International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-10 Issue-3, February 2021

# Building ML Based Intelligent System to Analyze Production LSI (Live Site Incidents)

**Himanshu Bajpai**

***Abstract*: Providing support on the rolled-out application/services is one of the major factors in increasing the customer satisfaction which in turn increases the customer retention. Since we are in the era of automation where most of the day-to-day jobs are taken care of or are facilitated by the technologies around us, hence there is a need to reduce manual effort in triaging the support tickets and hence facilitating the person on call to better close the tickets on time with proper remediation. The machine learning model which will be the product of this complete paper will not only help in classifying the tickets but also, if applicable will give the best possible remediation of the ticket there by reducing the manual effort and the time taken on providing necessary solution on the ticket. The objectives of the work are as follows -***

***a) Understand the data that is present in the ticket and figure out the basic understanding like, categories of issues, trends etc.***

***b) Prepare the data which is ready for applying different classification algorithms.***

***d) Identify the best machine learning model which can classify the new incident with utmost accuracy.***

***e) Prepare a machine learning model which can suggest the best possible remediation of the ticket.***

***f) Integrate the best classification model and solution recommender model and wrap it as an API which can be used by end user.***

***Keywords: Application Development, Classification, Model Evaluation, Natural Language Processing, Semantic Similarity***

## I. INTRODUCTION

Support system is one of the important parameters in deciding the fate of any application. We use many applications in our day-to-day life, and how well the support folks helped in resolving our concerns while using the application leads us to be regular user of that application. The manual effort in resolving the tickets within SLA (Service Level Agreement) is time consuming and vulnerable to human errors. So, there is a need to build an intelligent system can facilitate in answering the grievances of the customer. However, the challenge while working with text data is,

- We cannot use the text data directly for modelling any Machine Learning algorithm.
- It is prone to contain insignificant terms which do not hold any importance.
- It is prone to contain incorrect/misspelled words, which can mislead the model.

This article proposes a baseline machine learning solution which can help in reducing human intervention in deciding what is the issue being faced by the customer and what can be the best possible solution for the filed incident using data science techniques.

The experiments are conducted on the actual production incidents for a cloud-IAAS (Infrastructure As-A Service) based web application which involves deployments of the resources in the cloud and hence involves below possible categorization –

- Cloud Deployment Failure
- Failure while saving entry in the database.
- Failure related to the unavailability of application.

The task here is to classify the incoming incident into the best suitable category and then providing the best solution for the filed incident. The dataset is taken from the incident repository related to the application.

## II. STRUCTURE OF THE DATASET

There are different datasets which are used as part of this work:

1. *Incident Category* –
   - This dataset acts as the primary source for categorization of tickets.
   - Columns- IncidentID, Title, and Category.

**NOTE**: Categorization of the tickets was done manually.

2. *Incident with Solution* –
   - This dataset contains information about the solution for the ticket provided by the team.
   - Columns – IncidentID, Summary, and TSG (Trouble Shooting Guide).
   - This dataset will be used to recommend the possible fix for the tickets based on semantic similarity between the summary provided by the user and the summary for already resolved tickets.
   - Since there were certain tickets had no summary, summary was refined manually to save time and maintain consistency.

Total number of words across all the incident titles with and without scrubbing of data –

```
In [20]: len(get_words_in_corpus('Title'))
Out[20]: 7178

In [23]: len(get_words_in_corpus('refined_title'))
Out[23]: 2966
```

**Fig.1.Total words with and without scrubbing of data.**

Manuscript received on January 17, 2021.
Revised Manuscript received on January 22, 2021.
Manuscript published on February 28, 2021.
* Correspondence Author
**Himanshu Bajpai*,** Engineering Services, Infosys Limited, Pune (M.H), India. Email: himanshubajpai869@gmail.com

Distinct words in the refined dataset –

```
In [22]: word_in_corpus = get_words_in_corpus('refined_title')

In [25]: len(set(word_in_corpus))
Out[25]: 679
```

**Fig.2.Distinct words in the dataset.**

### A. Pre-processing data for Modelling

Since the data is created using the incidents filed by user it is prone to have lot of discrepancies and since the entire data is in text form, there is a need to pre-process the data before applying machine learning technique.

Example – If a user files incident as "The ARM deployment of resource group RG1 has failed in the west us2 due to invalid parameter passed for deployment".

Here, we can see that words like "The", "of", "RG1", "has", "in", "west", "us2" etc., do not add any value in the reason why the incident must be filed. So, we need to pre-process the data so that we get the texts which add value to the machine learning algorithms.

Major Discrepancies or issues in the dataset –

- Upper and lower case in the text.
- URLs in the text.
- Alpha-numeric string in the text.
- Punctuations
- Stop words related to the application like resource name, server name, region name etc.
- Abbreviations related to the application.
- English stop words.
- Reducing the words to their root forms (Lemmatization)

A python-based script was written to get the clean text from the input text provided:

```
get_clean_text(
    "The ARM deployment of resource group RG1 has failed in the west us2 due to invalid parameter passed for deployment")

'arm deployment resource group fail due invalid parameter pass deployment'
```

**Fig.3.Sample output showing the clean text from the input text provided.**

As seen in the above figure, all the stop words, alpha-numeric strings are removed, and words are reduced to root word. Example- "**failed**" is reduced to "**fail.**"

### B. Converting the Category to numeric

The dataset is divided in four categories, namely, 'Deployment', 'Org', 'Others', 'Unhealthy Instance'.

```
set(refined_dataset["Category"])
{'Deployment', 'Org', 'Others', 'UnhealthyInstance'}
```

**Fig.4.List of Categories in the dataset**

Since machine learning model works with numeric data, let's convert the categorization in the numeric form, where 1 corresponds to Deployment, 2 to Org, 3 to Unhealthy Instance and 4 to others.

Categories after converting string to numeric as mentioned above –

```
set(refined_dataset['Category'])
{'1', '2', '3', '4'}
```

**Fig.5.Categories after converting them into integers.**

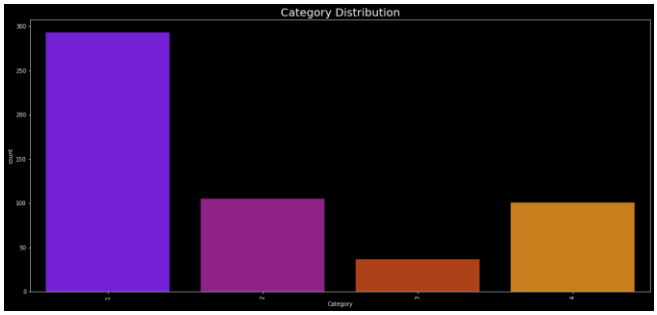


**Fig.6.Category Distribution in the corpus.**

```
refined_dataset['Category'].value_counts()

Deployment            293
Org                   105
Others                101
UnhealthyInstance      37
Name: Category, dtype: int64
```

**Fig.7.Value Count of each category.**

As seen from the above figure, the data set is biased towards 'Deployment' category, which can harm model performance and lead to over fitting model as well. Hence, to proceed, only three categories were considered under the current work – Deployment, Org and Others, and from each category 101 rows were taken.

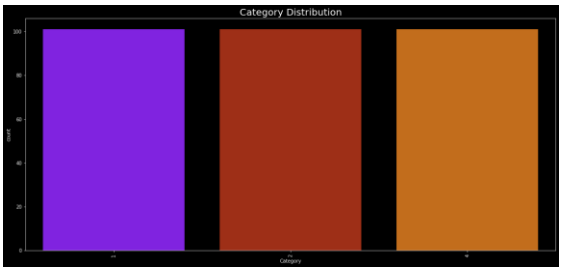


**Fig.8.Dataset containing equal number of rows for all the three categories.**

## III. EVALUATION OF THE CLASSIFICATION MODEL

Let's have some idea on the basic evaluation metrics for classification –

### A. Confusion Matrix

Let us assume, we have a classification model predicting if the issue reported is deployment issue or not with YES for the cases where the issue reported was deployment issue and NO for the cases where the issue reported was not deployment issue.

**Table- I: Sample Confusion Matrix.**

| N = 170 | Predicted<br>Is Deployment Issue?<br>No | Predicted<br>Is Deployment Issue?<br>Yes |
|---|---|---|
| Actual<br>Is Deployment Issue? No | 55 | 10 |

*Retrieval Number: 100.1/ijeat.C21780210321*
*DOI:10.35940/ijeat.C2178.0210321*
*Journal Website: www.ijeat.org*



*Published By:*
*Blue Eyes Intelligence Engineering and Sciences Publication*

| | | |
|---|---|---|
| Actual Is Deployment Issue? Yes | 5 | 100 |

Important terms related to confusion matrix –

***True Positive***

These are the cases where model predicted the issue as 'Yes' and actually it turned out to be 'Yes' as well. In the above table, that value is 100.

***True Negative***

These are the cases where model predicted the issue as 'No' and it turned out to be 'No' as well. In the above table, that value is 55.

***False Positive***

These are the cases where model predicted the issue as 'Yes' and it was 'No'. In the above table, the value is 10.

***False Negative***

These are the cases where model predicted the issue as 'No' and it was 'Yes'. In the above table, the value is 5.

***Precision***

Ratio of the true positive labels to total number of positive labels predicted by the model, i.e.

**Precision = True Positive / (False Positive + True Positive)**

***Recall***

It basically tells how well the model predicted the positive cases, i.e.

**Recall = True Positive / (False Negative + True Positive)**

***F1-Score***

It is single metric addressing the concerns in Precision and Recall both i.e.

**F1-Score = 2* (Precision * Recall) / (Precision + Recall)**

From the sample confusion matrix,

Precision = 100 / (100 + 10) = 0.9090

Recall = 100 / (100 + 5) = 0.952

F1-Score = 2 * 0.9090 * 0.952 / (0.9090 + 0.952) = 0.8095

### B. Cross Validation

Since we want to make sure that model performs well with respect to unseen data, we apply one more method to identifying the best model for this project using **Cross Validation** technique. This is one of the ways of testing the skill of the trained model by letting it make predictions on the unseen data and later we can use the performance shown by the model against all such scenarios to finally determine the best accuracy of the model.

Example – If a dataset comprises of 100 rows, then we can train the model using first 70 rows and then test it using next 30 rows. This will give us an accuracy with which model classifies the 30 rows. Likewise, we can again train a model by taking alternate 70 rows and then use the remaining rows to test the model. This will also give us some accuracy as well. Likewise, we can train model against a different set of training data and test it also against the different set of test data and later can get average of all the accuracies obtained to signify the overall accuracy of the model.

This is known as **K-Fold Cross Validation**, where K is number of samples which are generated from the dataset.

## IV. APPLICATION OF THE DIFFERENT MACHINE LEARNING ALGORITHMS ON THE DATASET

Since, we have pre-processed our data set in the previous sections, now we will apply machine learning algorithms to classify the incidents.

For this Python's scikit-learn library is used which is having various machine learning utilities available as open source.

Results from the model predictions are as below –

*Decision Tree*

Accuracy using Decision Tree is :- 70.32967032967034

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 1 | 0.90 | 0.78 | 0.84 | 23 |
| 2 | 0.60 | 0.88 | 0.71 | 34 |
| 4 | 0.76 | 0.47 | 0.58 | 34 |
| accuracy | | | 0.70 | 91 |
| macro avg | 0.75 | 0.71 | 0.71 | 91 |
| weighted avg | 0.74 | 0.70 | 0.70 | 91 |

**Fig.9.Result from Decision Tree Classifier.**

*Logistic Regression*

Accuracy using Logistic Regression is :- 71.42857142857143

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 1 | 0.72 | 0.78 | 0.75 | 23 |
| 2 | 0.72 | 0.76 | 0.74 | 34 |
| 4 | 0.70 | 0.62 | 0.66 | 34 |
| accuracy | | | 0.71 | 91 |
| macro avg | 0.71 | 0.72 | 0.72 | 91 |
| weighted avg | 0.71 | 0.71 | 0.71 | 91 |

**Fig.10.Result from Logistic Regression Classifier.**

*Support Vector Machine*

Accuracy using Support Vector Machine is :- 25.274725274725274

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 1 | 0.25 | 1.00 | 0.40 | 23 |
| 2 | 0.00 | 0.00 | 0.00 | 34 |
| 4 | 0.00 | 0.00 | 0.00 | 34 |
| accuracy | | | 0.25 | 91 |
| macro avg | 0.08 | 0.33 | 0.13 | 91 |
| weighted avg | 0.06 | 0.25 | 0.10 | 91 |

**Fig.11.Result from Support Vector Machine Classifier.**

*Naïve Bayes*

Accuracy using Naive Bayes is :- 69.23076923076923

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 1 | 0.73 | 0.96 | 0.83 | 23 |
| 2 | 0.65 | 0.59 | 0.62 | 34 |
| 4 | 0.70 | 0.62 | 0.66 | 34 |
| accuracy | | | 0.69 | 91 |
| macro avg | 0.69 | 0.72 | 0.70 | 91 |
| weighted avg | 0.69 | 0.69 | 0.68 | 91 |

**Fig.12.Result from Naïve Bayes Classifier.**

*K-Nearest Neighbor*

Accuracy using KNN is :- 65.93406593406593

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 1 | 0.47 | 0.96 | 0.63 | 23 |
| 2 | 0.84 | 0.76 | 0.80 | 34 |
| 4 | 0.92 | 0.35 | 0.51 | 34 |
| accuracy | | | 0.66 | 91 |
| macro avg | 0.74 | 0.69 | 0.65 | 91 |
| weighted avg | 0.78 | 0.66 | 0.65 | 91 |

**Fig.13.Result from Decision Tree Classifier**



Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication

To test the model against different combination of training and test data, 10- Fold Cross Validation was also performed and the accuracies were as followed –

```
Average accuracy of Decision Tree Classifier after applying 10- Fold CV is 75.92583732057416
Average accuracy of Logistic Regression after applying 10- Fold CV is 83.43301435406698
Average accuracy of SVM after applying 10- Fold CV is 36.82296650717703
Average accuracy of Naive Bayes after applying 10- Fold CV is 77.45933014354067
Average accuracy of KNN after applying 10- Fold CV is 70.39234449760765
```

**Fig.14.Result on applying Cross Validation on all the models.**

| Algorithm | Accuracy without Cross Validation | Accuracy with Cross Validation |
|---|---|---|
| Decision Tree | 70.32 | 75.92 |
| Logistic Regression | 71.42 | 83.43 |
| Support Vector Machine | 25.27 | 36.82 |
| Naïve Bayes | 69.23 | 77.45 |
| K Nearest Neighbor | 65.93 | 70.39 |

**Table-II: Accuracies of Different Classification model with and without cross validation.**

As seen from the above table and reports of the different classification model applied on the dataset, **Logistic Regression** is the clear winner in classifying the incidents into the pre-defined categories.

### A. Tune the Logistic Regression Model to get the optimum parameters for modelling.

The focus in this section will be on tuning the Logistic Regression Model to get the optimum values of the parameters required by the model. So far, we have already tried to get the best working model by pre-processing the data and then we have applied different machine learning techniques on the processed dataset and in all the cases we used default parameters. Now, here we are going to explore how to get the values of the parameters required by the model to produce the best working Logistic Regression Model.
The default parameters were used in the earlier section to get the optimum model for this study :

```
lr_classifier
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='warn', tol=0.0001, verbose=0,
                   warm_start=False)
```

**Fig.15.Default Parameters for Logistic Regression**

For this study, we will focus on below parameters and see how close we can reach to the accuracy obtained from the cross validation -
**Penalty** - Used to specify the norm used in the penalization. Accepted values - 'l1', 'l2', 'elasticnet' or 'none'.
**C** - Inverse of regularization strength; must be a positive float.
Process which we will follow is as below –

- Prepare the set of different values which can be taken by penalty and C.
- Apply Grid Search Cross Validation on the Logistic Regression model and pass the hyper parameters in the above figure to the model.

```
grid.best_params_
```

```
{'C': 166.81005372000593, 'penalty': 'l2'}
```

**Fig.16.Best Parameters of Logistic Regression from Grid Search**

So, the best value of C is **166.81** and penalty is '**L2**'.
Let's now see the score obtained by the model with optimum parameters –

```
grid.best_score_
```

```
0.7641509433962265
```

**Fig.17.Best Score of Logistic Regression from Grid Search.**

Voila! we found the best parameter which we can use for modelling our Logistic Regression Model and from **71%** accuracy we can reach **76%** accuracy.

## V. IDENTIFYING THE BEST SOLUTIONS FOR THE TICKET

In the previous chapter we learnt about how to get the best suited category for the filed incident. In this chapter, we will focus on identifying the best possible solutions for the incident using natural language processing using NLTK's corpus reader library, '**WordNet'**.

### A. Introduction to WordNet

WordNet is the outcome (a large lexical database of English) of the research done in Princeton University. In WordNet, nouns, verbs, adverbs and adjectives are organized by a variety of semantic relations into synonym sets called *synsets*, which represent one concept. WordNet can also be taken as a semantic dictionary of words, interlinked by semantic relations. Here, we are going to use Summary column from the data set and will try to identity the incident with the summary like the summary of the filed incident.

### B. How WordNet Works?

A basic idea on how WordNet works is as follows:

- It organizes information in a hierarchy.
- Verbs, Nouns, adjectives, all have their own hierarchies.
- Different Similarity metrics are applied on these hierarchies to get the score signifying the similarity between the two concepts.

Example –
Let's try to find out the semantic similarity between deer and elk. So, the possible tree that can related the above two concepts can be as below:

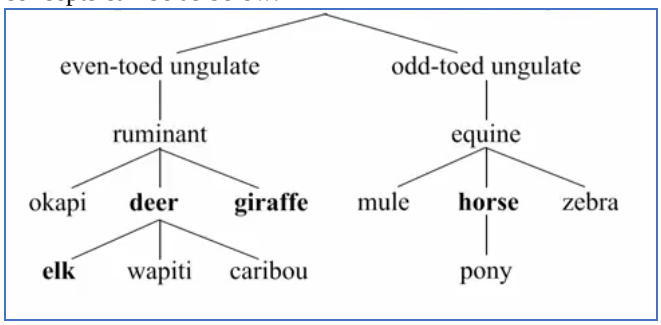


**Fig.18.Sample Semantic Hierarchy in WordNet**

We can get multiple information from the above tree, like,

- Elk, Wapiti and Caribou are types of deer.
- Pony is a type of Horse.
- Deer and Giraffe are siblings.
- Deer and Horse are ungulates.

### C. Different Semantic Similarity Metrics

*Path Similarity*

• The idea here is to find the shortest path between the two concepts in the hierarchy.

• Similarity measure is inversely related to path distance. i.e.

**Score = 1 / (1 + Path Distance)**

Example, Similarity between Deer and Elk – 1 / (1 + Distance (Elk, Deer)

= 1 / (1+1)

= 0.5

*Lin Similarity Lowest Common Subsumer*

• The idea here is to find the closest ancestor to both the concepts.

Example,

LCS (Deer, Elk) = Deer (Closest Ancestor here is Deer)

We can use LCS concept in calculating Lin Similarity which is based on the information contained in the LCS of two concepts i.e.

**LinSim(u, v) = 2 x log P(LCS(u, v)) / (log P(u) + log P(v))**

In this study, we will keep our focus on Path Similarity and other metrics can be taken as try-outs for future.

A python notebook was created to get the path similarity between the two texts. Sample output from the notebook –

```
document_path_similarity("I am a deer","I am an elk")
0.6488636363636364
```

**Fig.19.Sample output from notebook to calculate Path Similarity between two documents/strings.**

The path similarity between the two strings, “I am a deer” and “I am an elk” is 0.64.

## VI. INTEGRATING LOGISTIC REGRESSION CLASSIFIER WITH MODEL SUGGESTING BEST POSSIBLE SOLUTION OF THE INCIDENT

For better availability and accessibility of the model, Flask based API is created which takes incident title and small description as input and performs data pre-processing on the input title before passing it to the model for classification and further for recommending the possible solution for the incident.

Process of the request execution to classify the incident and suggest the solution for the ticket:

- Dump the logistic regression model via pickle library.
- Use the dumped model to classify the incident.
- Once the incident is classified, send the predicted category and the input summary to the language model.
- The language model gets all the incident of the predicted category and calculates the path similarity of the input summary with the already existing incidents.
- Consider the solution given to the incidents where the path similarity matches the threshold as the solution for the input incident.

API with both classification model and ticket solution recommender model –

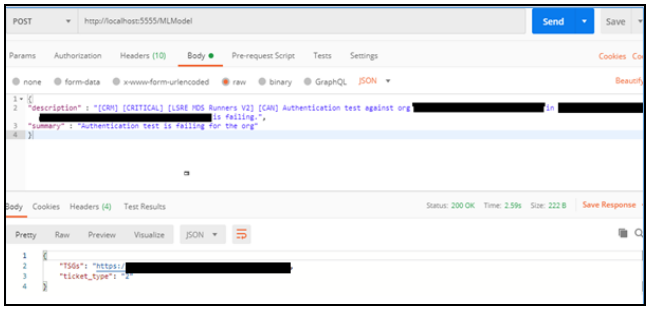

**Fig.20.Output from the API.**

*API architecture explanation* –

- Input Properties –

  **Description** – Incident title with details about what failed or what is not working.

  **Summary** – A brief description of the problem.
- Output –

  **Ticket_type** – The category of the ticket to which incident belongs to.

  **TSGs** – List of TSGs (Trouble Shooting Guide) based on the already resolved incidents.

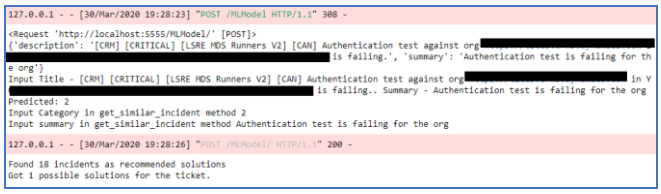

**Fig.21.Logs from the API**

## VII. RESULT AND DISCUSSION

For our problem where we must deal with lot of text data and which has lot of inconsistency in the data, machine learning modelling becomes more challenging. Hence below is the solution that we are proposing: -

Classification Process –

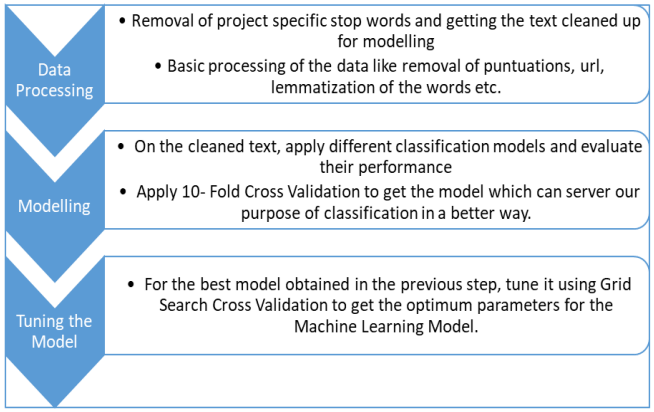


**Fig.22.Flow diagram showing the classification process used in the current work.**

Finding Solution for the ticket using Path Similarity Metrics-

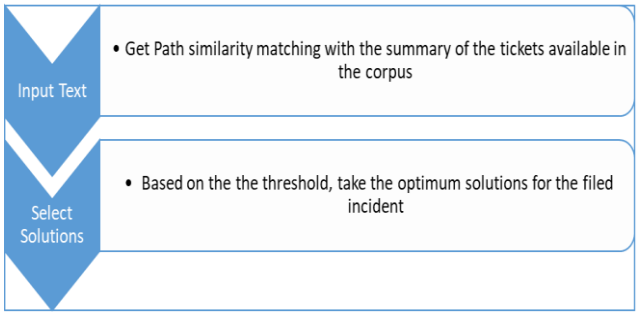


**Fig.23.Flow diagram showing the process used in finding the best solution for the ticket.**

On following the above process to solve our problem, it is found that out of all the algorithms, Logistic Regression is the best performing model, and we can fine tune it to achieve the better performance somewhere near to what we obtained after applying 10-Fold Cross Validation.
Accuracy without tuned model : - **71%**
Accuracy after tuning the model : - **76%**
The path similarity model to find the similarity between the texts is useful in getting the possible solution for the tickets.
Both the models are combined into one single framework and exposed via API where classifier classifies the incident and language model suggests the possible solution of the input incident.

## VIII. DIRECTIONS FOR FURTHER IMPROVEMENTS

- Deep Learning techniques can be tried on the data set to see if they give better classification results.
- Different other text similarity metrics can be tried for the language model.
- Improving the NLP techniques in finding solution of the tickets using word embedding, POS tagging etc.

## IX. CONCLUSION

The proposed work demonstrates how machine learning can be leveraged in solving a business problem to classify and recommend possible solution for the ticket. The work explains the methodology on how to prepare the dataset so that it is ready for applying machine learning algorithms. The work also focusses on the results obtained by applying different classification algorithms on the dataset and how the performance can be improved by using Hyper Parameter Tuning. Finally, after performing all the steps, it is found that the logistic regression combined with path similarity language model is the go-to solution for our problem. The directions of further improvements are also stated which can improve this baseline model.

## REFERENCES

1. C.D. Manning, P. Raghavan and H.Schutze, Cambridge University, Introduction to Information Retrieval, 2008
2. Giridhar N S, Prema K.V, N .V Subba Reddy, A Prospective Study of Stemming Algorithms for Web Text Mining
3. Tan P. N., Steinbach M & Kumar V., Introduction to Data Mining Pearson Education, 2006,
4. Tom M. Mitchell, Machine Learning, The McGraw-Hill Companies, Inc. International Edition 1997.
5. David M Blei, Andrew Y Ng, Michael I Jordon,Latent Dirichlet Allocation, Journal of Machine Learning Research, 2003.
6. Thabet Slimani, Description and Evaluation of Semantic similarity Measures Approaches, Taif University & LARODEC Lab

## AUTHORS PROFILE

**Himanshu Bajpai,** is working as Technology Analyst at Infosys Pune. His major research domains are Natural Language Processing and is avid solver of real-world problems using Data Science concepts.