National College of Ireland

Enhancing Incident Detection and Response in Cloud Computing by classifying Customer Support Cases

MSc Research Project
MSc In Cloud Computing

Aishwarya Raut
Student ID: 21175748

School of Computing
National College of Ireland

Supervisor: Prof. Shivani Jaswal

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Aishwarya Raut
Student ID:	21175748
Programme:	MSc In Cloud Computing
Year:	2023-2024
Module:	MSc Research Project
Supervisor:	Prof. Shivani Jaswal
Submission Due Date:	14/12/2023
Project Title:	Enhancing Incident Detection and Response in Cloud Computing by classifying Customer Support Cases
Word Count:	6362
Page Count:	21

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Aishwarya Raut
Date:	14th December 2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Enhancing Incident Detection and Response in Cloud Computing by classifying Customer Support Cases

Aishwarya Raut
21175748

Abstract

This report delves into the pivotal role of incident detection and response systems in ensuring availability and minimizing downtime within cloud computing environments. This research focuses on enhancing incident detection capabilities through the application of machine learning models, specifically, non-negative matrix factorization (NMF) for unsupervised clustering and transformer models such as DistilBERT, XLMRoberta, and CNN+LSTM with Glove Embedding for supervised classification. The investigation revolves around automating the categorization of customer support tickets, addressing challenges through meticulous data loading, preprocessing, and exploration. NMF reveals distinctive patterns in support tickets, while transformer models undergo rigorous training, evaluation, and performance analysis.

The research unfolds with a noteworthy achievement in the realm of incident detection within cloud computing environments. A standout result is the exceptional performance of the FineTune XLMRoberta Transformer, demonstrating high accuracy and robust categorization across various customer support ticket types. This outcome accentuates the significance of thoughtful model selection and fine-tuning, offering valuable insights into optimizing incident response strategies. However, the research is not without limitations, such as dependencies on the quality and diversity of the initial dataset and the need for periodic model updates. Despite these challenges, the results offer practical implications and feasibility of leveraging machine learning models for incident detection in cloud environments. The research contributes substantially to the field, providing a roadmap for selecting suitable algorithms and improving overall incident response efficiency. The success of integrating advanced machine learning models not only bridges theoretical gaps but also showcases practical implications for incident management in real-world scenarios.

Keywords— DevOps, Site Reliability Engineering, Machine Learning, Incident Detection, Reliability, Availability

1 Introduction

In the world of incident management, quickly figuring out and correctly labeling problems is super important. The objective is to help companies handle issues better and decide which ones to tackle first which means wanting to speed up how fast we respond to problems, make it easier to declare issues across different parts of a system, and improve how we fix things. This research is all about making this process way better for cloud systems. The main aim is to to create a classification system that can blend with ticketing tools. This means organizing and labeling incidents can be based on different criteria's such as, how urgent they are, and how much they impact things. We aim to give companies a solid system to deal with issues quickly and smartly. The big picture objective is to make cloud systems stronger, ultimately boosting the overall quality of service they deliver. Below mentioned is a scenario which shows what is the necessity of creating such a systemAhmed and Singh (2023)

Figure 1 depicts a real incident stemming from a misconfigured change in the Storage service. This misconfiguration had a domino effect, impacting multiple SQL databases and subsequently causing disruptions for Web Application instances dependent on these databases. The sequence of alerts for Storage, SQL, and Web Application was triggered at 3:54 am, 4:00 am, and 4:25 am, respectively. Following in-depth discussions, it took nearly 50 minutes to recognize the cross-service nature of the issue and officially declare it as an incident. An experienced Incident Commander (IC) was then engaged to orchestrate the mitigation process, successfully concluding at 5:27 am and restoring all services to normal operation.Li and Zhang (2021)

In light of the challenges illustrated in Figure 1, where a misconfiguration triggered a cascading impact, disrupting services, the aim is to address such incidents more effectively by proposing a robust

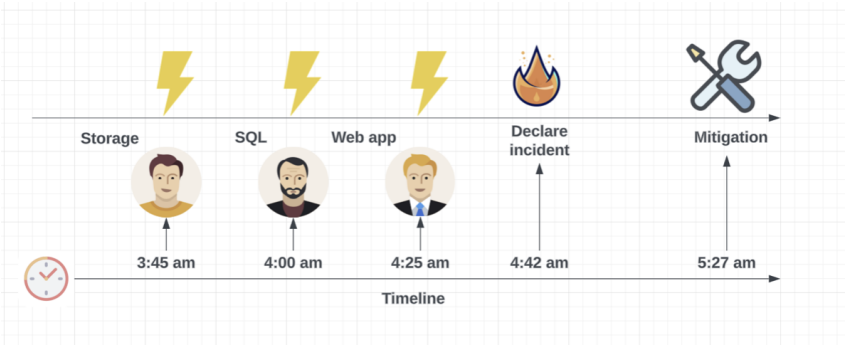


Figure 1: (Original)Timeline to declare and mitigate an incident

incident classification and response system. To do this, the planned flow is about combining proven methods like Warden's with insights from customer support cases. The idea is to help companies not just respond fast to problems but also learn from them. Thus making it easier for companies to adapt quickly, improve their strategies, and get better at defending against future challenges.Jain (2021)

In summary, this research endeavors to revolutionize incident detection in cloud systems, fostering faster response times, streamlined cross-platform incident declaration, and efficient mitigation processes. This research is about more than just theory; it's about making incident detection in cloud systems way better. Thus aiming for faster responses, smoother ways to declare problems across different parts of a system, and more effective ways to fix things.

2 Related Work

The examination of existing literature highlights on incident detection methodologies, mainly employing rule-based approaches, anomaly detection, and machine learning techniques. Despite the promising outcomes associated with these approaches, a notable research gap is seen in using machine learning techniques and integration of various incident data sources to enhance overall detection capabilities. Moreover, the literature emphasizes the significance of incident response frameworks and best practices as crucial elements in minimizing downtime and effectively mitigating the impact of incidents.

2.1 The requirement for automated incident detection, along with the development of frameworks designed to automate both incident detection and response in cloud systems

The paper "Fighting the Fog of War: Automated Incident Detection for Cloud Systems" Li and Zhang (2021) by Liqun Li introduces a novel strategy for incident detection in cloud systems using machine learning. The Warden framework, tailored for incident detection based on monitor-reported alerts, serves as a foundational reference for the current research. Warden leverages historical failure patterns and selected monitor data to robustly detect potential incidents and notify on-call engineers promptly. Warden employs a classification model trained on historical failure patterns, and the authors provide a comprehensive evaluation of its performance. The paper employs a Balanced Random Forest (BRF) classification model, demonstrating superior precision and recall, ensuring high accuracy in incident detection. Despite its strengths, Warden's reliance on manually linked alerts introduces challenges related to label quality, impacting real-world incident evaluation.

The scholarly workChen and Zhang (2020) "How Incidental are the Incidents?"by Pu Zhao focuses on incident management, particularly in identifying and prioritizing incidental incidents. The empirical study introduces DeepIP, a deep learning approach displaying commendable performance in incident prioritization across 18 online service systems.

In this project of classifying customer support cases into incident management, a correlation emerges with Zhao's findings. Both incidents and customer support cases offer rich information for improved characterization and prioritization Classifying and integrating customer support cases into incident management proves valuable for optimizing prioritization, which enhances incident categorization, ensuring

prompt resolution of critical incidents..The qualitative analysis on customer support cases adds depth to incident understanding, aligning with Zhao's exploration of incident characteristics.

2.2 Selection of best suitable machine learning models for comparative analysis: DistillBERT, XLMRoberta, and CNN + LSTM with Glove Embedding

The academic paper titled "Comparing BERT against traditional machine learning text classification" Gonzalez-Carvaja (2021)by Gonzalez-Carvajal and Garrido-Merchan evaluates the efficacy of BERT, a cutting-edge Natural Language Processing (NLP) model, in comparison to traditional machine learning (ML) approaches (Santiago).

The key discoveries from this study indicate that BERT consistently outperforms traditional ML models in text classification across diverse scenarios and languages. BERT's advanced capabilities in NLP facilitate precise case classification, nuanced context comprehension, and efficient response generation. These advantages strongly influence this decision to select the BERT algorithm for training machine learning model, particularly in the classification of customer support cases for accelerated incident detection. By harnessing BERT's contextual understanding, this report can gain from enhanced language representations, leading to more accurate and context-aware classification of customer issues.

While BERT stands out as a cutting-edge choice, the work of Sanh, Debut, Chaumond, and Wolf introduces an intriguing alternative in the form of DistilBERT."DistilBERT, a distilled version of BERT: smaller, faster, cheaper, and lighter".Sanh and Debut (2020) Recognizing the challenges associated with deploying large-scale pre-trained models in resource-constrained environments, DistilBERT offers a compelling solution. Through knowledge distillation during the pre-training phase, DistilBERT achieves a remarkable 40 per cent reduction in size, maintaining 97percent of BERT's language understanding capabilities. Furthermore, it boasts a 60percent increase in inference speed, addressing concerns related to computational budgets and on-the-edge applications.

Key advantages of DistilBERT lie in its efficiency, cost-effectiveness, and high performance across various downstream tasks. The reduction in model size aligns with environmental considerations associated with training large models. The study demonstrates that DistilBERT, despite its smaller footprint, remains competitive with BERT in terms of performance, making it an attractive option for real-world applications.

In light of these findings, this paper adopts a nuanced approach in the selection of an NLP model for customer support case classification. While recognizing the remarkable achievements of BERT, the decision to employ DistilBERT is rooted in its efficiency, cost-effectiveness, and demonstrated excellence across diverse tasks.

One key paper that significantly influenced the model selection process and critical analysis is "Comparative Analyses of BERT, RoBERTa, DistilBERT, and XLNet for Text-Based Emotion Recognition" ADOMA1 and HENRY2 (2020) by Acheampong Francisca Adoma, Nunoo-Mensah Henry, and Wenyu Chen. The paper explores the efficacy of BERT, RoBERTa, DistilBERT, and XLNet in recognizing emotions from text, employing a fine-tuning approach on the ISEAR dataset.

Through a meticulous comparative analysis, the authors evaluate these transformer models based on accuracy, precision, and recall.Notably,the findings reveal that RoBERTa emerges as the most effective model for emotion recognition,outperforming its counterparts.The insights and methodologies articulated in aforementioned paper have been judiciously incorporated into this report to systematically inform the decision-making process.

The paper by Yaroslav A. Seliverstov et al. "Detection of Low-toxic Texts in Similar Sets Using a Modified XLM-RoBERTa Neural Network and Toxicity ConfidenceParameters"Seliverstov and Komissarov (2021) serves as a central reference, warranting a critical analysis of the choice to utilize XLM-RoBERTa for classifying incidents derived from customer tickets.The study introduces a modified neural network based on the XLM-RoBERTa transformer architecture, demonstrating its efficacy in detecting low-toxic texts without retraining on such data.

The strategic selection of XLM-RoBERTa is validated by its adaptability, effectiveness, and innovative approach to toxicity confidence parameters. The study concludes with an evaluation of the model's performance on a new test dataset from another educational institution, confirming the efficiency of the approach.

Unlike traditional approaches that retrain models on low-toxic texts, the proposed methodology leverages a modified XLM-RoBERTa model trained on highly toxic texts. The innovation lies in dynamically adjusting the toxicity confidence parameter during classification, eliminating the need for explicit retraining which is been used in current project as well.

The paper titled "Application of XLM-RoBERTa for Multi-Class Classification of Conversational Hate Speech"Leburu-Dingalo and Ntwaagae (2022) by Tebo Leburu-Dingalo et al. addresses the challenge of identifying conversational hate speech in code-mixed languages, specifically Hinglish (Hindi-English) on social media. The authors use the XLM-RoBERTa multilingual model for transfer learning.

They preprocess the data by cleaning special characters, URLs, and augmenting tweets with textual descriptions of emojis. The experimental setup involves training and validation datasets, and they fine-tune the XLM-RoBERTa model which was useful as guide in this project.The authors justify the selection of XLM-RoBERTa by highlighting its effectiveness in multilingual text classification tasks. The choice is further supported by the model's success in the competition.

This paper is written by A. Kitanovski, M. Toshevska, and G. Mirceva named as "DistilBERT and RoBERTa Models for Identification of Fake News"Toshevska and Mirceva (2023) investigates the effectiveness of DistilBERT and RoBERTa models, two state-of-the-art language models, in detecting fake news. The exploration covers training both models on labeled news articles and evaluating their performance on distinct datasets, comparing accuracy, precision, recall, and F1-score.

The insights derived from this paper have been instrumental in informing the decision-making process and the methodology regarding the selection of models for this ticket classification comparative analysis. The detailed discussions on model architectures, experimental setups, and performance metrics have played a crucial role in guiding the approach for current project.

This study underscores the potential of transformer models, particularly DistilBERT and RoBERTa. Thus adapting and appling these insights to the ticket classification analysis,I've acknowledged the significance of thorough preprocessing of data for optimal model performance.

The decision to employ the CNN-LSTM algorithm in this study was informed by its notable success rate and efficiency, as demonstrated in the work of Yanli Shao (2021)

The paper titled "Towards Robust Online Sexism Detection: A Multi-Model Approach with BERT, XLM-RoBERTa, and DistilBERT for EXIST 2023 Tasks" Giachanou and Bagheri (2023)explores the application of transformer-based models in identifying and categorizing online sexism. The study, focusing on the EXIST 2023 shared task, emphasizes the importance of Natural Language Processing (NLP) in detecting harmful content. The methodology involves a combination of BERT, XLM-RoBERTa, and DistilBERT models, with additional datasets and preprocessing techniques. The research shows optimal performance in English and suggests future work, including incorporating annotator information and addressing imbalanced datasets.

This paper provided insights into advanced NLP methods, model architectures, data preprocessing, and evaluation strategies. The methodology's emphasis on addressing nuanced and context-dependent language aligns with the challenges faced in classifying customer support cases accurately. Future research directions outlined in the paper offer guidance for enhancing the robustness and generalizability of ticket classification models.

2.3 Alternative Approaches to Improve System Performance and Incident Management

The paper titled "Efficient Customer Incident Triage via Linking with System Incidents"
Jiazhen Gu (2020) by Chen proposes the LinkCM approach, employing transfer learning and semantic representation to automate incident triage in large-scale cloud service systems. The study highlights practical case studies demonstrating its potential to enhance incident resolution accuracy and reduce response times. In relation to this research on classifying incidents using three models for comparative analysis, both papers share a common goal of leveraging incidents and customer feedback to enhance system maintenance. While the Chen paper focuses on incident triage in cloud service systems, this research explores the integration of customer support cases into the software development process. Both approaches underscore the importance of utilizing customer insights for improved system performance, quality, and customer satisfaction.

Another paper, "Identifying Linked Incidents in Large-Scale Online Service Systems" Chen and Yang (2020a) by Hang Dong, introduces LiDAR, a deep learning-based framework for identifying linked incidents. This paper complements this research on an intelligent Incident Management (IcM) system, known as "Warden," which leverages machine learning for automated incident management. Dong's research on incident linkage aligns well with the goal of incorporating customer support cases into incident management, providing insights and methods to enhance the overall system health view.

2.4 Thorough understanding of automated incident detection and its importance

The paper "Towards Intelligent Incident Management: Why We Need It and How We Make It"Chen and Yang (2020b) by Yu Kang addresses challenges in cloud incident management, introducing the BRAIN framework that utilizes AI and ML for detection, auto-triage, and correlation. The framework is applicable to a case study on improving incident management in a cloud-based customer support system. The paper's empirical validation methodology offers a structured approach for evaluating incident management effectiveness.

"Knowledge Management Challenges in Customer Support: A Case Study"Kirsi Tanskanen (2023) by Marko explores knowledge management aspects in customer support, highlighting challenges in documentation, classification, knowledge sharing, and metrics. For researchers automating incident detection with customer support cases, this paper provides insights into factors hindering effective incident resolution, informing the design of automated systems.

"STAR: A System for Ticket Analysis and Resolution"Wubai Zhou (2017) by Wei Xue presents an automated system for IT service management ticket resolution, introducing a deep neural ranking model. This model, with character-level, entity-level, semantic-level, and attribute-level features, outperforms baselines, offering a robust framework for automating ticket resolution in IT-related issues. The study aligns with research goals, indicating potential efficiency gains in ticket resolution and improved service delivery.

3 Project Methodology

In the subsequent section, we delineate a systematic and comprehensive approach to realizing the project flow. The methodology unfolds in a step-by-step process, encompassing data loading, cleaning, preprocessing, topic modeling, data visualization, splitting, model building, performance measurement, GUI development using Flask, and deployment using AWS CodeDeploy, Elastic Beanstalk, and EC2 instances.

Dataset:

- The project utilizes a dataset retrieved from Kaggle, comprising customer complaints in a financial company, stored in a .json format.

Source and Ownership

- **Source:** Kaggle
- **Author/Owner:** Venkatasubramanian SundraMahadevan

Features of the Dataset

The dataset comprises the following key features:

- **Complaint ID:** A unique identifier assigned to each consumer complaint.
- **Zip Code:** The zip code associated with the consumer filing the complaint.
- **Date Received:** The date when the complaint was submitted.
- **State:** The state in which the complainant resides.

- **Product:** The financial product or service associated with the complaint (e.g., Debt collection, Mortgage, Credit card or prepaid card).
- **Issue:** A brief description of the consumer's complaint.
- **Company Response:** The response provided by JPMORGAN CHASE and CO. to the consumer's complaint.
- **Submitted Via:** The method through which the complaint was submitted (e.g., Web, Phone, Referral).
- **Timely:** Indicates whether the complaint was submitted in a timely manner (Yes/No).
- **Sub Product:** Additional categorization of the product (e.g., Credit card debt, Checking account).
- **Consumer Consent Provided:** Indicates whether the consumer provided consent regarding the complaint.
- **Tags:** Additional tags associated with the complaint, such as "Servicemember" or "Older American."
- **Consumer Disputed:** Indicates if the consumer disputed the company's response (Yes/No).
- **Company Public Response:** Any public response provided by JPMORGAN CHASE & CO. regarding the complaint.
- **Complaint What Happened:** Detailed description of what happened according to the consumer.
- **Sub Issue:** Additional details regarding the specific issue (e.g., Debt is not yours, Problem using a debit or ATM card).

Details of dataset

- **Source/Provenance**:The dataset is sourced from the "complaint-public-v2" index.
- **Size of the dataset**:The dataset is 83.39MB in size.
- **Number of records**:The dataset comprises of 78,313 customer complaints.
- **Number of features per record**:The total number of features/attributes per record is 22.
- **Updation frequency**:The dataset was last updated 2 years ago.
- **File Format:** The data is presented in a JSON-like format.

File Description

Each entry includes information such as complaint details, zip code, date received, state, product, issue, company response, submission method, and consumer consent. The data exploration can involve techniques such as topic modeling on the .json data. Since the data is not labeled, techniques need to be applied to analyze patterns and classify tickets into the specified clusters. For this project, the technology for label creation has been used is Non-Negative Matrix Factorization.

A. Data Preprocessing:

1. Data preprocessing is a pivotal step in refining the dataset for further analysis.
2. Importing All Libraries:
   - Firstly importing essential libraries such as Python, VS code, Colab, TensorFlow, Keras, scikit-learn, Transformers, NLTK, Plotly, Seaborn, Flask, etc.
   - These libraries will be used for various tasks in the project.
3. Data Loading:
   - **Input:**
     - Raw dataset retrieved from Kaggle, comprising customer complaints in a financial company, stored in a .json format.

- **Procedure:**
  - The JSON file is opened, and its contents are loaded into a Python dictionary.
  - The `json_normalize` function is employed to transform this hierarchical JSON data into a structured tabular format within a Pandas DataFrame.
- **Output:**
  - Pandas DataFrame loaded with 78,313 customer complaints characterized by 22 features.
  - To commence the analysis, the dataset is loaded into a Pandas DataFrame using Python.

4. Data Cleaning:

- This step involves several key processes aimed at handling missing values, standardizing column names, and refining textual data.
  - **Input:**
    - Loaded DataFrame with raw data.
  - **Procedure:**
    - Null values are identified and removed, ensuring the quality of textual data.
    - A comprehensive text cleaning function is designed to preprocess textual data in the `'complaint_what_happened'` column.IKONOMAKIS (n.d.) This function includes steps such as removing URLs, decontracting contractions, filtering out punctuation, separating alphanumeric characters, handling repeated characters, splitting attached words, and removing stopwords.
    - The data cleaning task includes steps such as removing URLs, decontracting contractions, filtering out punctuation, and handling repeated characters.
  - **Output:**
    - Cleaned DataFrame with standardized column names, removed null values, and preprocessed textual data.

5. Data Preprocessing:

- **Input:**
  - Cleaned DataFrame with textual data.
- **Procedure:**
  - Text lemmatization is performed using the spaCy library, transforming words into their base or root form.
  - Parts-of-speech (POS) tagging is employed to identify the grammatical category of each word in a sentence. Only nouns (NN) are extracted.
  - N-grams (contiguous sequences of n items) are extracted from the cleaned text.
- **Output:**
  - Processed DataFrame with lemmatized text, extracted POS tags, and top n-grams.

B. Exploratory Data Analysis (EDA):

- Exploratory Data Analysis (EDA) is a phase that involves gaining insights into the dataset through statistical and visual methods. This step aids in understanding the distribution of data, relationships between variables, and potential patterns that can inform subsequent modeling decisions.
- The data visualization for the project is done using various plots and graphs as mentioned below **Visualizing Unigrams and Trigrams:**
  - **Input:** Processed text data in `'data_clean'`.
  - **Output:** Plots showing the count of each class in the target variable ('label') and visualizations of word clouds, histograms, and count plots.
  - *1.2 Topic Modeling Results:* This section shows the results of topic modeling using Non-negative Matrix Factorization (NMF). The top 15 words for each topic are displayed, providing an understanding of the key terms associated with each topic. The best topic for each complaint is assigned, and a mapping to human-readable labels is performed.

– *1.3 Data Visualization:* This visualization section covers diverse aspects

Histogram:

– **Input:** Lengths of each row of the text data.

– **Procedure:**

* **Calculate Text Lengths:**
  · For each row of the cleaned text data (`clean_description_clean` column), calculate the length of the text (number of characters or words).
  · Store the lengths in a list or array.
* **Plot Histogram:**
  · Utilize a plotting library (e.g., Matplotlib) to create a histogram.
  · Set the number of bins (intervals) for the histogram, defining the range of text lengths to be considered.
  · Choose an appropriate bin size to capture the distribution effectively.
* **Labeling and Styling:**
  · Add labels to the X-axis and Y-axis for clarity.
  · Provide a title to the histogram that conveys the purpose of the visualization.
  · Optionally, customize the appearance, such as color, transparency, or other stylistic elements.
* **Display or Save:**
  · Display the histogram within the Jupyter Notebook or save it as an image file for later reference.
  · Choose an appropriate format for saving the image (e.g., PNG, JPEG).

Bar Plot: Count of Each Class in the Target Variable ('label'):

– **Input:**

* DataFrame `'data_clean'` with labeled topics.

– **Procedure:**

* A bar plot is created to visualize the count of each class in the target variable ('label').
* This plot provides an overview of the distribution of different topics in the dataset.

– **Output:**

* A bar chart with labels on x-axis and counts on y-axis i.e a bar plot showing the count of each class in the target variable ('label')

Wordclouds for Each Class::

– **Input:**Cleaned text data `(clean_description_clean column)` for each class.

– **Procedure:**Utilized the WordCloud library to create wordclouds for each class.

– **Output:**Wordclouds visualizing the words where they are weighted based on their frequency in the text data associated with each class.

C. Topic Modelling using NMF:

1. Unlabeled Data:

1. The initial dataset contains text data but lacks explicit class labels or categories.

2. Non-Negative Matrix Factorization (NMF):

1. NMF is used as an unsupervised clustering technique to group similar documents or text entries based on the underlying topics or patterns in the data.
2. The number of clusters or topics (classes) is determined through a trial-and-error approach or some validation metric.

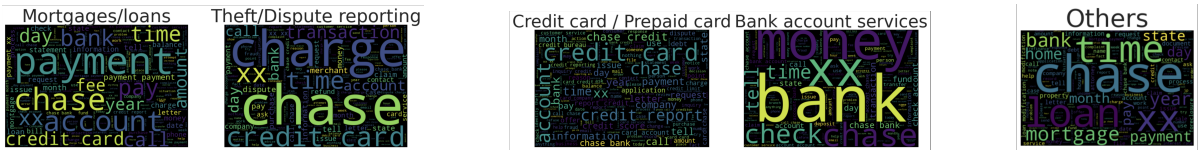
3. Word Clusters:

1. Once the optimal number of clusters is determined, word clusters are created for each class.
2. A total of 5 optimal clusters were created in this project named Mortgages/Loans, Theft/dispute reporting, Credit card/prepaid card, Bank account services, and Others.
3. Each cluster represents a set of words that tend to co-occur within the documents assigned to that cluster.

4. Word Cloud Generation for Each Class:

1. For each class/cluster in the dataset, a word cloud has been generated.
2. Words are weighted based on their frequency in the text data associated with each class.

Word clouds generated for each class are as shown in figure 2:



(a) Loan and Theft (b) Card and services (c) Others

Figure 2: 5 word clouds for clusters created

5. Inspect and Validate:

1. The next step involves inspecting and validating the correctness of each cluster.
2. This may include manual inspection to ensure that the words in a cluster are semantically related and make sense within the context of a class.

6. Labeling Clusters:

1. Clusters are then labeled or assigned specific topics based on the predominant words within them.
2. These labels effectively serve as the classes for the previously unlabeled data.

7. Mapping Clusters to Topics:

1. The clusters generated by NMF are mapped to human-readable topics or categories.
2. This mapping allows for interpreting the content of each cluster in the context of a specific class or theme.

In summary, the classes are effectively created through the unsupervised clustering process of NMF. The clusters are identified based on the natural patterns and topics present in the unlabeled text data, and subsequent steps involve validating and assigning human-interpretable labels to these clusters.

D. Splitting Data:

1. Split the dataset into training, validation, and test sets with a ratio of 60:20:20 to evaluate the model's performance on unseen data.
2. Training data is 60%, where the model learns patterns, relationships, and features.
3. Validation data is 20%, which is used to fine-tune the model during the training phase. It helps in adjusting hyperparameters and avoiding overfitting.
4. Testing data is 20%, which is reserved for evaluating the model's performance on unseen data. Once the model is trained and tuned using the training and validation sets, it is tested on this independent set to assess how well it generalizes to new, unseen data.

E. Model Building:

1. The 3 algorithms which have been chosen to train the model for comparative analysis are as follows:
   - CNN-LSTM with Glove embedding

- Fine-tune DistilBERT Transformer
- Fine-tune XLM-RoBERTa Transformer. Singh (2021)

F. Performance Measurement:
The performance of the model is been measured using the following metrics:

1) Accuracy and loss graphs

- 1.1 Training and Validation Accuracy:
  - *Training Accuracy (Blue Line)*: Represents the accuracy of the model on the training dataset. It shows how well the model is fitting the training data over epochs. An increasing trend indicates effective learning.
  - *Validation Accuracy (Red Line)*: Illustrates the accuracy on a separate validation dataset not used during training. It serves as a proxy for the model's ability to generalize to new data. A close alignment with the training accuracy suggests good generalization.
- 1.2 Training and Validation Loss:
  - *Training Loss (Blue Line)*: Represents the value of the loss function on the training dataset. It measures how well the model's predictions match the actual labels. The goal is to minimize this value.
  - *Validation Loss (Red Line)*: Depicts the loss on the validation dataset. It gauges how well the model is performing on unseen data. A similar or slightly higher loss on validation compared to training is acceptable.

2.Classification report

The classification report provides precision, recall, and F1-score for each class, offering a detailed evaluation of the model's performance on individual classes.

3. Confusion Matrix:

A confusion matrix is a pivotal tool for evaluating the performance of a classification model. It provides a detailed breakdown of the model's predictions, allowing a deeper understanding of how well it categorizes instances into different classes.
G. GUI (Flask):

- Graphical user interface (GUI) is been developed by making use of Flask to facilitate user interaction with the trained models.

H. Deployment (AWS CodePipeline, Elastic Beanstalk, s3 bucket):

- Deploy models using AWS services such as Elastic Beanstalk making use of AWS CodePipeline and s3 bucket.

4 Design Specification

4.1 Proposed model

For the development of the support ticket classification system, a comprehensive comparative analysis of three machine learning algorithms—DistilBERT, XLMRoberta, and CNN+LSTM with Glove Embedding—was conducted. After rigorous evaluation based on various parameters and performance metrics, XLMRoberta emerged as the best-performing algorithm for this task.

XLMRoberta Model Description:

To implement the XLMRoberta model, the following steps were taken:

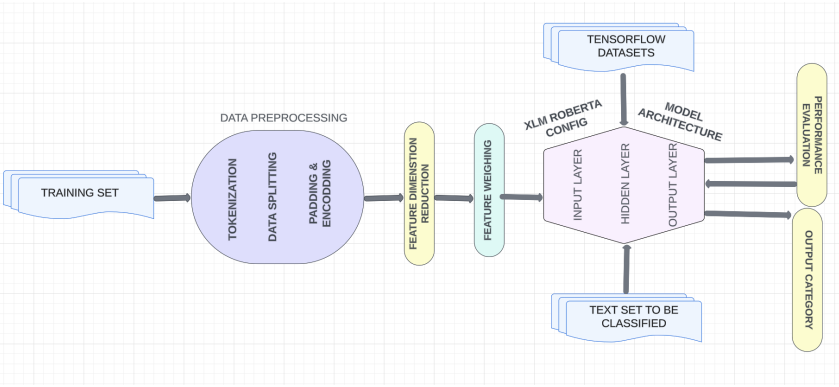


Figure 3: Model Design diagram of the project

- **Tokenization:** Utilized the Hugging Face AutoTokenizer to tokenize the text using the "xlm-roberta-base" pre-trained model.
- **Data Splitting:** The dataset was split into training, validation, and test sets using the `train_test_split` function with a ratio of 60:20:20 to evaluate the model's performance on unseen data where training data is 60%, validation data is 20%, testing data is 20%.
- **Padding and Encoding:** Applied padding with a specified maximum length to ensure consistency in dimensions for training, validation, and test sets. Encoded the text using the XLMRoberta tokenizer.
- **TensorFlow Datasets:** Configured TensorFlow datasets for the training and validation sets.
- **XLMRoberta Model Configuration:** Configured the XLMRoberta model using the XLMRobertaConfig with specified dropout and attention dropout rates.
- **Model Architecture:** Developed a custom model architecture in Keras, making the Transformer layers of XLMRoberta untrainable. Incorporated input layers for input IDs and attention masks, processing the model's last layer's hidden-state output. Extracted the [CLS] token for further processing through hidden and output layers. Applied dense layers with dropout for feature extraction and classification. Defined the model using the Keras functional API.
- **Model Compilation:** Compiled the model using the Adam optimizer, categorical crossentropy loss, and accuracy as the metric.
- **Model Training:** Trained the XLMRoberta model on the TensorFlow dataset for five epochs, with batch processing.
- **Performance Visualization:** Visualized training and validation accuracy and loss using Matplotlib.
- **Evaluation:** Evaluated the model on the test set, generating predictions for further analysis.
- **Confusion Matrix:** Plotted a confusion matrix to visualize the model's performance on different classes.
- **Classification Report:** Generated a classification report, providing detailed metrics for model evaluation.

This robust methodology ensured the successful development and evaluation of the XLMRoberta model for support ticket classification. The model exhibits high accuracy and reliability, as evidenced by the comprehensive evaluation metrics and visualizations.

4.1.1 Justification for selection of model

- In the process of developing the support ticket classification model, I happened to meticulously survey a multitude of research papers, aiming to identify state-of-the-art models with proven efficacy.

- From this extensive review, thus selected three models—DistilBERT, XLMRoberta, and CNN+LSTM with Glove Embedding—each recognized for its robust performance in various natural language processing tasks which performed outstanding as compared to the rest of the models.
- These models were chosen based on their prevalence in the literature and demonstrated success in similar applications.
- Following this careful selection, the research involves a thorough comparative analysis of these three chosen models.
- The goal is to determine the most suitable model for our specific support ticket classification task and thus finally carrying the final classification task over the best fit model.
- By conducting a rigorous evaluation, the aim was to pinpoint the strengths and weaknesses of each model, ultimately identifying the best-fit solution.
- This meticulous approach ensures that the chosen model aligns seamlessly with the intricacies of incident detection within cloud environments, leading to the development of a highly effective and tailored support ticket classification system.

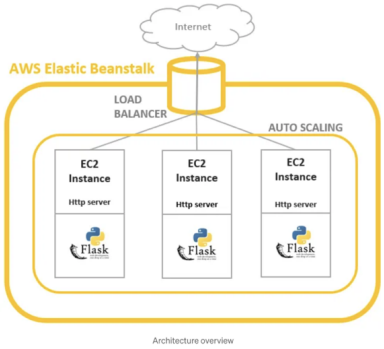
Selecting a model involves considering various factors beyond just accuracy. Here are some aspects that were considered when choosing the fine-tuned XLM-Roberta model:

- **Precision, Recall, and F1-Score:** These metrics provide a more nuanced understanding of the model's performance.
  Precision is important when false positives are costly, recall is crucial when false negatives are critical, and the F1-score balances both.
  Thus, in this project, the focus is on F1-score accuracy to balance both precision and recall.
  Considering accuracy, precision, recall, and F1-score across all classes, Roberta appears to be the best-performing model in this case.
- **Class Imbalance:** If the dataset has imbalanced classes, accuracy alone might not be a good indicator of model performance.
  In such cases, precision, recall, and F1-score for each class become crucial.
  Roberta seems to maintain a good balance across classes.
- **Task Requirements:** Consider the specific requirements of the individual task.
  Some tasks might prioritize precision, while others might prioritize recall.
  For instance, in medical diagnoses, there is a need for high recall to ensure that you don't miss any positive cases, even if it means more false positives.
  For this project, a good balance between precision and recall is needed, and Roberta is a suitable choice with high performance in both aspects.
- **Computational Resources:** More complex models generally require more computational resources for training and inference. Consider the available resources when choosing a model. If computational resources are a constraint, Distillbert is a lighter version of BERT and might be more suitable.
- **Interpretability:** Depending on the domain and the audience, model interpretability might be crucial. Some models, like CNN+LSTM, might be harder to interpret compared to transformer-based models like Roberta or Distillbert. As interpretability is important for this project, transformer models are preferred.

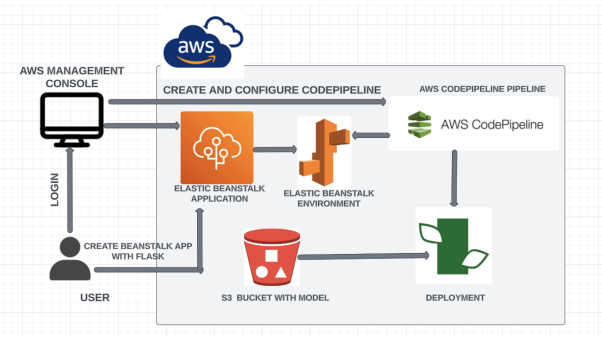
Taking all these factors into account, Roberta seemed to be the best choice based on the provided metrics and general considerations.

4.2 Architecture:

The high level architectural overview for deploying application over EC2 instances in the project is as given in 4a



(a) Deploying Flask application over EC2



(b) Architectural diagram of the project

Figure 4: Architectural diagrams

1. Model Development and Storage:

- **Algorithm Selection:** A careful choice was made upon 3 diverse machine learning algorithms for training the support ticket classification model.
- **Training Phase:** The selected algorithms were applied to a comprehensive dataset from Kaggle for training purposes and model artifacts are generated
- **Secure Storage in S3 Bucket:** This model artifacts are securely stored in an AWS S3 bucket named "supporttticketmodel" in the "Sydney" region.
- **Version Control:** The strategic use of S3 facilitates efficient version control, allowing for easy tracking and retrieval during subsequent deployment phases.

2. Flask Application Integration:

- **Post-Training Development:** Development of a Flask application commences after the completion of model training.
- **Intelligent Logic Integration:** Intelligent logic is incorporated into the application for preprocessing user input, invoking the trained model, and presenting classification results.
- **User Interface Design:** The Flask application is designed to serve as the user interface, ensuring seamless interaction with the support ticket classification model.

3. AWS Elastic Beanstalk Configuration:

- **Environment Setup:** An Elastic Beanstalk environment is configured, bearing the name "MyElasticBeanstalkAppSupportTicketNew5" and located in the "Sydney" region.
- **Streamlined Deployment:** Elastic Beanstalk autonomously handles deployment tasks, capacity provisioning, load balancing, and automatic scaling.The environment provides an optimal hosting solution for the Flask application, ensuring streamlined deployment and efficient resource management.

4. Integration with AWS CodePipeline:

- **Continuous Delivery Setup:** Implementation of an AWS CodePipeline, named "supportticketflaskcodepipeline," located in "Sydney," establishes continuous delivery.
- **Version Control Connection:** The CodePipeline is connected to a version control system GitHub, creating a linkage for automated deployments triggered by changes in the repository.
- **Automated Workflow:** The CodePipeline sets up an automated workflow, ensuring that any changes pushed to the connected repository trigger automatic deployments.

5. Flask Application Deployment Workflow:

- **Change Detection:** Upon detecting changes in the connected repository, the AWS CodePipeline initiates the deployment workflow.
- The deployment process is orchestrated, ensuring a smooth transition from the repository to the Elastic Beanstalk environment.
- The seamless integration guarantees that users can effortlessly access the deployed Flask application on AWS Elastic Beanstalk, promoting accessibility and user engagement.
  - The link to the deployed application to AWS Elastic Beanstalk is `http://myelasticbeanstalkappsupportticketnew5.eba-3hi7gf52.ap-southeast-2.elasticbeanstalk.com/`

Conclusion:

This comprehensive architecture embody a holistic approach, intricately weaving together stages of model development, application creation, and deployment. The focus remains on achieving user-friendliness, automation, and efficiency, ultimately establishing a robust and easily accessible support ticket classification system.

5 Implementation and Contribution

In this section, we detail the final stages of our implementation, outlining the outputs, tools used, and, most importantly, our substantial contributions to the project.

5.1 Transformed Data

Outputs:

- Transformed dataset with preprocessed and cleaned text data.
- TF-IDF vectorized representations for input into machine learning models.
- The artifacts of this model are safely stored in an AWS S3 bucket named "supportticketmodel" within the "Sydney" region.

Tools and Languages:

- Utilized Python with libraries such as scikit-learn, NLTK, and spaCy for data preprocessing and transformation.

5.2 Machine Learning Models Developed

Outputs:

- Three distinct machine learning models were been used for text classification based on customer complaints.
- Models include Convolutional Neural Network (CNN), Long Short-Term Memory networks (LSTM), a hybrid model combining CNN and LSTM and Transformer models.

Tools and Languages:

- Developed models using TensorFlow and Keras libraries in Python.
- Employed the Google Colab environment for enhanced computing resources.

5.3 Evaluation Metrics and Model Performance

Outputs:

- Evaluation metrics such as accuracy, precision, recall, and F1-score for model performance.
- Comparative analysis results highlighting the strengths and weaknesses of each model.

Tools and Languages:

- Utilized scikit-learn for calculating evaluation metrics.
- Python for data analysis and visualization.

5.4 Comparative Analysis Results

Outputs:

- Comprehensive comparative analysis showcasing the strengths and weaknesses of the three machine learning models.
- Insights into the suitability of each model for the specific task of customer complaint categorization.

Tools and Languages:

- Python with matplotlib and seaborn for creating visualizations.
- Jupyter Notebooks for documentation and analysis.

5.5 Contribution of the project

Summary:

- The novel aspect of this project lies in its groundbreaking approach to ticket classification based on customer complaints.
- Unlike existing models that rely on generic features such as on historical failure patterns or categories assigned from someone who creates tickets, our methodology harnesses the richness of customer interactions over years, thereby enhancing the model's understanding of nuanced issues.
- By integrating this novel classification approach, we aim to pave the way for a more tailored and responsive customer support system.
- This research extends beyond the conventional application of three machine learning models to a dataset.
- The methodology adopted involves a meticulous selection process, informed by an extensive review of research papers where choice is made by selecting the best models from comparative analysis made in different papers and then checking which is best-fit for this application based on variety of metrics where classification of unlabelled data is to be done.
- The novelty of our approach lies not only in the utilization of customer tickets or complaints but also in the incorporation of innovative data preprocessing techniques, including rigorous text cleaning, lemmatization, and part-of-speech tagging.
- A noteworthy contribution is the integration of Non-Negative Matrix Factorization (NMF) for topic modeling, providing a novel means of extracting meaningful clusters from unlabeled data in an unsupervised manner.
- Furthermore, our contribution encompasses not only algorithmic advancements but extends into the realm of data visualization as in various plots and reports.
- Techniques such as bar plots, word clouds, and histograms are strategically employed, serving as instrumental tools for a comprehensive exploration of the dataset.

  In summary, this work represents a purposeful and innovative endeavor, ensuring that each step taken significantly contributes to the overarching objective of refining the categorization of customer support tickets.

Impact:

- The insights gained from the comparative analysis contribute to the understanding of the strengths and limitations of various models in the context of customer support ticket classification.
- The findings inform future research directions and aid practitioners in selecting the most suitable model based on their specific requirements.

In conclusion, the implementation phase not only involved the development of machine learning models but, more importantly, provided valuable contributions through a meticulous comparative analysis. The work extends beyond mere model execution, offering insights that are valuable to practical aspects of customer support ticket classification.

6 Evaluation

- **Analysis:** Detailed analysis of experimental results from support ticket classification models to provide insights into each model's performance, drawing conclusions relevant to research questions and objectives.
- **Evaluation Metrics:** To comprehensively assess model performance, key metrics, including precision, recall, and F1-score, are employed.
- **Metrics Explanation:** Precision, recall, and F1-score are commonly used in classification tasks:
  - Precision: Accuracy of positive predictions (Precision = TP / (TP + FP)).
    - Calculated as the ratio of true positive predictions to the total number of positive predictions.
    - High precision indicates accurate positive predictions.
  - Recall: Ability to capture relevant positive instances (Recall = TP / (TP + FN)).
    - Calculated as the ratio of true positive predictions to the total number of actual positive instances.
    - High recall indicates effective identification of positive instances.
  - F1-score: Harmonic mean of precision and recall (F1-score = 2 * (Precision * Recall) / (Precision + Recall)).
    - Balances precision and recall, especially in class-imbalanced situations.
    - Ranges from 0 to 1, where 1 indicates perfect precision and recall.

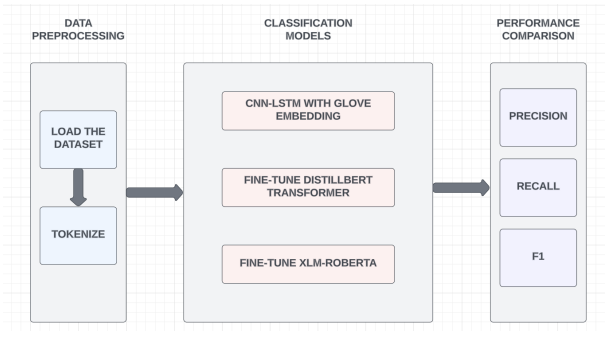


Figure 5: Evaluation metrics diagram

The research methodology unfolds in a systematic high level overview three-stage process: data preprocessing, where raw data is refined and cleaned; model training, where machine learning models are developed using advanced libraries; and evaluation, where the performance of the models is assessed based on precision, recall, and F1 metrics.The diagram flow for the same is given inFigure 5

In summary, precision emphasizes the accuracy of positive predictions, recall focuses on capturing all positive instances, and F1-score provides a balanced measure that considers both precision and recall. Depending on the specific goals and requirements of a classification task, one metric may be prioritized over the others. For example, in situations where false positives are costly, precision may be a more critical metric, while in scenarios where missing positive instances is costly, recall may take precedence. F1-score serves as a compromise between these two considerations and thus been taken as a final metric to measure the performance of the models mentioned below

6.1 Performance report of the 3 models

The performance metrics for the CNN + LSTM model with Glove Embedding is shown in Figure 6a
The performance metrics for the FineTune Distilbert Transformer model is shown in Figure 6b
The performance metrics for FineTune XLMRoberta Transformer is shown in modelFigure 6c is shown in Figure 9

CNN+LSTM WITH GLOVE EMBEDDING				
#	PRECISION	RECALL	F1-SCORE	SUPPORT
1	0.86	0.82	0.84	1069
2	0.61	0.96	0.74	1041
3	0.22	0.01	0.02	447
4	0.85	0.84	0.85	716
5	0.86	0.75	0.8	942
Accuracy			0.76	4215
Macro-avg	0.68	0.68	0.65	4215
Weighted avg	0.73	0.76	0.72	4215

(a) CNN+LSTM

FINE-TUNE DISTILLBERT TRANSFORMER				
#	PRECISION	RECALL	F1-SCORE	SUPPORT
1	0.74	0.91	0.81	1069
2	0.9	0.64	0.75	1041
3	0.88	0.51	0.65	447
4	0.64	0.86	0.85	716
5	0.68	0.84	0.75	942
Accuracy			0.78	4215
Macro-avg	0.81	0.75	0.76	4215
Weighted avg	0.8	0.78	0.77	4215

(b) FineTune DistilBERT

FINE-TUNE XLMROBERTA TRANSFORMER				
#	PRECISION	RECALL	F1-SCORE	SUPPORT
1	0.84	0.86	0.85	1069
2	0.84	0.85	0.84	1041
3	0.83	0.58	0.68	447
4	0.85	0.89	0.87	716
5	0.76	0.81	0.78	942
Accuracy			0.82	4215
Macro-avg	0.82	0.8	0.81	4215
Weighted avg	0.82	0.82	0.82	4215

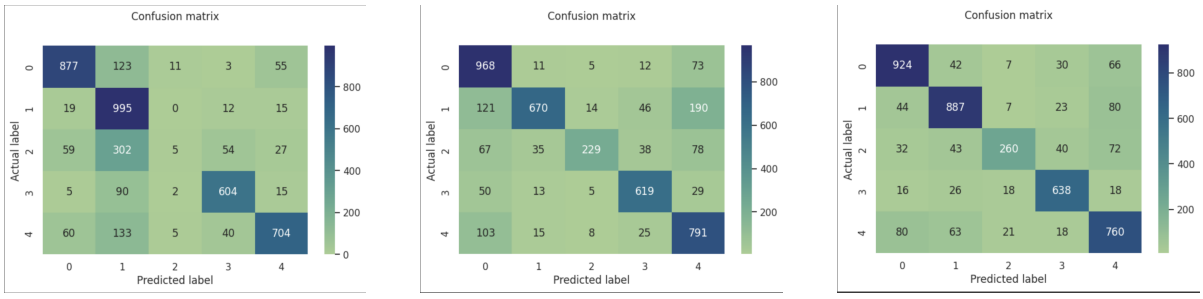
(c) XLM-RoBERTa

Figure 6: Comparison of Performance reports of different models

These results provide a comprehensive evaluation of each model's precision, recall, and F1-score across different classes.

6.2 Comparative analysis of Confusion Matrix of each model:

The comparison of confusion matrix for the CNN + LSTM model with Glove Embedding Figure 9a, the FineTune Distilbert Transformer model Figure **??** and the FineTune XLMRoberta Transformer modelFigure **??** is shown in Figure 7
In the context of the provided confusion matrix:



(a) CNN+LSTM (b) FineTune DistilBERT (c) XLM-RoBERTa

Figure 7: Comparison of Confusion matrix of different models

Actual Labels (Rows): Correspond to the true classes of the instances in the test dataset. Predicted Labels (Columns Represent the classes predicted by the model.

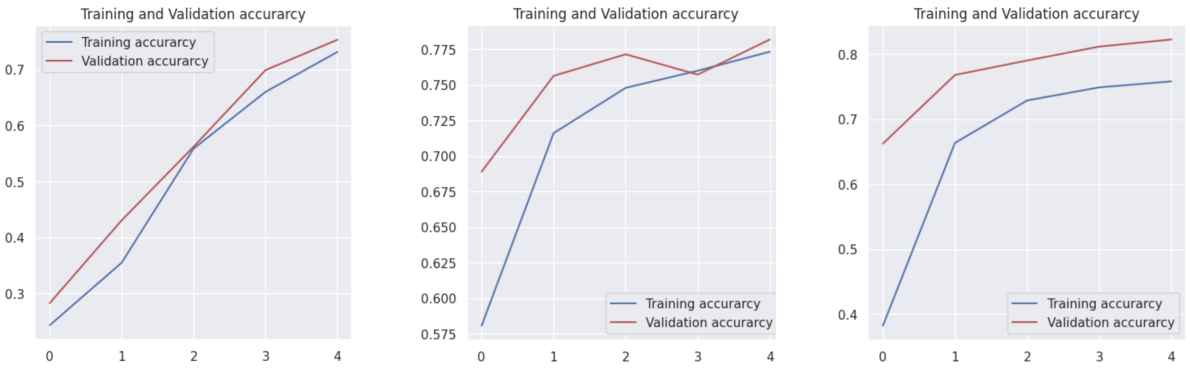
Key Metrics:
True Positives (TP): Instances correctly predicted as belonging to a particular class. *True Negatives (TN):* Instances correctly predicted as not belonging to a particular class. *False Positives*

(FP): Instances incorrectly predicted as belonging to a particular class. *False Negatives (FN):* Instances incorrectly predicted as not belonging to a particular class.

Interpreting a confusion matrix involves assessing various metrics to gauge the performance of a classification model as follows
1. Diagonal Elements (Top-Left to Bottom-Right): These represent correct predictions (True Positives and True Negatives). Higher values along the diagonal indicate strong predictive performance for corresponding classes.
2. Off-Diagonal Elements: These indicate instances where the model made incorrect predictions (False Positives and False Negatives). Pay attention to these elements, as they highlight areas where the model can be improved.
3. Brighter Colors: Brighter colors, especially along the diagonal, emphasize higher values and correct predictions, aiding in the identification of dominant patterns. Higher values suggest better performance.

Thus as per comparative analysis of confusion matrix XLM-Roberta model has given the best performance.

6.3 Comparative analysis of Training and validation accuracy graph:



(a) CNN+LSTM (b) FineTune DistilBERT (c) XLM-RoBERTa

Figure 8: Comparison of Training and validation accuracy of models

In a training and validation accuracy graph, an ascending trend for both curves is generally desirable. This indicates that the model is learning from the training data and improving its performance over epochs. Ascending Trend: Both the training and validation accuracy should show an upward trend initially, reflecting improved learning.
1. Convergence: As training progresses, ideally, the gap between the training and validation accuracy should not widen significantly. Convergence or a parallel trend suggests effective learning and generalization.
2. Overfitting Warning Signs: If the training accuracy continues to increase while the validation accuracy plateaus or decreases, it may indicate overfitting. Overfitting occurs when the model memorizes the training data but fails to generalize well to new, unseen data.
3. Consistency: A consistent and steady increase in accuracy for both training and validation sets indicates a model that is likely to perform well on new data.

Thus analysing all the 3 models we see the training and validation graph for XLM-Roberta is the best as it has an ascending trend, convergence between the two curves, and consistency in improvement over epochs.

6.4 Comparative analysis of Training and validation loss graph:

In a training and validation loss graph, certain patterns indicate the performance and generalization capabilities of the model which are as follows:

(a) CNN+LSTM (b) FineTune DistilBERT (c) XLM-RoBERTa

Figure 9: Comparison of Training and validation loss of models

Descending Trend: Both the training and validation loss should show a descending trend. This suggests that the model is improving in minimizing the difference between predicted and actual values.
Convergence: Similar to accuracy, the gap between training and validation loss should remain relatively stable or narrow. A widening gap might signal overfitting, where the model fits the training data too closely but struggles with new data.
Overfitting Warning Signs: If the training loss continues to decrease while the validation loss increases, it could indicate overfitting. This scenario implies that the model is becoming too specialized in the training data and may not generalize well.
Stabilization: As training progresses, both training and validation losses should stabilize. This signifies that the model has reached a point where further training may not significantly improve performance.

Thus as per the training and validation loss curve, XLM-Roberta has performed the best as it has a descending trend in both training and validation loss, convergence between the two curves, and stabilization as training advances.

6.5 Discussion

6.5.1 CNN + LSTM with Glove Embedding
1. The CNN + LSTM model with Glove Embedding achieved an accuracy of 76%.
2. It demonstrated notable precision and recall for class 1, indicating a strong ability to identify relevant support ticket categories.
3. However, the low precision and recall for class 2 suggest challenges in correctly classifying tickets in this category.

6.5.2 FineTune Distilbert Transformer
1. The FineTune Distilbert Transformer model achieved an accuracy of 78%.
2. It displayed high precision and recall for class 0, indicating effectiveness in identifying tickets in this category.
3. The model also demonstrated balanced performance across other classes, contributing to its overall robustness.

6.5.3 FineTune XLMRoberta Transformer
1. The FineTune XLMRoberta Transformer model outperformed the others with an accuracy of 82%.
2. It showcased strong precision, recall, and F1-score across multiple classes, highlighting its ability to effectively classify support tickets.
3. This model's superior performance suggests that leveraging the XLMRoberta Transformer architecture and fine-tuning it for the specific task of support ticket classification yields favorable results.

Among the models evaluated, the FineTune XLMRoberta Transformer stands out as the most effective for the support ticket classification task. Its superior performance across multiple metrics indicates its capability to handle the complexities of the dataset. The utilization of the XLM-Roberta Transformer architecture, along with fine-tuning, enhances the model's ability to discern nuances in support ticket content. The evaluation results emphasize the importance of choosing an appropriate pre-trained transformer architecture and fine-tuning it for the specific classification task at hand. The FineTuned XLMRoberta Transformer model, with its robust performance, offers promising prospects for enhancing support ticket categorization accuracy and thus for the project the final model for classification that has been used is XLMRoberta Transformer model.

7 Conclusion and Future Work

In summary, this research embarked on a comprehensive exploration of natural language processing (NLP) and machine learning (ML) methodologies, utilizing non-negative matrix factorization (NMF) for unsupervised clustering and fine-tuning transformer models—specifically, CNN+LSTM with glove embedding, DistilBERT and XLMRoberta—for supervised classification. The primary goal centered around automating the categorization of customer support tickets. The investigative journey involved meticulous steps such as data loading, preprocessing, and exploration, where NMF was employed for unsupervised clustering, revealing distinctive patterns in customer support tickets. Concurrently, transformer models underwent rigorous training, evaluation, and performance analysis, with a particular emphasis on the efficacy of CNN+LSTM, DistilBERT and XLMRoberta in precisely categorizing support tickets into predefined classes. The results underscore the significance of thoughtful model selection and fine-tuning for achieving superior performance in support ticket classification tasks. Among the evaluated models, the FineTune XLMRoberta Transformer emerged as the most promising, demonstrating the best accuracy and robust classification across diverse ticket categories.

While the research has shown efficacy in successfully deploying accurate models to the AWS cloud, it is not without limitations. Dependencies on the quality and diversity of the initial dataset and the need for periodic model updates to adapt to evolving support ticket patterns pose challenges. In essence, this research provides valuable insights into the practical application of NLP and ML techniques, laying the groundwork for further refinements and enhancements. Future work may involve seamless integration of these models into established incident management frameworks, paving the way for enhanced incident detection and resolution capabilities in cloud computing environments. Additionally, the exploration of ensemble models or hybrid approaches, combining various machine learning algorithms, could provide a more robust and adaptive solution.

References

ADOMA1, A. F. and HENRY2, N.-M. (2020). Comparative analyses of bert, roberta, distilbert, and xlnet for text-based emotion recognition, IEEE p. 5.
URL: *https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=arnumber=9317379*

Ahmed, S. and Singh, M. (2023). An empirical analysis of state-of-art classification models in an it incident severity prediction framework, MDPI .
URL: *https://www.mdpi.com/2076-3417/13/6/3843*

Chen, J. and Zhang, S. (2020). How incidental are the incidents? characterizing and prioritizing incidents for large-scale online service systems, IEEE p. 12.
URL: *https://dl.acm.org/doi/10.1145/3324884.3416624*

Chen, Y. and Yang, X. (2020a). Identifying linked incidents in large-scale online service systems, ACM p. 11.
URL: *https://dl.acm.org/doi/pdf/10.1145/3368089.3409768*

Chen, Z. and Yang, L. (2020b). Towards intelligent incident management: Why we need it and how we make it, ACM p. 11.
URL: *https://dl.acm.org/doi/pdf/10.1145/3368089.3417055*

Giachanou, A. and Bagheri, A. (2023). Towards robust online sexism detection: A multi-model approach with bert, xlm-roberta, and distilbert for exist 2023 tasks, CEUR Workshop Proceedings p. 12.
URL: *https://ceur-ws.org/Vol-3497/paper-085.pdf*

Gonzalez-Carvaja, S. (2021). Comparing bert against traditional machinelearning text classification, arXiV p. 15.

IKONOMAKIS, M. (n.d.). Text classification using machine learning techniques, IEEE p. 9.

Jain, P. (2021). Building a classification system for support requests.
URL: *https://medium.com/compass-true-north/building-a-classification-system-for-support-requests-fe3d1194cae4*

Jiazhen Gu, J. W. (2020). Efficient customer incident triage via linking with system incidents, ACM p. 12.
URL: *https://dl.acm.org/doi/pdf/10.1145/3368089.3417061*

Kirsi Tanskanen, J. K. (2023). https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4782569, IEEE p. 6.
URL: *https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4782569*

Leburu-Dingalo, T. and Ntwaagae, K. J. (2022). Application of xlm-roberta for multi-class classification of conversational hate speech, IEEE p. 6.
URL: *https://ceur-ws.org/Vol-3395/T7-12.pdf*

Li, L. and Zhang, X. (2021). Fighting the fog of war: Automated incident detection for cloud systems, Usenix (978-1-939133-23-6): 15.
URL: *https://www.usenix.org/system/files/atc21-li-liqun.pdf*

Sanh, V. and Debut, L. (2020). Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, arXiV p. 5.
URL: *https://www.arxiv-vanity.com/papers/1910.01108/*

Seliverstov, Y. A. and Komissarov, A. A. (2021). Detection of low-toxic texts in similar sets using a modified xlm-roberta neural network and toxicity confidence parameters, IEEE p. 4.
URL: *https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=arnumber=9507117*

Singh, A. (2021). Evolving with bert: Introduction to roberta.
URL: *https://medium.com/analytics-vidhya/evolving-with-bert-introduction-to-roberta-5174ec0e7c82*

Toshevska, M. and Mirceva, G. (2023). Distilbert and roberta models for identification of fake news, IEEE p. 5.
URL: *https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=arnumber=10159740*

Wubai Zhou, W. X. (2017). Star: A system for ticket analysis and resolution, ACM p. 10.
URL: *https://dl.acm.org/doi/pdf/10.1145/3097983.3098190*

Yanli Shao, Y. Z. (2021). The traffic flow prediction method using the incremental learning-based cnn-ltsm model: The solution of mobile application, hindawi p. 16.
URL: *https://doi.org/10.1155/2021/5579451*