is very efficient, does not require too many computational resources, it’s highly interpretable, it doesn’t require input features to be scaled, it doesn’t require any tuning, it’s easy to regularize, and it outputs well-calibrated predicted probabilities. Another advantage of Logistic Regression is that it is incredibly easy to implement and very efficient to train.



**3.1.3 SVM Classifier**

SVM separates data points using a hyperplane with the largest amount of margin. That's why an SVM classifier is also known as a discriminative classifier. SVM finds an optimal hyperplane which helps in classifying new data points. The benefit is that it can capture much more complex relationships between data points without having to perform difficult transformations on your own.

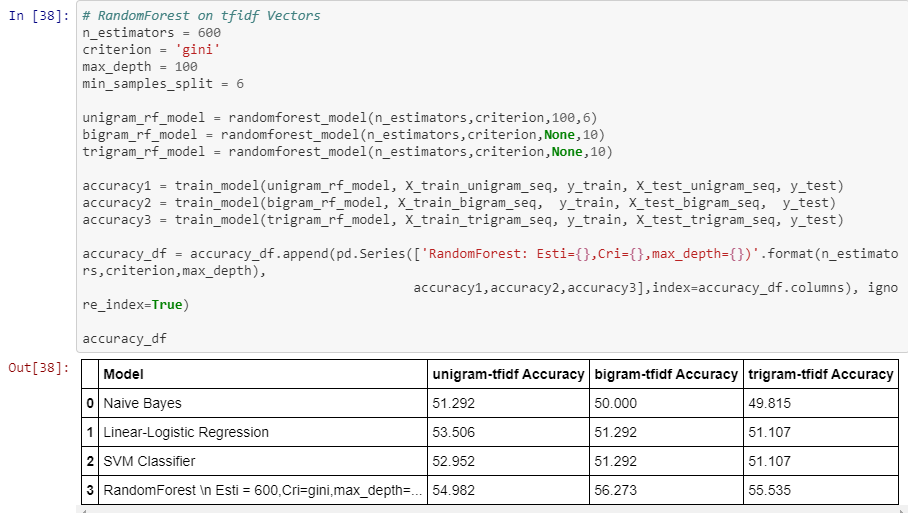
SVM's can model non-linear decision boundaries, and there are many kernels to choose from. They are also fairly robust against overfitting, especially in high-dimensional space.



**3.1.4 Random Forest**

Random forest is considered as a highly accurate and robust method because of the number of decision trees participating in the process. It does not suffer from the overfitting problem. The main reason is that it takes the average of all the predictions, which cancels out the biases.

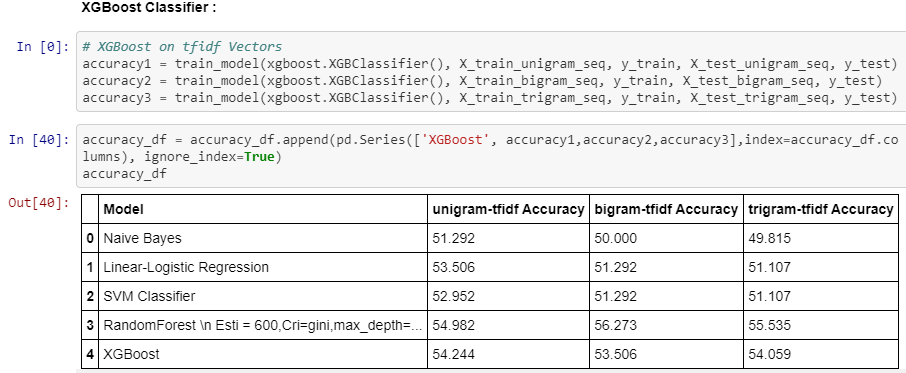
Random forests can also handle missing values. There are two ways to handle these: using median values to replace continuous variables, and computing the proximity-weighted average of missing values.



**3.1.5 XGBoost**

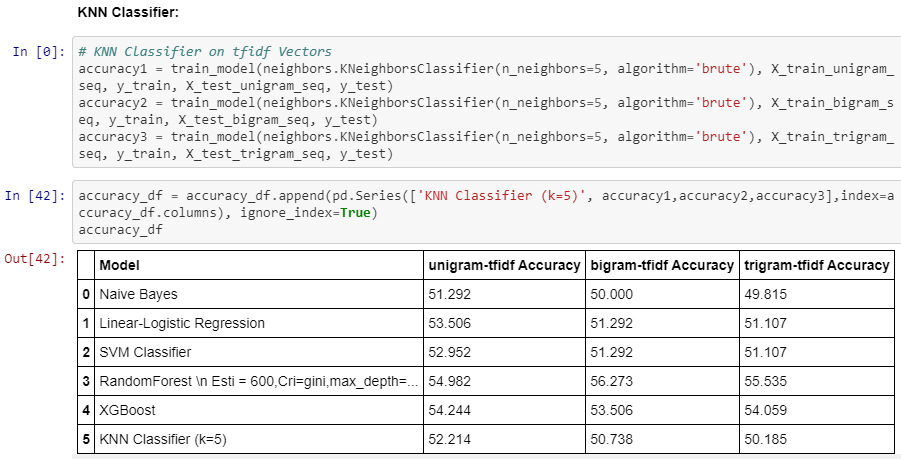
The XGBoost library implements the [gradient boosting decision tree algorithm](https://en.wikipedia.org/wiki/Gradient_boosting). XGBoost goes by lots of different names such as gradient boosting, multiple additive regression trees, stochastic gradient boosting or gradient boosting machines.

XGBoost is an implementation of gradient boosted decision trees designed for speed and performance. Generally, XGBoost is fast when compared to other implementations of gradient boosting. XGBoost dominates structured or tabular datasets on classification and regression predictive modeling problems.



**3.1.6 KNN**

KNN uses lazy training which means all computation is deferred till prediction. This works very well if we have good training data. Naïve Bayes is a quick classifier and K-NN should be preferred when the data-set is relatively small. For this particular data set the KNN was not performing well.



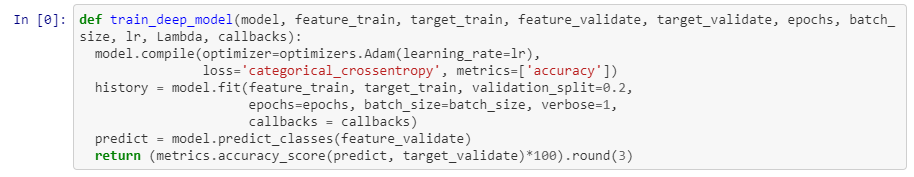
**Conclusion:** Naïve Bayes performed the worst Random Forest performed the best amongst all classifier.

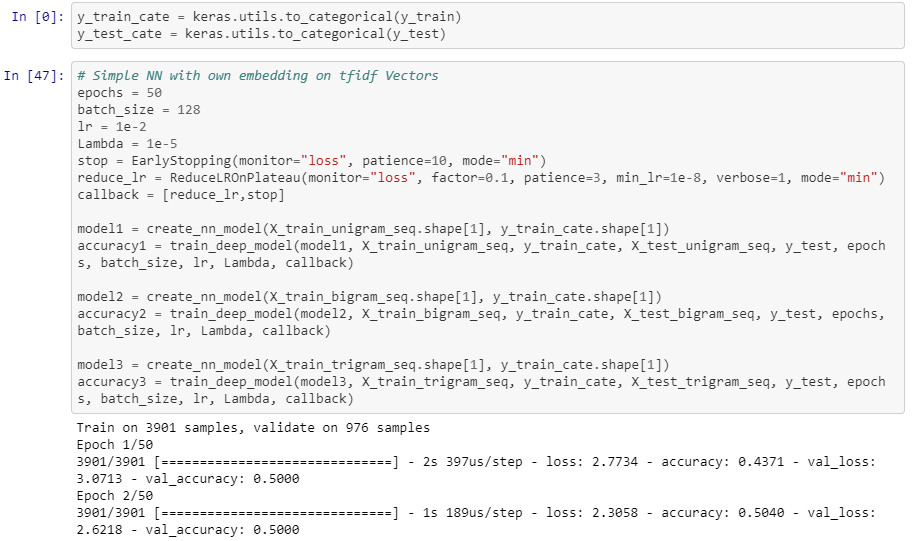
**3.2 Deep learning Models**

**3.2.1 Simple NN**

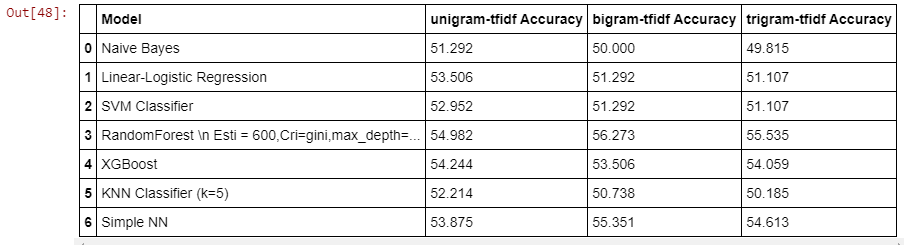
After Machine learning models, we tried deep learning models. Initially we tried simple neural network.







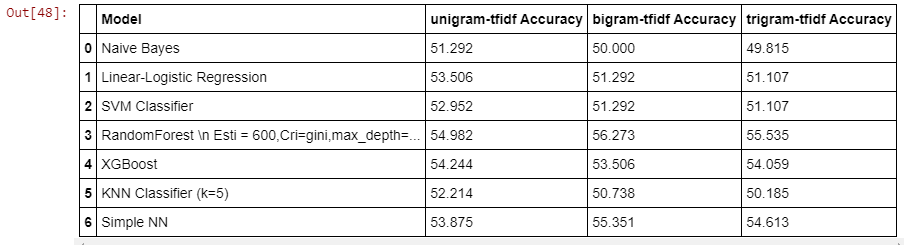
The final accuracy score of the models are below:



**Conclusion :** Random Forest performed the best amongst the basic Deep Learning and Machine Learning models.

**3.3 Hyper parameter tuning**

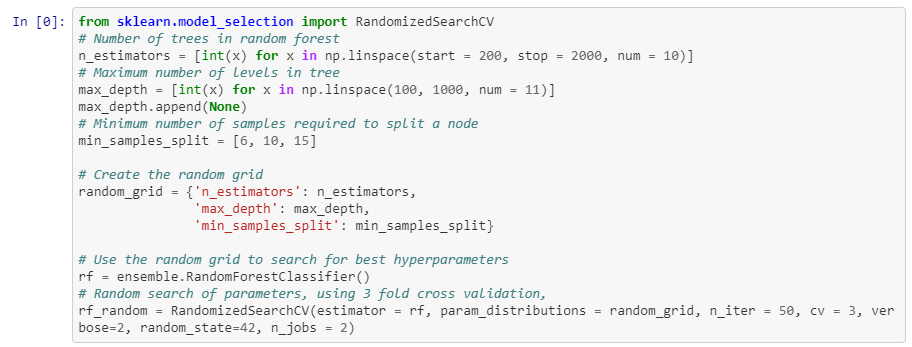
As mentioned in above section, we removed all the stop words and cleansed the data and combined the rows where there were few data sets and then did modeling on that. For better performance of the model we tested for uni-gram, bi-gram and tri-gram vectors. However, the results were not great. The accuracy score was ~54%.



Random Forest and Simple Neural Network has better accuracy score in bi-gram and tri-grams. However, the others performed well in uni-grams.

**3.3.1 Random Forest Tuning**

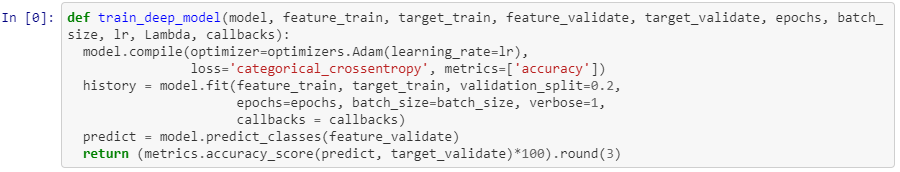
We performed hyper-parameter tuning for Random Forest to find best parameters.

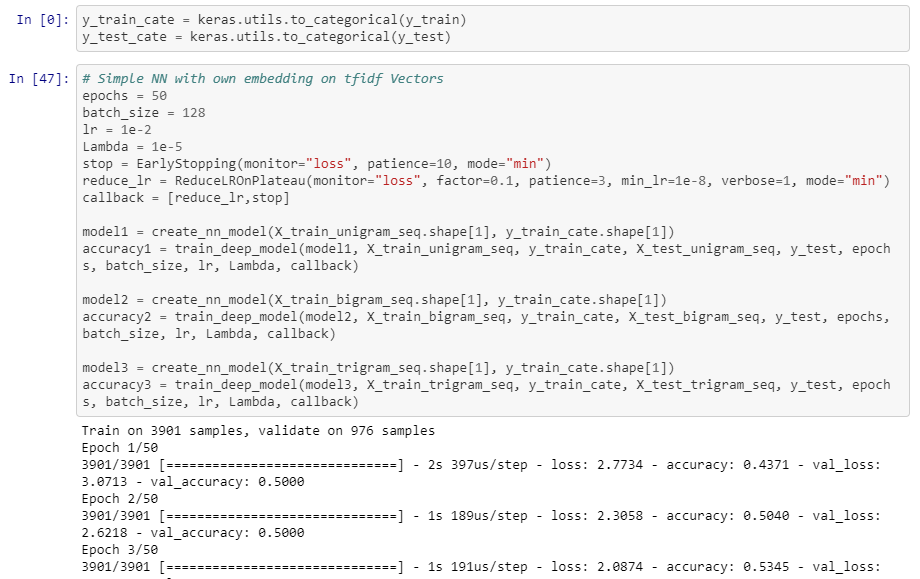


**3.3.2 Deep learning model**

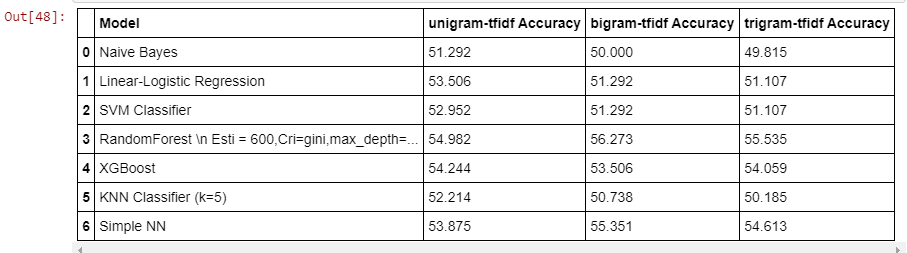
Initially the neural network score was 50.526, 52.105 and 53.333 for uni-gram, bi-gram and tri-grams. To improve the model score we added more Dense Layer, performed batch normalization, added Dropout Layer, used callback and optimizer regularization loss.







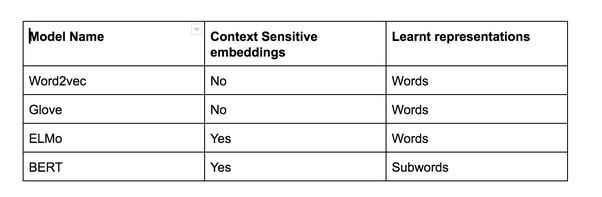
We ran the model for 50 epochs. Below are the accuracy score after performing the above mentioned steps.



1. **Embeddings**

There are different types of Embeddings available. Following is a list of a select few.

1. ELMO
2. BERT
3. Fast Text
4. Word2Vec
5. GloVe



Out of Above mentioned models, ELMO and BERT are not content sensitive embeddings. And therefore, we believe that these may not be useful in our application. Hence, our main focus was on Fast Text, Word2Vec and Glove.

**4.1 Fast Text:**

FastText is a library created by the Facebook Research Team(FLAIR) for efficient learning of word representations and sentence classification.

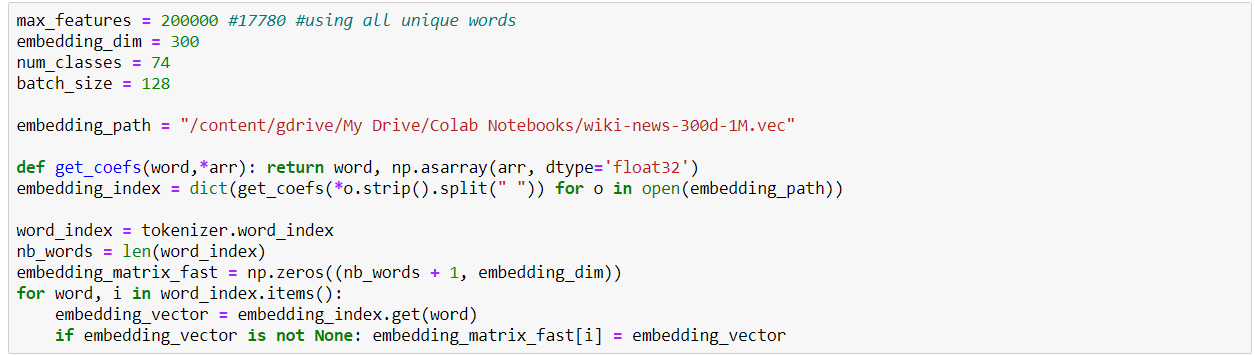
Uses of FastText

Text Classification

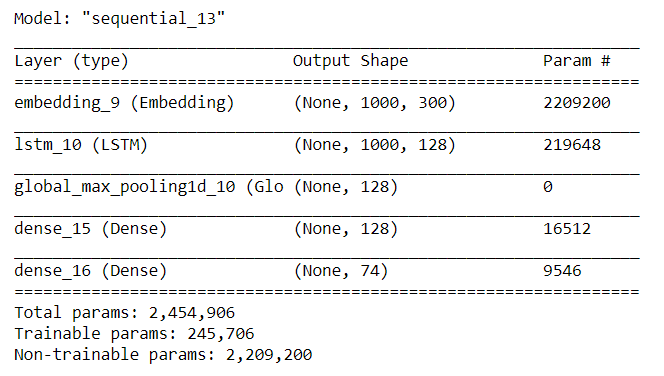
Word Representation

Word vectors treats every single word as the smallest unit whose vector representation is to be found but FastText assumes a word to be formed by a n-grams of character, for example, sunny is composed of [sun, sunn,sunny],[sunny,unny,nny]  etc, where n could range from 1 to the length of the word. This new representation of word by fastText provides the following benefits over word2vec or glove.

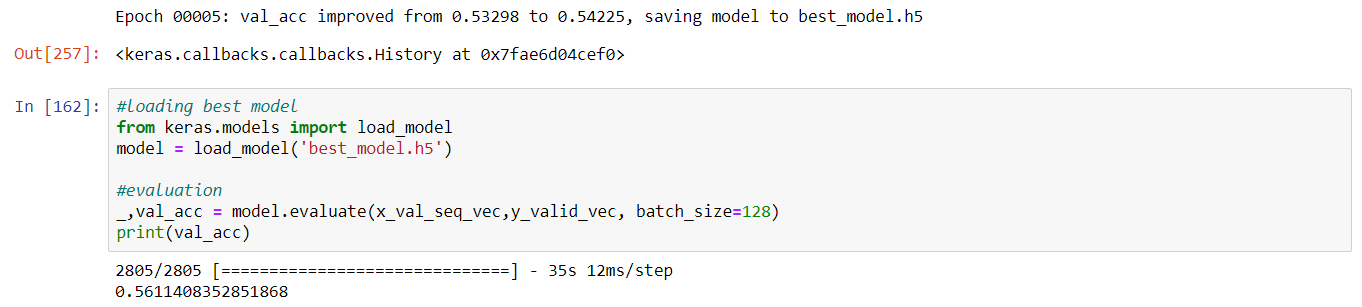
It can give the vector representations for the words not present in the dictionary (OOV words) since these can also be broken down into character n-grams. word2vec and glove both fail to provide any vector representations for words not in the dictionary. character n-grams embeddings tend to perform superior to word2vec and glove on smaller datasets.



Model was constructed using these embeddings:



Model was trained for 5 Epochs.



**Output:** Validation Accuracy was found to be 54.2%.

**4.2 Word2Vec:**

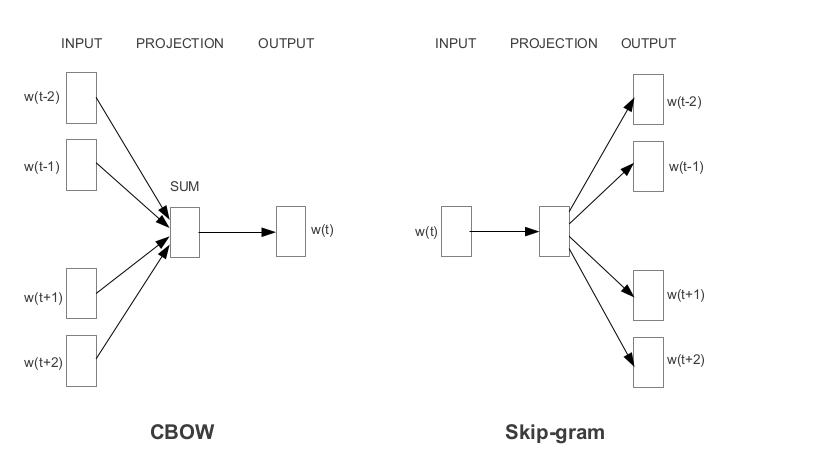
Word2Vec is one of the most popular pretrained word embeddings developed by Google. Word2Vec is trained on the Google News dataset (about 100 billion words). The architecture of Word2Vec is really simple. It’s a feed-forward neural network with just one hidden layer. Hence, it is sometimes referred to as a **Shallow Neural Network architecture**.

Word2Vec is classified into two approaches:

* Continuous Bag-of-Words (CBOW)
* Skip-gram model

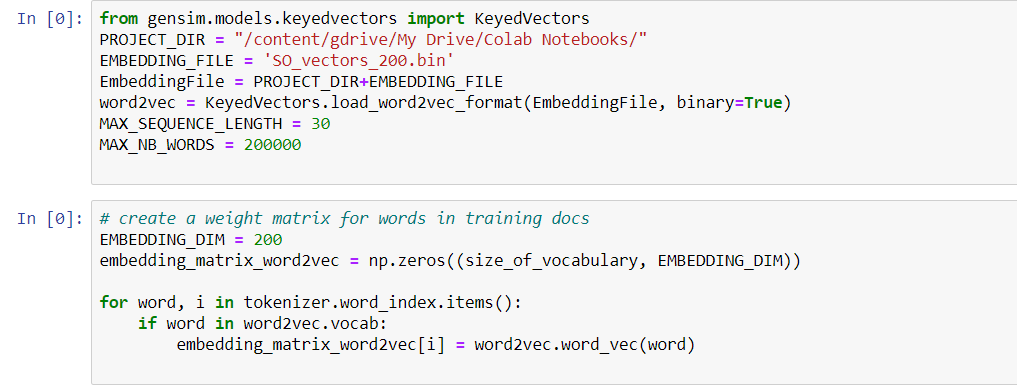
Continuous Bag-of-Words (CBOW) model learns the focus word given the neighbouring words whereas the Skip-gram model learns the neighbouring words given the focus word. That’s why, Continuous Bag Of Words and Skip-gram are inverses of each other.

CBOW accepts multiple words as input and produces a single word as output whereas Skip-gram accepts a single word as input and produces multiple words as output.

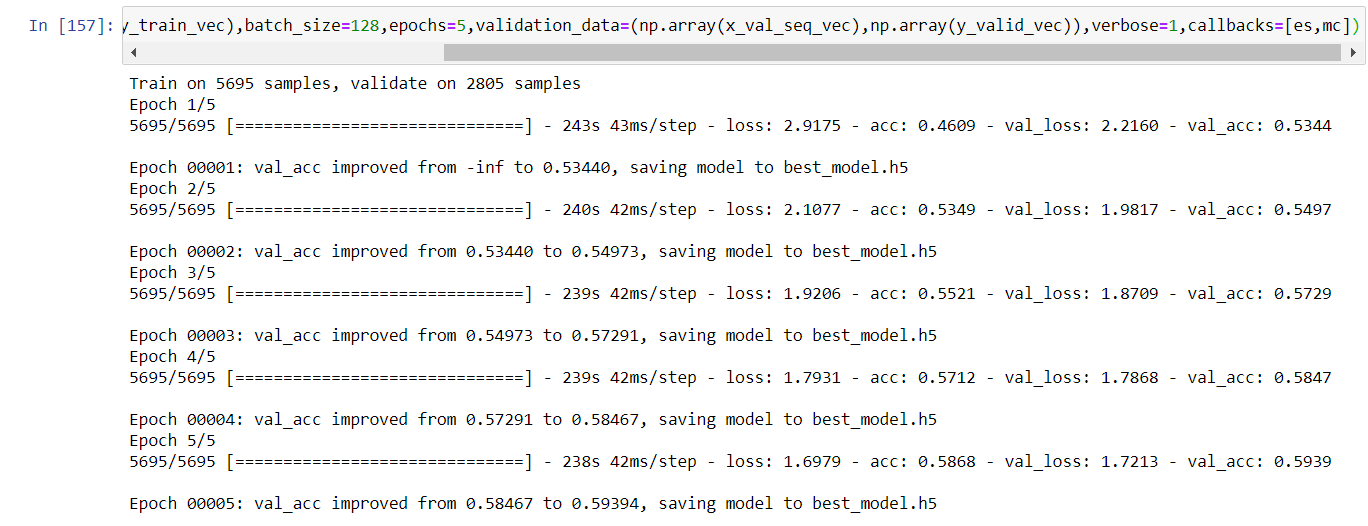
[](https://cdn.analyticsvidhya.com/wp-content/uploads/2020/03/Screenshot-from-2020-03-12-13-05-42.png)

CBOW being probabilistic is nature, it is supposed to perform superior to deterministic methods(generally). It is low on memory. It does not need to have huge RAM requirements like that of co-occurrence matrix where it needs to store three huge matrices.

Skip-gram with negative sub-sampling outperforms every other method generally



Model similar to FastText was constructed using this embedding layer. Model was run for 5 epochs.



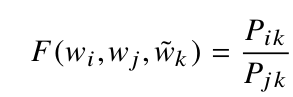
**Output:** Maximum validation accuracy was found to be 59.4%

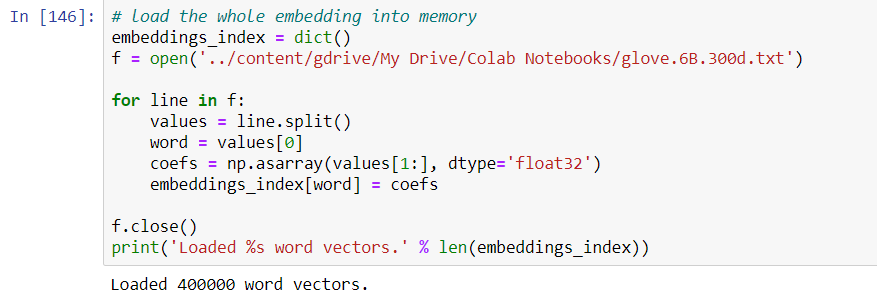
**4.3 GloVe:**

The basic idea behind the GloVe word embedding is to derive the relationship between the words from Global Statistics. One of the simplest ways is to look at the co-occurrence matrix. A co-occurrence matrix tells us how often a particular pair of words occur together. Each value in a co-occurrence matrix is a count of a pair of words occurring together. A co-occurrence matrix tells us how often a particular pair of words occur together. Each value in a co-occurrence matrix is a count of a pair of words occurring together.

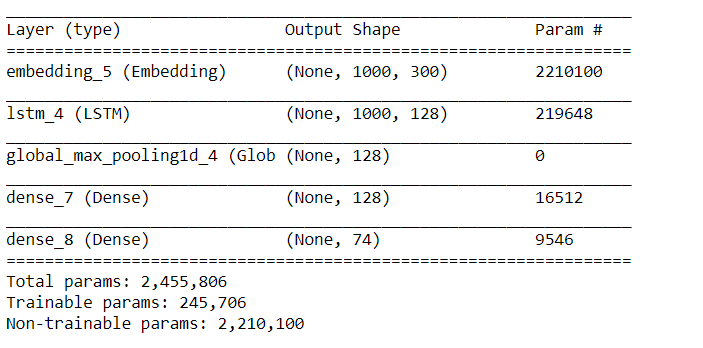
We derive the relationship between the words using simple statistics. This the idea behind the GloVe pretrained word embedding.

GloVe learns to encode the information of the probability ratio in the form of word vectors. The most general form of the model is given by:





Model was constructed using LSTM layer



The model was run for 5 Epochs. 

|  |
| --- |
|  |

**Output :** Validation Accuracy was found to be 57.2%

**4.4 BiLSTM**

All the three Embeddings were incorporated in model using BiLSTM layers as well and Accuracy was recorded. Following table gives a comparison of accuracy achieved with these models.

|  |  |  |
| --- | --- | --- |
| **Embedding** | **LSTM** | **BiLSTM** |
| Glove | 57.6 | 60.2 |
| Word2vec | 59.3 | 61.8 |
| Fastext | 54.2 | 57.6 |
|  |  |  |

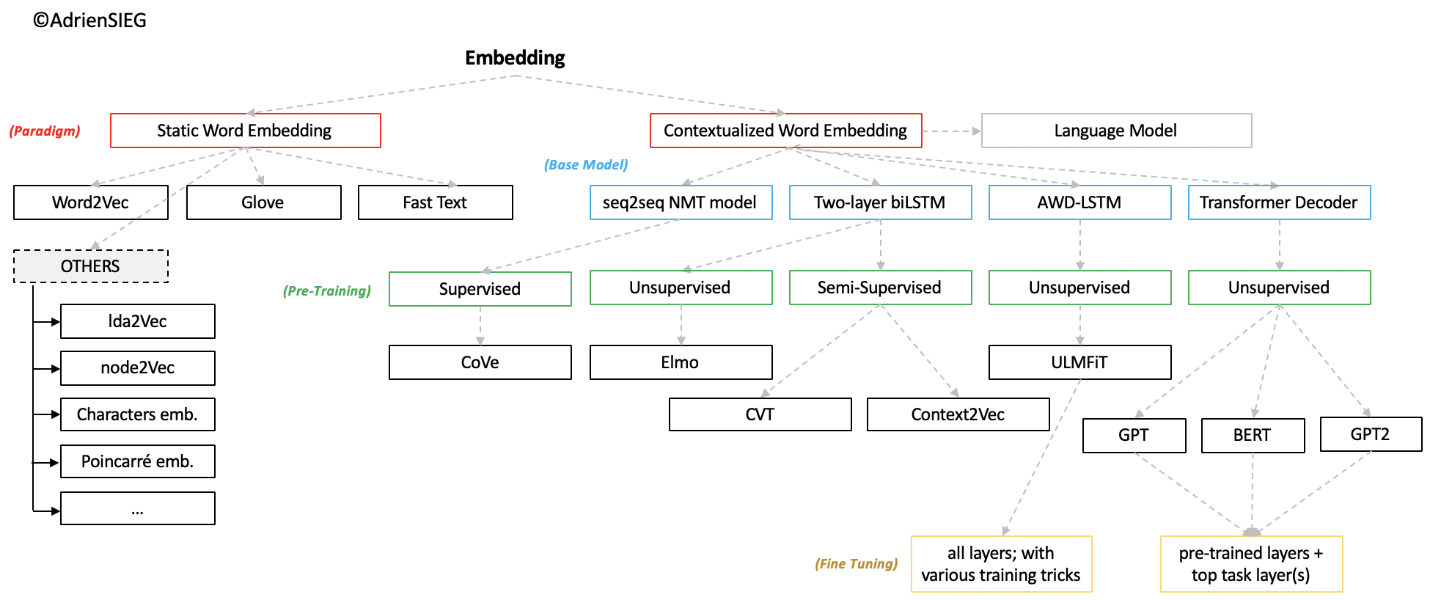
**Conclusion**:

Word2vec , Glove and Fastext are more about word embeddings. So the above table shows the overall accuracy of the different pretrained embeddings with two models LSTM and BiLSTM. The use of BiLSTM shows good results.

**4.5 Future Improvements :**

Pretrained models will be used instead of pretrained embeddings . Below are some of the examples.

In Future the embeddings will be dethroned and will be replaced by pretrained language models.



**5. Next Steps**

Based on our efforts till now, We have achieved best accuracy of 61.8% . Going forward, Efforts will be made to achieve higher accuracy. More model building and testing will be performed with focus on model construction parameters, layer selection and identifying correct node size. Other than Static word embeddings, Contextualized word embedding will be experimented with as well.