IT SERVICE TICKET MANAGEMENT

Parvathy Vysakh
Artificial Intelligence & Machine Learning
Lambton College
Ontario, Canada
parvathy001ps@gmail.com

Patricia Adolph
Artificial Intelligence & Machine Learning
Lambton College
Ontario, Canada
patriciaadolph1@gmail.com

Pooja Selby
Artificial Intelligence & Machine Learning
Lambton College
Ontario, Canada
poojaselby18@gmail.com

Suchithra Chandrasekharan
Artificial Intelligence & Machine Learning
Lambton College
Ontario, Canada
suchitra596@gmail.com

Abstract*— IT service ticketing system can be implemented in all the major organizations irrespective of industrial sectors like IT, healthcare, retail, finance, marketing, etc. It is obvious that businesses capable of keeping up with trends and latest technologies like Machine Learning models are preferred by the clients. Service ticket data grows at an exponential rate, and handling them manually is a tedious task. We have chosen the IT sector for the analysis of service tickets as they play a major role in the software industry. A help desk system that serves as a single point of contact for both users and IT personnel. This paper offers a help desk system that may be deployed in a variety of sectors to serve as a single point of contact for users and IT professionals. It begins by associating a help desk ticket with the appropriate service using an accurate ticket classification machine learning model, lowering ticket resolution time, saving human resources, and boosting customer satisfaction. Because strong AI systems can learn, adapt, and decide what steps to take without the need for human intervention, using machine learning to classify tickets overcomes this problem. Natural language processing (NLP) allows machines to analyze, understand, and synthesize human language in a fast, consistent, and cost-effective manner, rather than humans analyzing and categorizing it. You could want to improve customer response times by efficiently routing tickets to the proper departments. Text classifiers can be used to categorize text. Tickets can be tagged by language, topic, or channel (Twitter, email, live chat, etc.) and sent to the appropriate team member or department. Let's imagine you've received a ticket from a consumer requesting a refund. This issue would be tagged as Refunds by a subject classifier and submitted to the accounts department. You may also utilize an urgency detection model, which can identify urgent tickets by analyzing their content for expressions like "right away," "immediately," or "as soon as possible," and then routing them to teams who deal with urgent problems. The model is created using an empirically developed process that consists of the steps below: generating training tickets, preprocessing ticket data, word stemming, feature vectorization, and machine learning algorithm tuning Nonetheless, the results of the experiments revealed that integrating the ticket comments and descriptions in the training data was one of the most important aspects in improving the model prediction accuracy.

Keywords—Ticket, prediction, preprocessing, KPI, ServiceNow, ConnectChat, TextBlob

I. INTRODUCTION

For large organisations that rely significantly on IT services and resources, an IT help desk system has become a must. This is because it acts as a single point of contact (SPOC) for IT employees and users in regards to requested services and reported concerns. It also allows for the automation of day-to-day IT tasks and the administration of IT tickets (e.g., assignment of tickets to service agents, email notifications for related parties, etc.). It can also assist in defining and minimising the activities that make up the various business processes. It also enables for the examination of the overall performance of the IT department using generated reports and analysed key performance indicators (KPIs). Not to add, implementing such software in a business boosts efficiency, improves service quality, and boosts customer satisfaction. In this regard, establishing a good support desk system necessitates achieving certain criteria:

• In-house development to provide a cost-effective, quick, and adaptable custom solution for all IT service needs.

• Integration with the employee portal, which allows users to access all online services from a single location using the same login credentials and with fewer system setup.

• Assist with all aspects of the IT issue management process, including opening, classifying, prioritizing, assigning, resolving, documenting, archiving, and closing.

• Allow collaborators to share comments and files about an incident, which makes users happy because they can talk to IT agents to figure out what's wrong or how to fix it. Provide email notifications for every ticket update to keep all parties informed.

• Ticket classification and routing to responsible IT personnel are automated to reduce ticket resolution time and improve customer satisfaction.

• Create reports to track the KPIs needed to assess the IT processes' health.

IT support Ticketing systems are software programs that help firms manage and streamline the process of resolving internal IT support issues. They have a lot of advantages for businesses. In a typical IT system, the majority of essential issues are handled and recorded through tickets. IT infrastructure is a collection of interconnected components in a network. As a result, when one component fails, it causes many other components to fail as well. Existing systems necessitate manual intervention, causing 30-40% of incident tickets to float about for days or weeks before being assigned to the right team, at which point the issue can be escalated to senior management. It could take days to discover out what went wrong and what caused it. As a result, it is critical to tell the component owners as soon as possible so that appropriate measures can be made to avoid this from happening again.

IT service tickets are usually reported by users to the service desk who are the first line of support to solve their issue. Unfortunately, it is difficult for anyone in service desk to be aware of all the components of the issue and hence tickets can be routed to the wrong team. Since this is a manual process, the process might even take days or weeks to reach the original team. When an issue or support ticket appears in your help desk, it needs to be processed and assigned a tag (or category) so that it's routed to the correct team member. This will either be manual, which involves reading the ticket and manually assigning a tag, or automatic, which involves setting up rule-based systems, which tend to follow the rule of 'IF X happens THEN do Y'. Manual classification systems are often complicated and cluttered, and support agents struggle to assign a category. After spending endless hours going through tickets, they'll often end up assigning the 'Other' tag to get through this tedious task faster. In order to deliver the best solution, the root cause of the occurrence must be recognized. Users frequently report these concerns to the hotline or Service Desk, which are the initial lines of support. Unfortunately, it is difficult for anyone to be aware of all of the components, and as a result, tickets are sent to the incorrect team. It could take days or weeks for it to reach the initial team. As a result, we reasoned that we could tackle the problem using natural language processing (NLP). IT Service Management generates more than 40 GB of data on average. This provides us with sufficient data to train and deploy our model. When effectively applied, our strategy might save business units billions of dollars by preventing them from losing many service level agreements. For example, how to effectively handle massive numbers of customer service inquiries received through various methods. Despite this, many organisations continue to disregard this customer service option. This is due to a misunderstanding of what a ticketing system is and how it may help a business.

The goal of this project is to use Machine Learning-based models to automate the IT service ticketing process. As a result of our proposed approach, we believe that costs will be lowered. It will take time for a company's help desk software team to learn how to use it after it is implemented. By reducing the complexity of the process with a machine learning-based solution, this learning curve can be decreased. We chose IT Service Management to solve a real-world business problem. We were particularly interested in two of the user cases that seemed to fit our project the best out of all the business cases.

Nearly 30–40% of incident tickets in Helpdesk are not routed to the correct team, and the tickets continue to float around, and by the time they reach the correct team, the problem may have spread and reached a large number of people. Let's pretend that people are having issues with their printers. The user contacts the help desk, files a ticket with IT Support, and they discover that they need to alter a configuration on the user's system, which they do before resolving the ticket. What if ten other people have the same problem? Using unstructured data, we can use this information to investigate the underlying problem. We can also examine the pattern and reduce the number of incident tickets.

Endava's internal data was utilised, which was imported from their helpdesk system. We were able to collect approximately 50k categorised support tickets, each with original user messages and labels already allocated. Data has been stripped of any sensitive information and anonymised for the purposes of this repository (encrypted).

The main objective of our project is to solve the real life business challenge in IT Service Ticket Management. We will start off this project with the motive of automating the IT Service Ticketing process where in reality, the real challenge involved is to manually identify and redirect the tickets to the appropriate team. The dataset consist we received from Endava consists of 48549 incident tickets with 27 fields. We will pull the dataset from ServiceNow with the id of python integration and classified those using ML techniques to make a labelled dataset. We will just keep the description and categories from the ticket and go ahead with analytics on it to come up with top 5 categories for incidents. The labels will be added after we classify the top 5 incident categories using Topic Modelling. The aforementioned steps will be possible only after undergoing the process of customized preprocessing of textual data like lower case conversion, stop word removal, tokenization, lemmatization, text vectorization and generation of a document word matrix. The classification of the incident tickets will be done rightly using the RNN algorithm, LSTM(Long Short Term Memory) model built using a Keras sequential model, compiled and fit to predict and classify into 5 topic classes. Subsequently, these labelled tickets will be properly assigned to the team members of the right team with the aim of maximizing the efficiency and profit using the linear sum assignment problem. We will also make sure to prioritize the incoming labelled tickets with the help of a Naïve Bayes Classifier using GridSearch method in order to show striking results on performance evaluation. As a final stage of the project the entire deployment process will be done using the flask framework and rest API calls from ServiceNow platform and the result of classified tickets will be transferred to the ServiceNow platform.

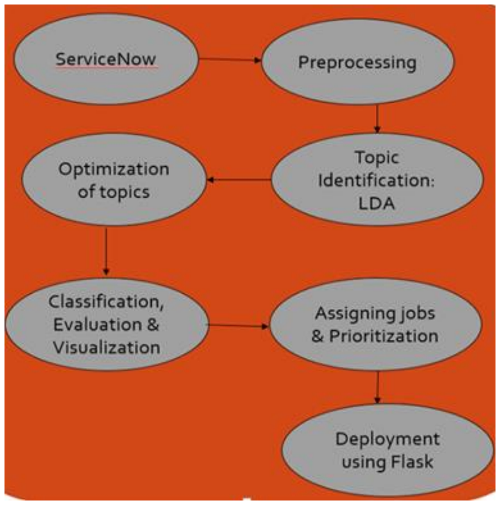

ServiceNow
Preprocessing
Optimization of topics
Topic Identification: LDA
Classification, Evaluation & Visualization
Assigning jobs & Prioritization
Deployment using Flask

Fig. 1. IT Service Ticket Management System

II. Data Gathering & Exploration

To better analyse trends inside incident tickets, we need to look at the data before selecting and training machine learning models. Service tickets are typically created using the ServiceNow platform. We can get data straight from the platform thanks to ServiceNow's strong Table API technology. Developers, administrators, platform owners, and service owners can use the Developer Program to learn, build, and deploy applications.

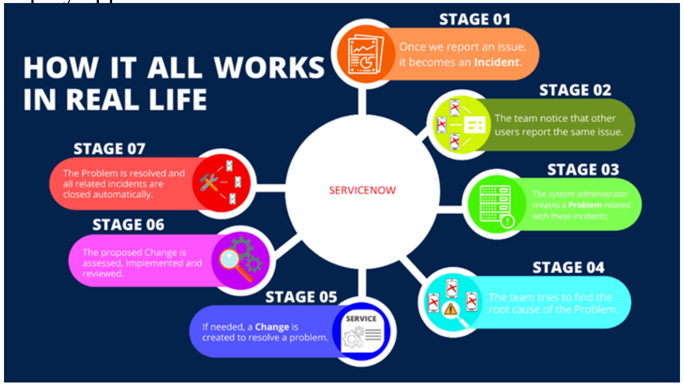

HOW IT ALL WORKS IN REAL LIFE
SERVICENOW
STAGE 01
Once we report an issue, it becomes an Incident.
STAGE 02
The team notice that other users report the same issue.
STAGE 03
The system administrator creates a Problem related with these incidents.
STAGE 04
The team tries to find the root cause of the Problem.
STAGE 05
If needed, a Change is created to resolve a problem.
SERVICE
STAGE 06
The proposed Change is assessed, implemented and reviewed.
STAGE 07
The Problem is resolved and all related incidents are closed automatically.

Fig. 2. ServiceNow Stages

III. Creation of Developer Account in ServiceNow

IT Service Ticket Management consolidates and automates service management processes, increase efficiency, lower cost. ServiceNow is a platform that is used by majority of the organizations to resolve their service tickets. To be able to develop a custom application, we need to have our own instance of the ServiceNow platform. We have created a developer account and requested a free instance by visiting the ServiceNow developer portal site at: https://developer.servicenow.com.

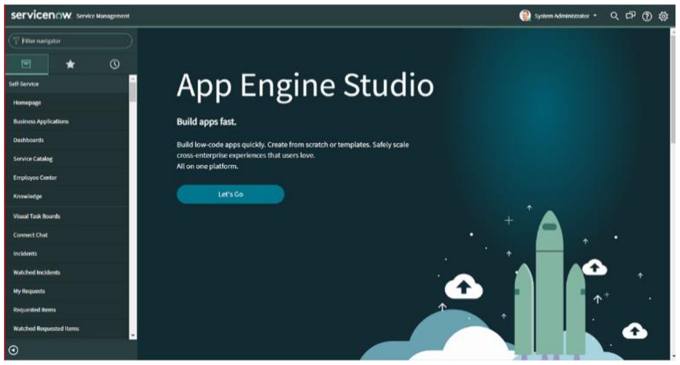


Fig. 3. ServiceNow Developer Account

The ServiceNow User Interface is depicted in the diagram below. As System Administrator, we are logged in. The following three elements make up its main screen: The ServiceNow logo is located in the left top corner of the banner. The Global Search Engine, Connectchat, Help Menu, and Settings are all located on the right. Navigator for Applications: A list of business applications and modules is available. Frame for content: It consists of many data formats such as Forms, Lists, and so on.

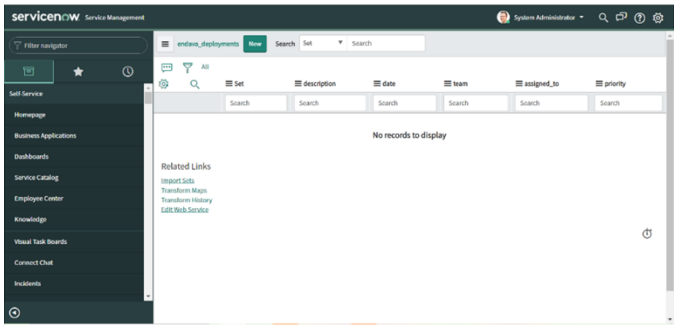
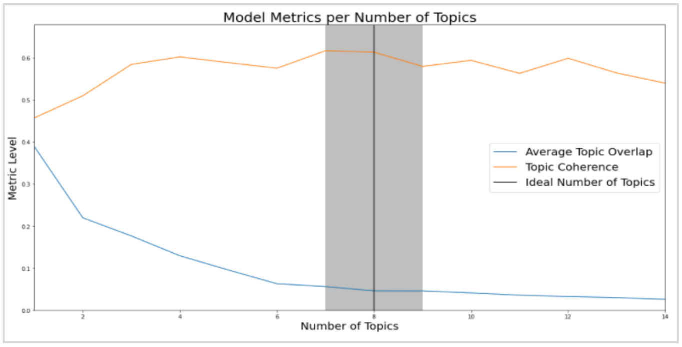


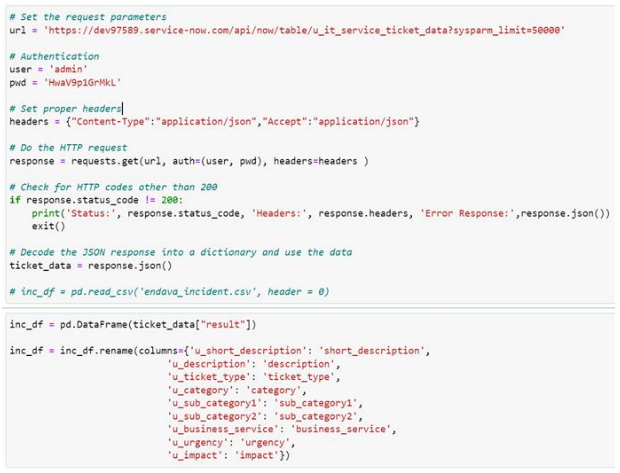
Fig. 4. ServiceNow Interface

IV. Dataset Description

After contacting several companies, we successfully managed to convince a UK based company named Endava that finally agreed to share their masked data by removing any sensitive information related to the company. The dataset consist we received from Endava consists of 48549 incident tickets with 27 fields The main problem we attempted to solve was to develop a model for automatically classifying support tickets for Endavas' helpdesk service. As Endava noted, helpdesk operators now waste a significant amount of time assessing tickets and attempting to give values to characteristics such as ticket type, urgency, impact, category, and so on for each submitted ticket.

Using the ServiceNow platform's developer account, I was able to effectively extract incident tickets using Python API calls. With the help of the pandas package, the JSON response from ServiceNow was converted into a data frame, which was then utilized for pre-processing. ServiceNow is a cloud-based IT Service Management (ITSM) software platform that aids in the automation of IT Business Management. It makes use of machine learning to help businesses become more efficient and scalable by leveraging data and workflows. We can get data straight from the platform thanks to ServiceNow's strong Table API technology. The incident tickets were pulled from the ServiceNow platform via API calls utilising the developer account that was created in the first week. The ServiceNow JSON response was decoded into a dictionary, and this data was then used in the preparation phase. Each incident ticket must be examined and investigated in order to determine the subject or root cause of the problem which in turn needs to be redirected to respective problem tacklers. Now that we have our dataset extracted from ServiceNow, our first initiative for the automation of the aforementioned process is to preprocess the vast amount of textual data so as to find the topics.





```
# Set the request parameters
url = 'https://dev97589.service-now.com/api/now/table/u_it_service_ticket_data?sysparm_limit=50000'

# Authentication
user = 'admin'
pwd = 'HwaV9p1GrMkL'

# Set proper headers
headers = {"Content-Type":"application/json","Accept":"application/json"}

# Do the HTTP request
response = requests.get(url, auth=(user, pwd), headers=headers )

# Check for HTTP codes other than 200
if response.status_code != 200:
    print('Status:', response.status_code, 'Headers:', response.headers, 'Error Response:',response.json())
    exit()

# Decode the JSON response into a dictionary and use the data
ticket_data = response.json()

# inc_df = pd.read_csv('endava_incident.csv', header = 0)
```

```
inc_df = pd.DataFrame(ticket_data["result"])

inc_df = inc_df.rename(columns={'u_short_description': 'short_description',
                                'u_description': 'description',
                                'u_ticket_type': 'ticket_type',
                                'u_category': 'category',
                                'u_sub_category1': 'sub_category1',
                                'u_sub_category2': 'sub_category2',
                                'u_business_service': 'business_service',
                                'u_urgency': 'urgency',
                                'u_impact': 'impact'})
```

Fig. 5. Importing Data from ServiceNow

The main objective of our project is to automate the IT service ticketing process which involves manual identification of the incident ticket topics and rerouting them to the appropriate team. In order to deliver the best solution, we find the root cause of the incident tickets using a technique called topic modeling. Topic Modeling is a technique to extract the hidden topics from a heap of text. Latent Dirichlet Allocation (LDA) is the primary algorithm for topic modeling with excellent implementations in the Python's Gensim package. We pull the dataset directly from servicenow and try to classify them using ML techniques to make a labelled dataset. We just kept the description and categories from the ticket and go ahead with analytics on it.

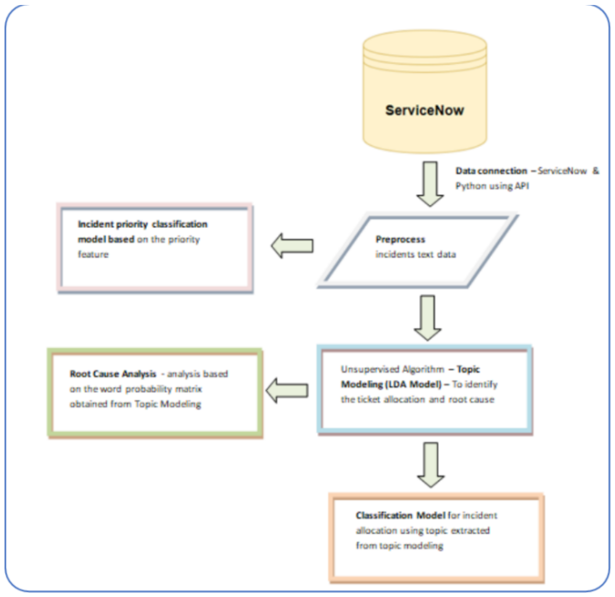

ServiceNow
Data connection – ServiceNow & Python using API
Preprocess incidents text data
Incident priority classification model based on the priority feature
Unsupervised Algorithm – Topic Modeling (LDA Model) – To identify the ticket allocation and root cause
Root Cause Analysis - analysis based on the word probability matrix obtained from Topic Modeling
Classification Model for incident allocation using topic extracted from topic modeling

Fig. 6. Model Workflow

V. DATA PREPROCESSING

Sentiment analysis of the Product Description corresponding to each category. This is done by measuring the polarity of each product's description. Sentiment Analysis on the Product Description was also performed by calculating the polarity with the help of the TextBlob library in Python. From the Seab We used the aid of python NLTK, Spacy and genism libraries for unsupervised topic modeling and natural language processing, using modern statistical machine learning. The first step in preprocessing was to remove stop words using NLTK library from the description feature of the incident ticket. In addition to this, emails, newline characters, single quotes and punctuations were also eliminated. The text so obtained was then lemmatized and the bigrams and trigrams were also generated for the tokens.

The lemmatized text is used to create a dictionary and corpus using which we find the term document frequency and finally convert into readable format of corpus. This was the input text we used to build the LDA model to extract good quality of topics that are clear, segregated and meaningful.

VI. TOPIC MODELING – TEAM IDENTIFICATION

To extract themes from a corpus, Latent Dirichlet Allocation (LDA) is a popular topic modelling technique. Topic modelling is a method for identifying hidden subjects in a large amount of text.Latent Dirichlet Allocation (LDA) is the primary algorithm for topic modeling with excellent implementations in the Python's Gensim package. Each incident ticket has to be analyzed to discover the topics or root cause of the issues in order to be redirected to respective problem resolvers. The first initiative for the automation of the aforementioned process is to preprocess the vast amount of

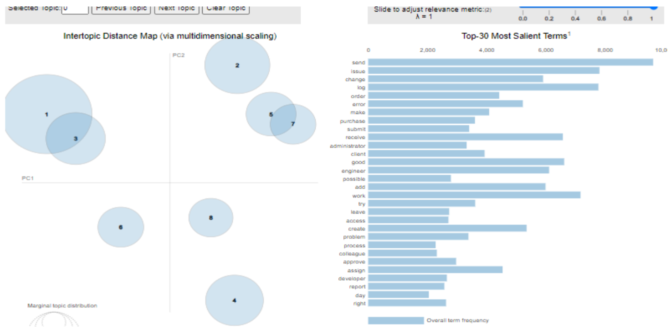


Fig. 7. TOPIC MODELLING(LDA USING GENSIM)

textual data so as to find the topics. Two approaches are used here: Gensim and Sklearn

The lemmatized text is used to create a dictionary and corpus using which we find the term document frequency and finally convert into readable format of corpus. This is the input text we used to build the LDA model to extract good quality of topics that are clear, segregated and meaningful. Since input to train the LDA model was corpus, dictionary and the number of topics, our initial attempt in creating LDA model using the topics as 20 failed in obtaining a desired topic coherence and perplexity (both of which provides a measure of how good a topic model is).

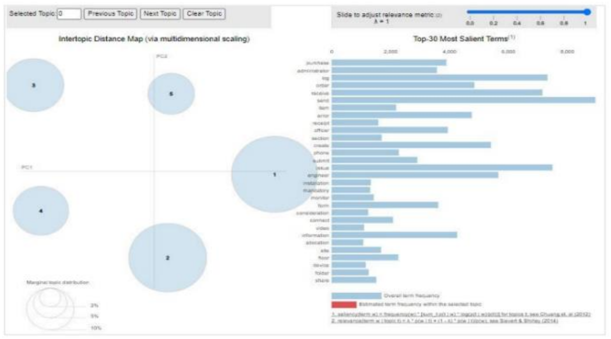
The best technique to identify the ideal number of topics is to develop several LDA models with different numbers of topics (k) and then choose the one with the highest coherence value. Choosing a 'k' that denotes the end of a rapid increase in subject coherence frequently yields themes that are meaningful and comprehensible. Choosing a higher number might occasionally result in more granular sub-topics. It's usually a hint that the 'k' is too large if you encounter the same terms in many topics.

VII. VISUALIZE AND OPTIMIZE THE TOPICS

After the LDA model has been developed, the following step is to review the generated topics and keywords.We need to optimize the total number of topics hence obtained. For finding the optimal number of topics, we run the LDA model for each number of topics from 1-15 and use Jaccard similarity to find the mean stabilities and coherence for each iteration. The maximum value of the coherence statistics gives the ideal number of topics. In order to have a better understanding we visualize the aforementioned steps to find out that 8 is the optimal number of topics. There is no better tool than the interactive chart in the pyLDAvis package, which is built to operate with jupyter notebook.

VIII. ADVANCED PREPROCESSING

The quality of topics is heavily dependent on the quality of preprocessing done and hence we found the need of removing some stop words from the incident ticket description for a much



more refined processing and classification. Each description of the incident tickets were converted into lower case and also removal of non-consecutive duplicates were done. It was found that the description column had some kind of misleading words that contributed to biased behavior due to which the topic segregation failed to show promising results. Some of these words are less relevant words or redundant words like Regards, Thank you etc. The ideal topic number hence obtained from Jaccard similarity was used to build a final LDA model to obtain a coherence score of 61 and Perplexity of -7.1714. Since these measures fall under the required band, we proceeded with the optimal topic number as 8, compared to a lower and undesirable score for these when the topics was 20.

IX. TOPIC MODELLING: OPTIMIZATION

A. Text Vectorization

The LDA topic model algorithm requires a document word matrix as the main input so the lemmatized text was then used to first initialized the CountVectorizer class with the required configuration and then applied fit_transform to actually create the matrix. We configured the CountVectorizer to consider words that has occurred at least 10 times (min_df), remove built-in english stop words, convert all words to lowercase, and a word can contain numbers and alphabets of at least length 3 for to be qualified as a word.

B. Building LDA Models Using Sklearn

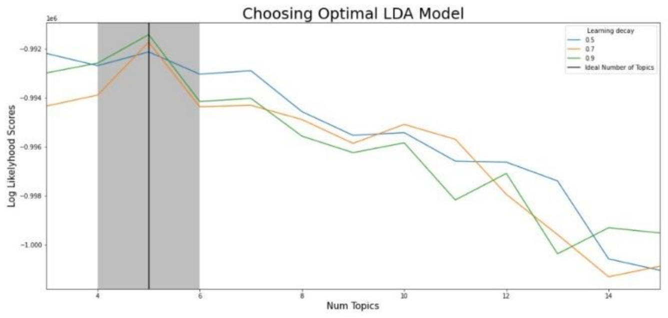


Fig. 8. TOPIC MODELLING(LDA USING SKLEARN)

While using GridSearch to find the optimal number of topics, we built 12 models each for .5, .7 and .9 learning decay (a parameter that control learning rate in the online learning method. The value should be set between (0.5, 1.0] to guarantee asymptotic convergence) search parameters which implies a total of 36 models. Lack of multicores in LDA using sklearn did affect the execution steps and to complete the modelling part and to obtain the best model among the 36 models, it took us almost 12 hours. The next task was to fit the models using the vector generated in the previous step and then the best model to find the optimal topics was picked using the log_likelihood (this was plotted graphically represented as well). Minitab optimizes the log-likelihood expression to find the best values for the estimated coefficients (). Because log-likelihood values are a function of sample size, they cannot be used as an index of fit by themselves, but they can be used to compare the fit of different coefficients. The higher the value, the better for maximizing log-likelihood. A log-likelihood value of -3, for example, is preferable than -7.

C. Finding the best model

We use perplexity and log-likelihood to diagnose model performance. A good model has a greater log-likelihood and a lower perplexity (exp(-1. * log-likelihood per word)). Because it ignores the context and semantic relationships between words, perplexity may not be the optimal metric for evaluating topic models. For time efficiency, Sklearns best estimator_ is used to select the best LDA model, which is then saved as joblib (used when the model comprises huge estimators).

D. Dominant topic in each document

To classify a document as belonging to a particular topic, we check which topic had the highest contribution to that document and assign it. To accomplish this we created the Document-Topic Matrix and transformed the best model. Now that we have the dominant topics, we assign labels and for each topics, we found the most frequent 15 words and its probability distribution. After visualization the result, we found that there were no overlap of words (indicates uniqueness).

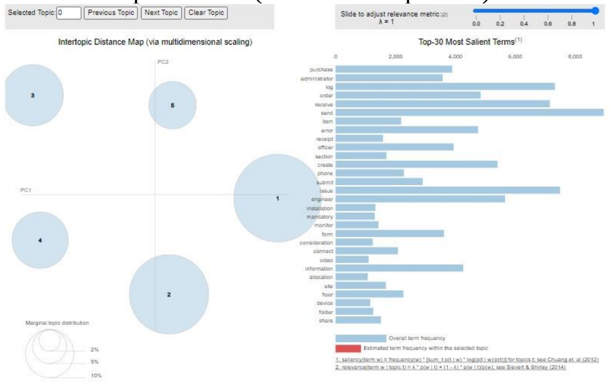
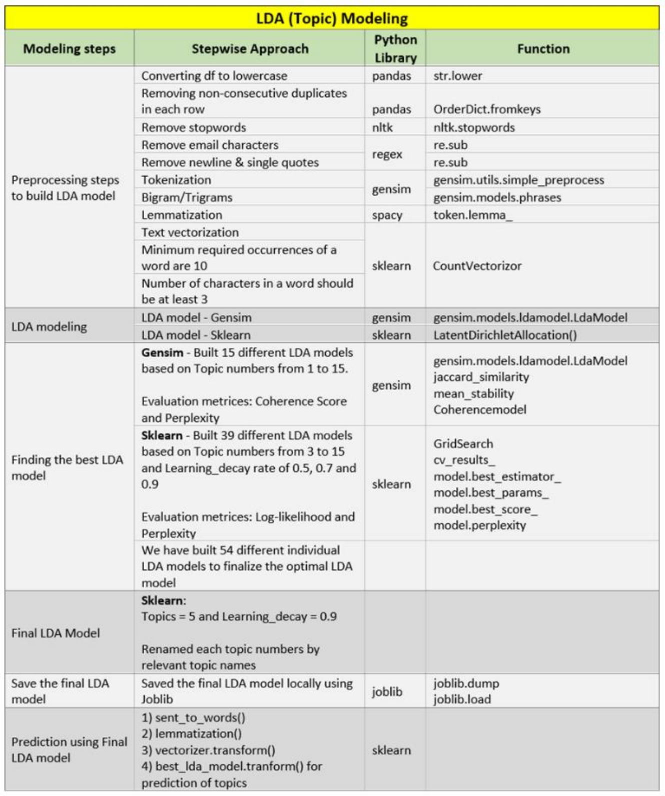

Selected Topic: 0
Previous Topic
Next Topic
Clear Topic
Slide to adjust relevance metric:(2)
λ = 1
0.0
0.2
0.4
0.6
0.8
1
Intertopic Distance Map (via multidimensional scaling)
PC2
PC1
1
2
3
4
5
Marginal topic distribution
2%
5%
10%
Top-30 Most Salient Terms(1)
0
2,000
4,000
6,000
8,000
purchase
administrator
log
order
receive
send
item
error
receipt
officer
section
create
phone
submit
issue
engineer
installation
mandatory
monitor
form
consideration
connect
video
information
allocation
site
floor
device
folder
share
Overall term frequency
Estimated term frequency within the selected topic
1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

Fig. 9. Dominant Topics



LDA (Topic) Modeling			
Modeling steps	**Stepwise Approach**	**Python Library**	**Function**
Preprocessing steps to build LDA model	Converting df to lowercase	pandas	str.lower
	Removing non-consecutive duplicates in each row	pandas	OrderDict.fromkeys
	Remove stopwords	nltk	nltk.stopwords
	Remove email characters	regex	re.sub
	Remove newline & single quotes		re.sub
	Tokenization	gensim	gensim.utils.simple_preprocess
	Bigram/Trigrams		gensim.models.phrases
	Lemmatization	spacy	token.lemma_
	Text vectorization	sklearn	CountVectorizor
	Minimum required occurrences of a word are 10		
	Number of characters in a word should be at least 3		
LDA modeling	LDA model - Gensim	gensim	gensim.models.ldamodel.LdaModel
	LDA model - Sklearn	sklearn	LatentDirichletAllocation()
Finding the best LDA model	**Gensim** - Built 15 different LDA models based on Topic numbers from 1 to 15. Evaluation metrices: Coherence Score and Perplexity	gensim	gensim.models.ldamodel.LdaModel jaccard_similarity mean_stability Coherencemodel
	Sklearn - Built 39 different LDA models based on Topic numbers from 3 to 15 and Learning_decay rate of 0.5, 0.7 and 0.9 Evaluation metrices: Log-likelihood and Perplexity	sklearn	GridSearch cv_results_ model.best_estimator_ model.best_params_ model.best_score_ model.perplexity
	We have built 54 different individual LDA models to finalize the optimal LDA model		
Final LDA Model	**Sklearn**: Topics = 5 and Learning_decay = 0.9 Renamed each topic numbers by relevant topic names		
Save the final LDA model	Saved the final LDA model locally using Joblib	joblib	joblib.dump joblib.load
Prediction using Final LDA model	1) sent_to_words() 2) lemmatization() 3) vectorizer.transform() 4) best_lda_model.tranform() for prediction of topics	sklearn	

Fig. 10. LDA-Topic Modeling steps

X. CLASSIFICATION MODEL BUILDING

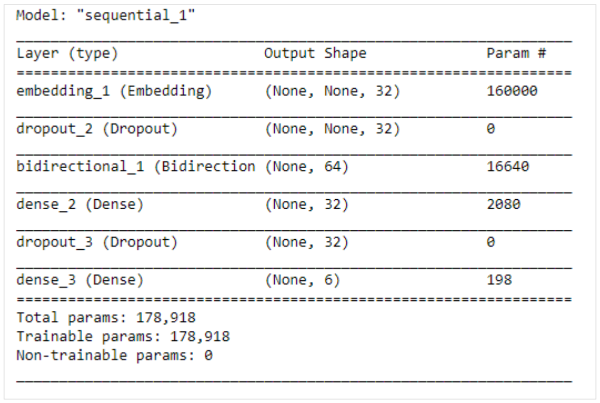
Long Short Term Memory networks, or "LSTMs," are a type of RNN that can learn long-term dependencies. LSTMs are specifically developed to prevent the problem of long-term dependency. They don't have to work hard to remember knowledge for lengthy periods of time; it's nearly second nature to them. We imported the libraries, including TensorFlow, and divided the entire dataset into training and validation sets, 80 percent for training and 20% for validation, according to the hyperparameters established. Preprocessing is the next step, which includes tokenizing and lemmatizing. When an unseen word is encountered, such as a word not in the word index, oov token is used to insert a specific value. fit on text scans all the text and generates a dictionary. Following tokenization, the tokens were converted into sequence lists. We started with an embedding layer after creating a tf.keras.Sequential model. An embedding layer saves information one vector per word. It turns sequences of word indices into vector sequences when invoked. Words with comparable meanings generally have similar vectors after training. The Bidirectional wrapper was used in conjunction with an LSTM layer to transport the input forwards and backwards through the LSTM layer before concatenating the outputs. This aids the LSTM's learning of long-term dependencies. After that, we used it to train a dense neural network for classification. We substituted relu for the tahn function because they are excellent substitutes. A Dense layer with 6 units and softmax activation was added. Softmax turns output layers into a probability distribution when we have multipled outputs.

We imported the libraries, including TensorFlow, and divided the entire dataset into training and validation sets, 80 percent for training and 20% for validation, according to the hyper parameters set. Preprocessing is the next step, which includes tokenizing and lemmatizing. When an unseen word is met, i.e. for words not in the word index, Oov token is used to store a particular value. The 'fit on text' function scans the entire document and creates a dictionary. Following tokenization, the tokens were converted into sequence lists. We started with an embedding layer and developed a tf.keras.Sequential model. One vector each word is stored in an embedding layer. It turns sequences of word indices into vector sequences when invoked. Words with comparable meanings generally have similar vectors after training. A LSTM layer was utilised with the Bidirectional wrapper, which propagates the input forwards and backwards through the LSTM layer and then combined the outputs. This aids the LSTM's learning of long-term dependencies. After that, we used it to train a dense neural network for classification. We substituted relu for the tahn function because they are excellent substitutes. A dense layer with 6 units and softmax activation was added. Softmax converts output layers into a probability distribution after they have been multiplied.

XI. VISUALIZATION AND EVALUATION

After compiling and fitting the model, the accuracy of the model is visualized as shown below which shows the problem of over fitting does not affect our model. Furthermore, the confusion matrix against the actual and predicted labels for 'Code', 'Connectivity', 'Permission', 'Administrative' and 'Network' depicts a precision, recall and f1 score between 80-95% which implies that our model is successfully able to categorize and route the incident tickets to the appropriate team (label assignment). Overall our model has an accuracy of 91% which is pretty good.

The following are the model summary and evaluation metrics:



```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, None, 32)          160000
_________________________________________________________________
dropout_2 (Dropout)          (None, None, 32)          0
_________________________________________________________________
bidirectional_1 (Bidirection (None, 64)                16640
_________________________________________________________________
dense_2 (Dense)              (None, 32)                2080
_________________________________________________________________
dropout_3 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_3 (Dense)              (None, 6)                 198
=================================================================
Total params: 178,918
Trainable params: 178,918
Non-trainable params: 0
_________________________________________________________________
```

Fig. 11. Classification Model – LSTM

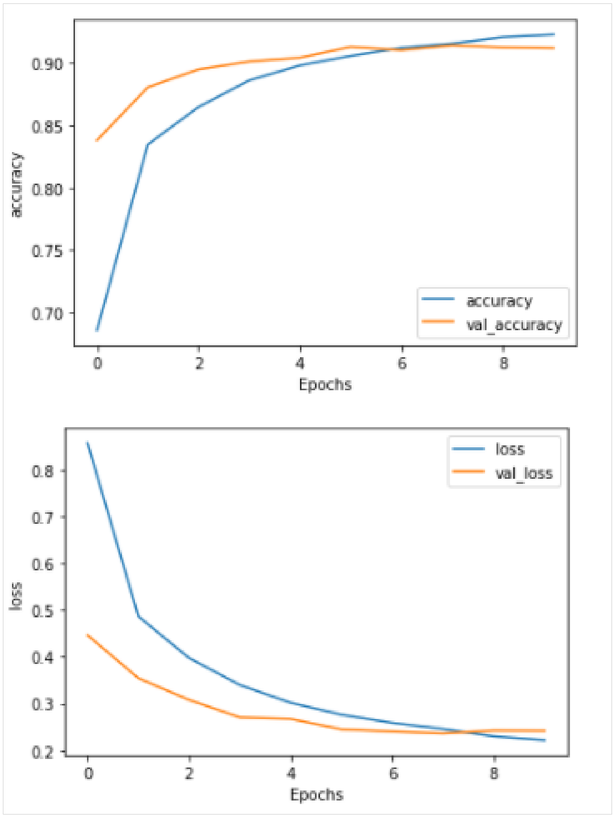
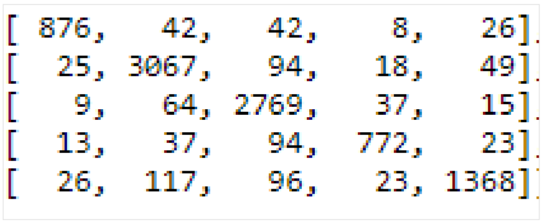
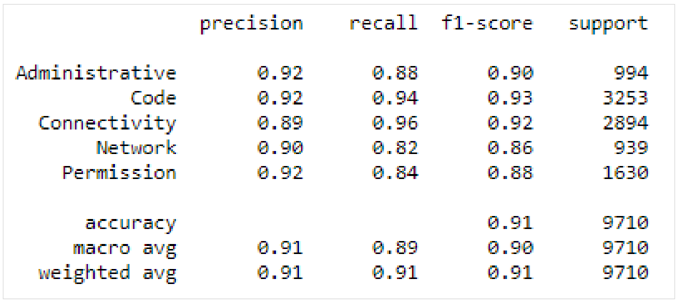


Fig. 12. Evaluation Of Classification Model



```
[ 876,   42,   42,    8,   26]
[  25, 3067,   94,   18,   49]
[   9,   64, 2769,   37,   15]
[  13,   37,   94,  772,   23]
[  26,  117,   96,   23, 1368]
```

Fig. 13. Confusion Matrix - Row * Column = Actual * Predicted



```
                precision    recall  f1-score   support

Administrative       0.92      0.88      0.90       994
          Code       0.92      0.94      0.93      3253
  Connectivity       0.89      0.96      0.92      2894
       Network       0.90      0.82      0.86       939
    Permission       0.92      0.84      0.88      1630

      accuracy                           0.91      9710
     macro avg       0.91      0.89      0.90      9710
  weighted avg       0.91      0.91      0.91      9710
```

Fig. 14. Confusion Matrix

XII. JOB SCHEDULING

Following the successful classification of incident tickets, another non-trivial job of our project is efficiently assigning the assignment to team members. "How should the assignments be made in order to optimize the specified objective?" to put it another way. The goal is to allocate a set of resources to an equal number of activities in order to reduce overall cost or increase total profit. Because available resources such as workers, machines, and other resources have differing degrees of efficiency for doing different activities, the cost, profit, or loss of completing the various operations varies.

Assume that there are n jobs to be done and that n people are available to complete them. Let cij be the cost if the i-th worker can accomplish each job at the same time, however with differing degrees of efficiency, let cij be the cost if the i-th person is assigned to the j-th job. The objective is to discover an assignment (which work should be assigned to which individual on a one-to-one basis) that minimizes the overall cost of doing all activities. This type of problem is known as an assignment problem.

$$\begin{array}{ll|lllll|}
 & & \multicolumn{5}{c}{\textit{Jobs}} \\
 & & 1 & 2 & 3 & j & n \\
 & 1 & C_{11} & C_{12} & C_{13} \cdots\cdots & C_{ij} \cdots & C_{1n} \\
 & 2 & C_{21} & C_{22} & C_{23} \cdots\cdots & C_{2j} \cdots & C_{2n} \\
 & 3 & C_{31} & C_{32} & C_{33} \cdots\cdots & C_{3j} \cdots & C_{3n} \\
\textbf{Persons} & \vdots & & & & & \\
 & i & C_{11} & C_{12} \cdots & C_{13} \cdots\cdots & C_{Cj} \cdots & C_{in} \\
 & n & C_{n1} & C_{n2} \cdots & C_{n3} \cdots\cdots & C_{nj} \cdots & C_{nn}
\end{array}$$

Fig. 15. ASSIGNMENT PROBLEM – JOB ALLOCATION

A fundamental combinatorial optimization problem is the assignment problem. There are several agents and tasks in this problem instance. Any agent can be assigned to any task, which comes at a cost that varies depending on the agent-task assignment. Because available resources such as workers, machines, and other resources have differing degrees of efficiency for doing different activities, the cost, profit, or loss of completing the various operations varies. In our project, incoming incident tickets are now labelled and grouped into five separate teams; as a result, these labelled tickets must be assigned to team members as efficiently as feasible. We created a prototype model for the assignment problem using system logic the week before.

In bipartite graphs, the linear sum assignment problem is also known as minimal weight matching. A matrix C describes a problem instance, A troublesome instance is C[i,j]. where C[i,j] represents the cost of matching vertex I of the first partite set (a "worker") with vertex j of the second partite set (a "task"). The goal is to assign all workers to low-cost jobs in a complete manner.

Let X be a boolean matrix, and if row I is assigned to column j, X[i,j]=1. The best assignment therefore has a cost of minijCi,jXi,j. Each column is assigned to a maximum of one row, and each row is assigned to a maximum of one column. The Hungarian algorithm, also known as the Munkres or Kuhn-Munkres algorithm, is the technique utilized.

Parameters:

cost matrix is an array that is used to calculate the cost of a product. (A bipartite graph's cost matrix.)

Returns: row ind, col ind: array (An array of row indices and one of the matching column indices that gives the best result. cost matrix [row ind, col ind] is a formula for calculating the assignment's cost. sum(). The indices for the rows will be sorted.in the case of a square cost matrix they will be equal to numpy.arange(cost_matrix.shape[0]).

PREDICT THE PRIORITY OF THE TICKETS

After the team has been assigned jobs, the next critical task is to anticipate the priority of each of the incident tickets. We employ the Naive Bayes (NB) algorithm, which is a supervised machine learning approach that relies on labelled input data that is divided into classes to predict the classification of a query sample. The ticket priority is what we predict in a project. The name naive comes from the fact that the method is based on the assumption of feature independence, and bayes comes from the fact that the algorithm employs the Bayes Theorem, a statistical classification technique

Step 1: For given class labels in training data, calculate the Prior Probability.

Step 2: For each class, calculate the Likelihood Probability for each feature attribute.

Step 3: Using Bayes' Theorem, calculate posterior probability.

Theorem of Bayes Equation

P(A|B) — the likelihood of event A occurring if event B has already occurred. [Posterior Probability] is a term that refers to the probability of something happening after it

P(B|A) — the likelihood of event B occurring if event A has already occurred. [Probability of Likelihood]

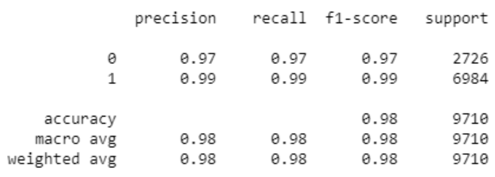
P(A) [Prior Likelihood of A] – the probability of occurrence A

P(B) [Prior Likelihood of B] — the probability of occurrence B

Step 4: Return the query sample's class label with a greater Posterior Probability forecast.

We divide each text file into words (for English splitting by space), count the number of times each word appears in each document, and then assign an integer id to each word. Each word in our lexicon will be associated with a functionality (descriptive feature). 'CountVectorizer' is a high-level component in Scikit-learn that creates feature vectors. We learn the vocabulary dictionary by using CountVectorizer, which returns a Document-Term matrix. [n features, n samples] The training data was then fed into a data processing pipeline, which was then fed into the model. By constructing a pipeline, we can write less code while accomplishing all of the above. The pipeline class in Scikit-learn is handy for encapsulating numerous distinct transformers and an estimator into a single object, so you only have to call your relevant methods once (fit(), predict(), and so on). GridSearch is used to set the selected parameters to find an optimal combination of hyperparameters that minimizes a predefined loss function to give better results.

We finally find the Score and evaluate model on test data using model without hyperparameter tuning and also using GridSearch. The classification report for naive bayes using grid search shows and accuracy of 98%.



```
              precision    recall  f1-score   support

           0       0.97      0.97      0.97      2726
           1       0.99      0.99      0.99      6984

    accuracy                           0.98      9710
   macro avg       0.98      0.98      0.98      9710
weighted avg       0.98      0.98      0.98      9710
```

Fig. 16. Classification report for Naïve Bayes Algorithm

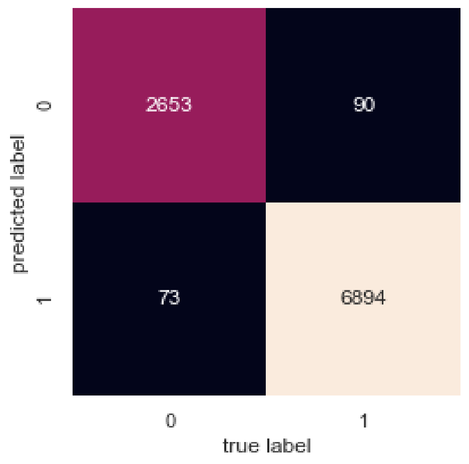


Fig. 17. Confusion Matrix

XIII. DEPLOYMENT

INTEGRATION BETWEEN FLASK AND SERVICENOW

Flask is a Python web application framework with many modules that make application deployment easier by giving us a number of options for constructing web applications and providing us with the required tools and libraries to build a web application. The incident tickets from Endava's ServiceNow are pulled via Python integration and used by our backend application Flask in our project. Flask APIs receive Incident Ticket information via GUI or API requests, computes and returns the anticipated value based on our model, and makes the results available to the ServiceNow platform (frontend). The key benefit of utilising Flask is its cost-effectiveness as well as the ease with which it can be deployed.

The steps involved in the deployment process are as follows:

- Extract the incident tickets from ServiceNow platform using API calls.
- Convert the extracted ticket into a data frame for processing in Python.
- Use the saved machine learning models for prediction by allocating the incident ticket to the right team, assigning the ticket to the responsible team member, and identifying the priority of the ticket.
- Post the result back to ServiceNow platform for solving the ticket within the stipulated time frame.

The flask framework and rest API calls from the ServiceNow platform are used throughout the deployment process. With the help of selenium, the server is refreshed every 5 seconds. When a user initially generates an incident ticket, the user enters a description and the status is set to 'Open.' The status will be changed to 'Pending' once the allocation is completed, guaranteeing that the ticket assignment is accomplished. As long as the server is up and running, the process is automated whenever a new ticket is created. The deployment procedure concludes with the transfer of classified tickets to the ServiceNow platform.

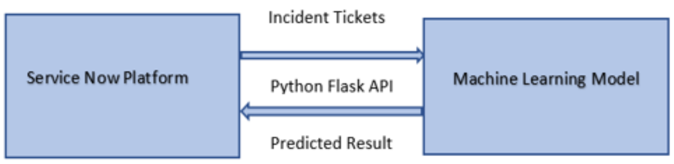
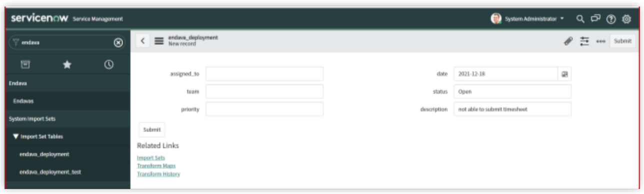


Fig. 18. Deployment Process

XIV. RESULTS

We began our project with the aim to unravel the genuine issue in IT Service Ticket Administration. The method now involves the manual identifying the tickets and after that diverting it to the fitting group which devours time and cash. To automate the method, we have developed a machine learning model which classifies the incident tickets and categorizes it into 5 categories. The classification was accomplished utilizing LSTM (Long Short-Term Memory) model with Keras sequential model and fit to foresee and classify into 5 teams.

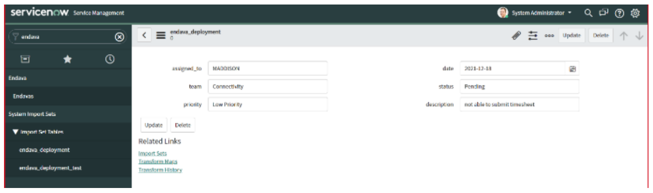
Before this process, the data had undergone multiple pre-processing steps like lower case conversion, stop word removal, tokenization, lemmatization, text vectorization and generation of a document word matrix. The next step was to dole out the named tickets to the group individuals using the linear sum assignment problem. Moreover, we prioritized the named tickets utilizing Naïve Bayes Classifier strategy to create the tickets settled as early as conceivable based on the needs. We made beyond any doubt that all the assignments gave us an extraordinary result with exact outcomes. We started the process of deployment using flask and integrating it with ServiceNow using API calls. This is one of the crucial steps in our entire project as the deployment involves extracting tickets from ServiceNow and using these tickets for preprocessing, classification, team assignment and prioritization and finally passing the predicted results back to ServiceNow using Flask API calls. The task outlined for the past week was model deployment in flask and publish results back to ServiceNow platform.



endava_deployment
New record

assigned_to		date	2021-12-18
team		status	Open
priority		description	not able to submit timesheet

Fig. 19. Result Interface 1



endava_deployment

assigned_to	MADDISON	date	2021-12-18
team	Connectivity	status	Pending
priority	Low Priority	description	not able to submit timesheet

Fig. 20. Result Interface 2

Created an incident sample in ServiceNow as "Not able to submit timesheet". The incident creation date and the status of the ticket is set manually and the ticket is submitted in ServiceNow. The model has accurately predicted the issue as "Connectivity", the resource as "Maddison" and the priority as "Low".

XV. Conclusions and Future Work

The main objective of our project was to automate the IT service ticketing process which involves manual identification of the incident ticket topics and rerouting them to the appropriate team.

We were able to create a ServiceNow developer account initially using which we were able to successfully extract the company's IT service tickets using Table API framework provided by ServiceNow. Our dataset contains information about 48549 incident tickets each bearing 27 features including short_description, description, ticket_type, etc. In order to deliver the best solution, we used the aid of python NLTK, Spacy and genism libraries for unsupervised topic modeling and natural language processing, using modern statistical machine learning.

The first step in preprocessing was to remove stop words using NLTK library from the description feature of the incident ticket. In addition to this, emails, newline characters, single quotes and punctuations were also eliminated. The text so obtained was then lemmatized and the bigrams and trigrams were also generated for the tokens. The lemmatized text was used to create a dictionary and corpus using which we found the term document frequency and finally converted into readable format of corpus. This was the input text we used to build the LDA model to extract good quality of topics that are clear, segregated and meaningful. We identified the root cause of the incident tickets using a technique called topic modeling to extract the hidden topics from a heap of text. In addition to this we performed optimization task in order to find the optimal number of topics and get rid of biased results. In the previous week we did a comparative study of LDA using genism and LDA using scikit-learn to explore options to find the optimal model and to present the results. The need for such a study was because though LDA using Genism was fast, the predictions at times led to biased results. To automate the method, we have developed a machine learning model which classifies the incident tickets and categorizes it into 5 categories (topics namely Connectivity, Network, Code, Administrative and Permission).

The classification was accomplished utilizing LSTM (Long Short-Term Memory) model with Keras sequential model and fit to foresee and classify into 5 teams. Before this process, the data had undergone multiple pre-processing steps like lower case conversion, stop word removal, tokenization, lemmatization, text vectorization and generation of a document word matrix. Since the incoming incident tickets were labeled and grouped to 5 different teams, the subsequent task was to allocate the job to the team members in the best possible manner using the linear sum assignment problem. Moreover, we prioritized the named tickets utilizing Naïve Bayes Classifier strategy to create the tickets settled as early as conceivable based on the needs.
After the assignment of jobs among the team wass done, the next crucial task was predicting the priority of each of the incident tickets. We used Naive Bayes (NB) algorithm which is a supervised machine learning algorithm to predict the classification of a query sample by relying on labeled input data which are separated into classes.

We started the process of deployment using flask and integrating it with ServiceNow using API calls. This is one of the crucial steps in our entire project as the deployment involves extracting tickets from ServiceNow and using these tickets for preprocessing, classification, team assignment and prioritization and finally passing the predicted results back to ServiceNow using Flask API calls. The incident tickets from ServiceNow from Endava is extracted using python integration and used by our backend application Flask. Flask APIs receives Incident Ticket details through GUI or API calls, computes the predicted value based on our model and returns it and make the results available to the ServiceNow platform (frontend). The main advantage of using Flask is the cost effectiveness as well as the simpler steps for the deployment process.

We completed the process of deployment where the result of classified tickets is transferred to the ServiceNow platform.

- Successfully implemented the IT Service Ticket Management system with pretty goof results.
- Tried to tackle the problem by using LDA, LSTM and Naïve Bayes machine learning algorithms.
- Implemented the Linear Sum Assignment Problem to identify the right resource.
- Deployed the models in ServiceNow platform using Flask.

- The model can be used for real-time predictions which will increase the productivity of the company and customer satisfaction.

XIV. References

[1] ServiceNow products and Services explained. Nelson Frank. (2021, March 4). Retrieved December 18, 2021, from https://www.nelsonfrank.com/insights/servicenow-products-services-explained

[2] says:, D. S. (2020, November 25). What is ServiceNow? - getting started with the ServiceNow platform. Edureka. Retrieved December 18, 2021, from https://www.edureka.co/blog/what-is-servicenow/

[3] Kishore, P. (2019, March 17). It support ticket classification and deployment using machine learning and Aws Lambda. Medium. Retrieved December 18, 2021, from https://towardsdatascience.com/it-support-ticket-classification-and-deployment-using-machine-learning-and-aws-lambda-8ef8b82643b6

[4] Eichhorn, G. (2020, April 27). Predict IT support tickets with machine learning and NLP. Medium. Retrieved December 18, 2021, from https://towardsdatascience.com/predict-it-support-tickets-with-machine-learning-and-nlp-a87ee1cb66fc

[5] Kandakumar, K. K. (2019, March 16). It support ticket classification using machine learning and ML model deployment. Medium. Retrieved December 18, 2021, from https://medium.com/@karthikkumar_57917/it-support-ticket-classification-using-machine-learning-and-ml-model-deployment-ba694c01e416