INERA – INTELLIGENT EMAIL REPLY ASSISTANT TO REDUCE EMAIL OVERLOAD BASED ON DEEP LEARNING

Dinuka Malinda Samarasinghe

208548D

Thesis/Dissertation submitted in partial fulfillment of the requirements for the degree Master of Science in Artificial Intelligence

Department of Computational Mathematics
Faculty of Information Technology

University of Moratuwa
Sri Lanka

May 2023

DECLARATION

I declare that this is my own work and this thesis does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to University of Moratuwa the non-exclusive right to reproduce and distribute my thesis/dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

Signature: Date: 28/06/2023

The above candidate has carried out research for the Masters thesis/dissertation under my supervision. I confirm that the declaration made above by the student is true and correct.

Name of the Supervisor: Dr. Thushari Silva

ABSTRACT

Email overload is a pressing concern in modern workplaces and businesses, as it hampers productivity and efficiency. Repetitive tasks, such as responding to customer inquiries, contribute to this escalating issue. Previous literature has investigated various methods to mitigate email overload, including email prioritization, structuring, missing attachment prediction, reply suggestion, keyword summarization, and task scheduling. Despite these efforts, existing solutions exhibit notable limitations, such as reliance on manual user validation and generating responses that lack comprehensiveness and context. In this research, we developed InERA – Intelligent Email Reply Assistant, a deep learning-based solution designed to alleviate email overload. InERA generates comprehensive, context-aware responses to general inquiries by leveraging insights from previous similar emails handled by the user. The system employs a classification module to identify responses that match the current email intent, utilizing natural language processing techniques and machine learning algorithms, such as k-means clustering, Multi-Layer Perceptron classifier, and LSTM model. InERA generates meaningful replies by considering previous user responses and email thread information, ultimately reducing the need for manual intervention. The thesis is structured with chapters on Introduction, Literature Review, Technology Adoption, Approach, Design, Implementation, Evaluation, and Conclusion. These chapters detail the development, implementation, and evaluation of InERA, including its user interface design, integration with email clients, and evaluation through user testing and performance metrics. By implementing InERA, organizations can alleviate the burden of email overload, significantly improving productivity and fostering more efficient communication. Furthermore, this research highlights the broader implications and potential future enhancements of the automatic email reply assistance system in various industries and academic fields, emphasizing ethical considerations and privacy concerns.

Keywords: Email Overload, Deep learning, Feature Extraction, Unsupervised Learning

DEDICATION

To my loving mother, Renuka Wanigasinghe, who have always believed in me and supported my dreams, and to my wife, Eshani Hansamali, who have been a constant source of encouragement and inspiration. This thesis is a testament to your unwavering love and faith in me.

To my dearest friends, who have been by my side throughout this journey, providing laughter, comfort, and companionship. Your friendship has made this experience all the more rewarding. And to all those who have inspired and motivated me to pursue my passion for research and knowledge, this work is dedicated to you.

ACKNOWLEDGEMENTS

My supervisor, Dr. Thushari Silva, has provided me with tremendous advice, inspiration, and support during the course of my project, and I would like to extend my sincere gratitude to her. This thesis' completion would not have been possible without their knowledge, understanding, and mentorship. I am also grateful to the members of the Department of Computational Mathematics at University of Moratuwa for providing a stimulating and supportive environment during my studies. Special thanks go to my fellow researchers and colleagues, whose camaraderie and insightful discussions have been a source of inspiration and motivation.

I would like to extend my appreciation to the external collaborators and industry partners who contributed their time, resources, and knowledge to this research. Their input has significantly enriched the quality and practicality of the study. I owe a debt of gratitude to my family for their consistent love, patience, and support throughout my academic career. I dedicate this work to them since their confidence in me has been an ever-present source of strength for me. Also, I would want to express my gratitude to my friends for their assistance and for being present throughout both the difficult and gratifying parts of our research. Your friendship has made this experience truly enjoyable and memorable.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviation	Description
InERA	Intelligent Email Reply Assistant
MLP	Multi-Layer Perceptron
ML	Machine Learning
RNN	Recurrent Neural Network

CHAPTER 1

INTRODUCTION

1.1 Prolegomena

Email occupies a key role in the modern businesses and workplaces. Companies allocate considerable time and effort to maintain proper communication with customers and maintain reasonable customer support to provide good customer experience [1]. This effort continuously keeps increasing as the daily incoming mail count increase for businesses and cooperate users when more clients engage with the business. As this volume of email communication grow companies often fail to keep with the increase in this high volume of email with sudden growth of business this is called “Email Overload” [1]. It has been identified that email overload creates problems for personal information management and reduce productivity of the user [2]. Companies often observe repetitive work when it comes to replying to customer questions. It appears completely automate this process might lead to bad customer experience. Nonetheless, any software assistance that alleviates email overload and results in a substantial boost in productivity is immensely advantageous for the organization.

In the previous literature they have used Email prioritizing and structuring [3], Missing Attachment Prediction [4], Reply suggesting [5], Keyword summarizing [1], Task scheduling [6] to reduce problem of email overload. But there are many drawbacks identified in the literature. Such as most of the proposed solutions rely on user to validate and send the response manually and responses generated are not comprehensive enough to provide good response only from the system without user involvement. These were considered as key problems identified in previous research work in the literature.

In this research, we developed InERA – Intelligent Email Reply Assistant to reduce email overload with deep learning. The rest of the research has been structured with literature review, technology adopted, approach, design, implementation, evaluation and conclusion.

1.2 Objectives

The following objectives have been identified to develop an InERA - Intelligent Email Reply Assistance to reduce email overload based on Deep Learning.

1. Critical Review of literature in automatic email response generating and replying assistant systems
2. In depth study of technologies adopted in automatic email replying and
3. negotiations systems.
4. Design and a develop prototype for InERA - Intelligent Email Reply Assistance.
5. Evaluate the prototype.

1.3 Background and Motivation

In today's fast-paced corporate world, email remains the dominant form of communication for businesses and professionals alike. As the volume of emails continues to grow, it becomes increasingly challenging to manage inboxes effectively and respond to messages in a timely manner. The lack of response to important emails or prolonged waiting time for crucial information from colleagues can have a significant impact on overall operations, productivity, and even business relationships [1]. In this context, the concept of email prediction emerges as a vital tool to address these challenges and help users manage their time and inboxes more efficiently.

Email prediction aims to determine whether an incoming email requires an immediate response or can be deprioritized, based on factors such as the content, sender, and context of the message [1]. Existing email reply assistance solution approaches are predicated on the notion that a user's previous patterns of communication might forecast their future behavior. However, these solutions often fall short in offering comprehensive strategies that can minimize human involvement in generating replies while simultaneously providing grammatically accurate and contextually appropriate responses.

The limitations of current email prediction systems create a pressing need for a more advanced and holistic approach that not only streamlines inbox management but also generates

well-structured, contextually relevant replies with minimal human intervention. This would enable users to focus on more critical tasks and foster better communication and collaboration among team members, resulting in increased productivity and efficiency.

Additionally, an advanced email prediction system would have broad implications across various industries, as businesses of all sizes and sectors could benefit from improved email management, quicker response times, and enhanced communication. By developing such a system, companies could reduce the workload associated with email management, allowing employees to concentrate on higher-value tasks and ultimately drive business growth.

In conclusion, the development of a comprehensive email prediction system that addresses the limitations of existing solutions is of paramount importance in today's corporate landscape. By enabling more efficient inbox management and generating accurate, contextually appropriate replies with minimal human input, such a system would significantly improve communication, productivity, and overall operational efficiency, providing substantial benefits to businesses and professionals worldwide.

1.4 Problem definition

Reducing email overload using automated email reply assistance system has been a research challenge due to diversity in email contents and difficulty in maintaining good user experience. There is many research done on smart email replying and assistance systems to overcome issue of getting overloaded with emails but most of the solutions are focusing on email inbox classifying and general email reply suggestions. Increase in productivity by using these systems are low since user has to look in to all to emails to process them. Also, there are general inquires that can be easily answered using information available in the previous responses for similar inquires.

1.5 InERA to reduce email overload based on Deep Learning

In this research, we developed InERA – Intelligent Email Reply Assistant to reduce email overload with deep learning. InERA system will identify whether the content in the email can be handled

by the InERA and forward emails to the InERA system to process. The developed solution will use the text to identify Context of the emails and predict the reply based on the context and previous responses validated by the user. The predicted responses will be used to generate better grammatically correct response based on that input. All the emails handled by user will be used to generate the intent and response map using the previous email and response dataset.

1.6 Structure of the thesis

In this thesis, the chapter 1 has explained Introduction to the research elaborating objectives, problem, solution developed and structure of the research. In the chapter 2 included the critical review of the literature elaborating limitations, advantages and technologies used in the previous research studies. In the chapter 3 is about the technology adopted by the developed solution. In the chapter 4 - we have described our approach to the solution. In the chapter 5 – we have described prototype design for InERA system. Chapter 6 is about the implementation of the InERA prototype. In the chapter 7 is about the evaluation of the InERA prototype and finally chapter 8 is about the conclusion of this research.

1.7 Summary

This episode presented a comprehensive introduction to the research study on Intelligent email reply assistance using deep learning. We have delved into the background, scope, and rationale behind the research, highlighting the importance of addressing the challenges associated with email management and the potential benefits of employing deep learning techniques for this purpose.

Furthermore, we have outlined the specific objectives and expected outcomes of the study, which serve as a roadmap for the research and help to ensure that the project remains focused on achieving its goals. By identifying the key areas of investigation and the anticipated contributions to the field, we have laid the groundwork for the subsequent chapters of the dissertation. We have also discussed the organization of the dissertation, providing an overview of the structure and

content of each chapter. This information is essential for guiding readers through the research journey and ensuring a logical flow of information throughout the document.

In Chapter 2, we will present a critical review of the literature on email reply assistance systems, covering the relevant theories, concepts, and technologies that underpin the research. By examining the existing body of knowledge, we will identify gaps and opportunities in the field, which will inform the development of our Intelligent Email Reply Assistance System and contribute to the advancement of the email management domain. Overall, this introductory chapter has set the stage for the research study and established a solid foundation for the subsequent investigation and analysis of the Intelligent Email Reply Assistance System using deep learning.

CHAPTER 2

LITERATURE REVIEW ON EMAIL REPLY ASSISTANCE SYSTEMS

2.1 Introduction

This chapter provided an overall picture of the thesis by defining the research problem and identification of technology to be used in the developed solution. Critical literature review of Intelligent Email Reply Assistant systems is presented in this chapter along with the summarize the literature in terms of the contribution, limitation, challenges and the technology adopted in the existing work on literature. We have structured our literature review as Early developments of email reply assistance systems, Recent development of email reply assistance systems to overcome email overload, Gaps identified and also the justification for the research. finally, we define the research problem.

2.2 Early developments of email reply assistance systems

Modern Organizations heavily rely on emails to communicate with organization stakeholders. Email serves as a primary channel for marketing, customer support, personal archiving, task administration, and asynchronous and synchronous internet communication [2]. The rapid increase in email usage can lead to "email overload," causing many users to feel inundated by the vast amount of emails getting collected in their mail inbox [2]. Given the direct impact of email overload on the customer experience and the reputation of a company, there has been a lot of study in the field of human-computer interaction on how to reduce it [2]. It was also noticed that people handling email fall into two categories, they are prioritizers and archivers [7]. Proioratizers are manage emails as they come in. Archivers make sure they did not miss any critical messages while they save information for later use. In their study, Tyler and Tang discovered a number of variables that may affect the likely outcome of a response.

In [8], to forecast which frequently used response a user would choose while replying to an email, the authors consider the task as a semi-supervised text classification problem. They looked at the effectiveness of support vector and naïve Bayesian classifiers in finding appropriate

email answers in a business customer service department. The study also addressed the reasons behind the advantages of co-training when just a little amount of labeled data is available, as well as the usefulness of the transductive Support Vector Machine and the co-training algorithm in utilizing unlabeled data. [8].

Also, Previous studies have looked at bringing AI to email in a user-oriented way. One of that research is using keyword summarization to help user to prioritize and reduce time wastage [3]. Here, they have employed machine learning to assist three tasks, including attachment prediction, reply prediction, and summary keyword creation [4].

2.3 Recent development of email reply assistance systems

In recent advancements of email reply assistance systems, a common theme has emerged. The issue of email reply assistance is divided into two separate predictive tasks within the scope of intelligent email, aiming to alleviate email overload by enhancing email interfaces [4]. The management of email replies is made easier by the first task of email reply prediction, which may notify users when an email requires a response. The second task, Attachment Prediction, alerts users before they send an email without the intended attachment or activates a document suggestion engine to discover missing attachment emails [4]. The assessments were carried out in both single-user and cross-user scenarios, and both tasks used a similar email categorization system and task-specific characteristics.[4].

In this research In the study, Researchers solved the issue by providing users with a novel framework for email reply prediction using unsupervised learning, which helped users organize and prioritize their email communications [1]. The aim is to deliver prioritized and well-structured emails, thereby reducing the time users spend sifting through each message individually. The paper demonstrates encouraging preliminary outcomes utilizing an unsupervised machine learning model [1].

In [5], the authors introduce a novel approach for automatically creating brief email responses, known as Smart Reply. This method is currently employed in Inbox by Gmail and contributes to

10% of all mobile responses [5]. Their methodology involves mainly response selection using LSTM neural network and by choosing a set of replies from the response space that was produced using a semi-supervised graph learning technique, they developed a response set, then set of responses were selected with diversity in responses set using LSTM and a triggering model is use to decide whether to show the responses or not using feedforward neural network [5]. This was prediction using LSTM was optimized to give best approximate prediction since LSTM networks computations can be expensive [5]. A completely generative model is created by training this framework on a huge corpus of dialogue data and is capable of responding to any input text sequence.

In [6], A. Faulring, B. Myers, and A. Steinfeld presented an agent-based system named RADAR, where agents observe experts executing tasks and later assist less experienced users in performing similar tasks. The Email Classifier learns to recognize tasks within emails and scrutinizes new emails for related tasks, while the Multi-task Coordination Assistant acquires a model of the sequence in which experts complete tasks, subsequently proposing a suggested schedule for users to finish tasks [6]. An extensive evaluation of RADAR demonstrated that novice users experiencing email overload performed considerably better (with a 37% higher overall score and a fourfold reduction in errors) when supported by both agents.

2.4 Analysis of literature

When we consider the existing solution proposed by the previous literature one common attribute identified is all solutions require users' interference to reply all emails and user need to process all the emails and attend accordingly. Most of the proposed solution fail to generate full replies to emails that can provide good user experience. Literature suggests using an automated email reply system to overcome the user dependency of the assistance systems and implement better context base response generation to answer general

2.5 Problem definition

Reducing email overload using automated email reply assistance system has been a research challenge due to diversity in email contents and difficulty in maintaining good user experience. There is many research done on smart email replying and assistance systems to overcome issue of getting overloaded with emails but most of the solutions are focusing on email inbox classifying and general email reply suggestions. Increase in productivity by using these systems are low since user has to look in to all to emails to process them. Also, there are general inquires that can be easily answered using information available in the previous responses for similar inquires.

2.6 Summary

This segment depicted a critical review of literature related to the email reply assistance systems, which supported the justification of the study area, with an overview of the reviewed literature. It was presented along with advantages, limitations and technology used in major research work done in the literature. Most importantly, our research problem was defined based on the gaps identified in the previous literature. In the chapter 3 – we have presented the In-depth study of technology adapted for solving the research problem.

CHAPTER 3

TECHNOLOGY ADOPTED

3.1 Introduction

The purpose of this technology adapted chapter is to provide an overview of the various programming languages, tools, libraries, frameworks, and hardware resources employed in the development and evaluation of the Intelligent Email Reply Assistance System. The selection of appropriate technologies is crucial to the success of any research project, as it directly impacts the efficiency, scalability, and robustness of the implemented solution.

In this chapter, we will discuss the rationale behind the choice of technologies used in our research, highlighting their advantages, limitations, and suitability for the tasks involved in creating the automatic email reply assistance system. By exploring the features and capabilities of these technologies, we aim to provide a comprehensive understanding of their role in the research process and their contribution to the development of an effective and efficient solution. We will cover the following aspects in this chapter:

- Feature Extraction Methods
- Intent Classification Algorithms
- Unsupervised Learning Techniques
- Deep Learning Algorithms
- Programming languages and tools
- Deep learning libraries and frameworks
- Natural language processing libraries
- Data collection and preprocessing tools
- Machine learning libraries and algorithms
- Evaluation and visualization tools
- Email client integration and user interface development
- Hardware and computational resources

By examining the technologies adapted for this research, we hope to offer insights into the development process and demonstrate the importance of selecting the right tools and frameworks for solving complex problems in the field of deep learning and natural language processing.

3.2 Feature Extraction Techniques

In natural language processing (NLP), the process of converting unstructured text input into a structured and numerical format that machine learning algorithms can quickly comprehend and handle is known as feature extraction. The goal of feature extraction is to capture the essential information and characteristics of the text while reducing the complexity and dimensionality of the data.

There are several feature extraction techniques commonly used in NLP, including:

- Bag-of-words (BoW): The BoW method counts the number of times each word appears in a given document. It disregards the order of the words and focuses solely on their occurrence. The resulting feature vector has a fixed length, equal to the size of the vocabulary, with each element representing the frequency of a specific word.

- Term Frequency-Inverse Document Frequency (TF-IDF): TF-IDF is an improvement over the BoW representation. It not only considers the frequency of words in a document (term frequency) but also accounts for the rarity of words across the entire corpus (inverse document frequency) [9]. This results in a more informative representation that gives higher importance to rare but meaningful words.

- Word embeddings: The semantic meaning of words is captured by word embeddings, which are dense vector representations in a continuous vector space. Techniques like Word2Vec, GloVe, and FastText generate word embeddings by analyzing the co-occurrence patterns of words in large text corpora. These embeddings can capture semantic

relationships between words and are more efficient than sparse representations like BoW and TF-IDF.

- Pre-trained language models: Pre-trained language models, such as BERT, GPT, and RoBERTa, have become popular for feature extraction in NLP tasks. These models learn contextualized word representations by training on massive text corpora and can be fine-tuned for specific tasks or used as feature extractors. These models provide embeddings that are rich in semantic information, which has enabled them to achieve cutting-edge results in a variety of NLP tasks.

The quality of the extracted features may have a big influence on how well machine learning models perform, hence feature extraction is a critical stage in NLP. By capturing relevant information from the text data and representing it in a structured format, feature extraction enables machine learning algorithms to understand, process, and make predictions based on textual data.

3.3 Intent Classification Algorithms

Intent classification is a one of the key operation in natural language processing that involves determining the underlying purpose or goal of a user's text input, such as a question or command. This task is critical for applications like chatbots, virtual assistants, and customer support systems, as it allows them to understand and respond appropriately to user requests. Various machine learning and deep learning classifiers, including logistic regression, support vector machines, and neural networks, can be employed for intent classification by learning patterns in the input text and mapping them to corresponding intent labels, enabling accurate and context-aware responses [10]. Various machine learning classifiers can be employed for this task, and some of the popular choices are:

- Logistic Regression: Logistic Regression is a simple, linear model often used for binary or multi-class classification problems. It learns the relationship between input features and class labels by estimating the probabilities using a logistic (sigmoid) function for binary

classification or a softmax function for multi-class classification. Logistic Regression is computationally efficient.

- Support Vector Machine (SVM): SVM is a powerful classifier that constructs a hyperplane or a set of hyperplanes in a high-dimensional space to separate the classes. It aims to maximize the margin between the classes, which can lead to improved generalization performance. SVM can handle linear and non-linear classification problems using appropriate kernel functions, such as the linear, polynomial, or radial basis function (RBF) kernel.

- Naive Bayes: Based on the Bayes theorem and the presumption of independence between input characteristics, Naive Bayes is a probabilistic classifier. Despite its simplicity and the strong independence assumption, Naive Bayes often performs surprisingly well in text classification tasks, including intent classification, because it can effectively model the conditional probabilities of words given a particular class.

- Random Forest: The training phase of the ensemble learning method Random Forest involves the construction of several decision trees, and the class label that is generated is the average of the class labels predicted by all of the trained trees. Random Forest is robust to overfitting, can handle a large number of features, and generally achieves high classification accuracy.

- Multi-Layer Perceptron (MLP): A feedforward artificial neural network called an MLP has three layers: an input layer, one or more hidden layers, and an output layer. It can model complex non-linear relationships between input features and class labels using non-linear activation functions such as ReLU, sigmoid, or tanh. MLP can adaptively learn representations of the input features, making it suitable for various classification tasks, including intent classification.

- Convolutional Neural Networks (CNN): While primarily used for image classification, CNNs can also be employed for text classification tasks. By applying convolutional filters on word embeddings, CNNs can capture local patterns and n-grams in the input text. These features are then aggregated using pooling layers and fed into fully connected layers for classification.

- Recurrent Neural Networks (RNN) and its variants (LSTM, GRU): RNNs are designed to capture sequential information in input data, making them a natural choice for text classification tasks. Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) are two popular RNN variants that can mitigate the vanishing gradient problem, allowing them to capture long-range dependencies in the input text effectively.

- Transformer-based models (e.g., BERT, GPT, DistilBERT): Transformer-based models have achieved state-of-the-art performance in various NLP tasks, including intent classification. These models use self-attention mechanisms to capture contextual information in the input text and can be fine-tuned for specific classification tasks. Pre-trained models like BERT and DistilBERT can be used as feature extractors or fine-tuned with additional classification layers to achieve high accuracy in intent classification tasks.

The choice of the classifier for intent classification depends on factors like the size and complexity of the dataset, the available computational resources, and the desired model performance. Often, it is beneficial to experiment with multiple classifiers and select the one that performs best on the validation dataset.

From the above classifiers, A feedforward artificial neural network with an input layer, many hidden layers, and a final output layer is represented by a multilayer perceptron (MLP) structure. The number of neurons in each layer is many, and these neurons are connected to neurons in adjacent layers through weighted connections. The MLP learns by adjusting these weights during the training process.

MLP represented using equations:

1. Input layer: Assume we have an input feature vector x = [x1, x2, ..., xn]^T, where T denotes the transpose operation.

2. Hidden layers: Each neuron in the hidden layers computes a weighted sum of its inputs and applies an activation function. Let's assume there's a single hidden layer with m neurons. The output of the j-th neuron in the hidden layer can be represented as:

   $h_j = f(\Sigma(w_{ij} * x_i) + b_j)$

   where f(.) is the activation function (e.g., ReLU, sigmoid, or tanh), w_{ij} is the weight connecting input i to hidden neuron j, and b_j is the bias term for hidden neuron j.

3. Output layer: Similar to the hidden layer, the output layer computes a weighted sum of its inputs (the outputs of the hidden layer) and applies an activation function. For a multi-class classification problem with k classes, the output of the p-th neuron in the output layer can be represented as:

   $o_p = g(\Sigma(w_{jp} * h_j) + b_p)$

   where g(.) is the activation function (e.g., softmax for multi-class classification), w_{jp} is the weight connecting hidden neuron j to output neuron p, and b_p is the bias term for output neuron p.

The training of an MLP consists of adjusting the weights (w_{ij} and w_{jp}) and biases (b_j and b_p) to minimize a loss function that measures the discrepancy between the predicted outputs and the true labels. This optimization process is typically performed using gradient-based methods like stochastic gradient descent (SGD) or adaptive optimization algorithms such as Adam.

The solver is responsible for updating the model parameters during training. For example, let's consider the Adam optimizer. Utilizing the first and second moments of the gradients, it calculates adaptive learning rates for each weight and bias:

1. Initialize the first moment estimates (m_t) and second moment estimates (v_t) to 0.

2. At each training iteration t, compute the gradients (g_t) of the loss function with respect to the weights and biases.

3. Update the first and second moment estimates as follows:

$$m\_t = \beta1 * m\_(t-1) + (1 - \beta1) * g\_t$$
$$v\_t = \beta2 * v\_(t-1) + (1 - \beta2) * (g\_t) \text{^}2$$

where β1 and β2 are the exponential decay rates for the first and second moment estimates.

4. Correct the bias in the first and second moment estimates:

$$m\_t\_hat = m\_t / (1 - \beta1\text{^}t)$$
$$v\_t\_hat = v\_t / (1 - \beta2\text{^}t)$$

5. Update the model parameters (weights and biases) using the corrected first and second moment estimates:

$$w\_t+1 = w\_t - \alpha * (m\_t\_hat / (sqrt(v\_t\_hat) + \varepsilon))$$

where α is the learning rate and ε is a small constant to prevent division by zero.

By updating the model parameters using the solver, the MLP learns to minimize the loss function and improve its performance on the given task.

3.4 Unsupervised Learning Techniques

Unsupervised learning is a type of machine learning where algorithms learn from and identify patterns in data without the guidance of labeled examples. The main objective of unsupervised learning is to uncover hidden associations, connections, or trends present in the data. In NLP, unsupervised learning techniques are often used to analyze text data, extract meaningful features, and uncover hidden semantics [11]. Several frequently employed unsupervised learning methods in the domain of natural language processing encompass:

- Clustering: Clustering algorithms group similar data points together based on their features, without using any predefined labels. In NLP, clustering techniques like k-means, hierarchical clustering, and DBSCAN can be applied to text data to identify groups of similar documents or to discover common topics within a corpus [12]. This can be useful for tasks like document organization, summarization, and topic modeling.

- Topic Modeling: An unsupervised method known as "topic modeling" is used to find abstract subjects that appear in a group of texts. Latent Dirichlet Allocation (LDA) is a popular topic modeling algorithm that represents documents as a mixture of topics, with each topic characterized by a probability distribution over words [13]. Topic modeling can be applied to text data for tasks like content categorization, information retrieval, and document summarization.

- Dimensionality Reduction: Techniques for reducing dimensionality, like Principal t-Distributed Stochastic Neighbor Embedding (t-SNE) and Component Analysis (PCA), serve to decrease the dimensions of the feature space while maintaining the fundamental organization of the information. In NLP, dimensionality reduction can be applied to visualize high-dimensional text data, improve the efficiency of machine learning models, and reveal relationships between words or documents [14].

- Word Embeddings: Unsupervised learning methods like Word2Vec, GloVe, and FastText can be used to generate word embeddings, which are dense vector representations that capture the semantic meaning of words in a continuous vector space. These algorithms learn word embeddings by analyzing the co-occurrence patterns of words in large text corpora [12]. Word embeddings can then be used as input features for various NLP tasks, such as text classification, sentiment analysis, and named entity recognition.

- Autoencoders: Autoencoders are unsupervised neural network models used for dimensionality reduction and feature learning [15]. They consist of an encoder, which maps the input data to a lower-dimensional latent space, and a decoder, which reconstructs the input data from the latent space. In NLP, autoencoders can be employed for tasks like text compression, denoising, and generating pre-trained features for supervised learning tasks.

Unsupervised learning techniques play a crucial role in NLP, as they enable the discovery of hidden structures and patterns within text data without the need for labeled examples [11]. These techniques can be applied to various NLP tasks and can complement supervised learning approaches by providing valuable insights and features for more effective modeling.

K-Mean clustering is one of the key unsupervised clustering technique used in NLP. And In this research, we have used K-Mean clustering for email clustering. Based on the distances between the data points and the cluster centroids, the k-means clustering method seeks to divide a dataset into K distinct, non-overlapping clusters. [12]. The algorithm can be summarized using the following notations and equations:

- Let X = {x1, x2, ..., xN} be the dataset containing N data points, where each data point xi has d dimensions.
- Let K be the number of clusters we want to partition the dataset into.
- Let C = {c1, c2, ..., cK} represent the centroids of the K clusters, where each centroid cj is a d-dimensional vector.

The k-means clustering algorithm proceeds as follows:

- Initialize the K centroids randomly by selecting K data points from the dataset X.

- Assign each data point xi to the nearest centroid cj by minimizing the distance metric, usually the Euclidean distance:

$$dist(xi, cj) = ||xi - cj||^2$$

- assignment(xi) = argmin(dist(xi, cj)) for all j in {1, 2, ..., K}
- Update the centroids by calculating the mean of all the data points assigned to each centroid:

$$cj = (1/|Sj|) * \Sigma_{(xi \in Sj)} xi, \text{ where } Sj = \{xi \mid assignment(xi) = j\}$$

- Keep repeating steps 2 and 3 until the centroids' locations remain stable or a predetermined maximum number of iterations has been achieved.

The objective of the k-means algorithm is to minimize the within-cluster sum of squares (WCSS), which can be represented as:

$$WCSS = \Sigma_{(j=1 \text{ to } K)} \Sigma_{(xi \in Sj)} ||xi - cj||^2$$

In conclusion, K-means clustering is an effective approach for clustering emails and identifying email classes due to its scalability, simplicity, ability to work in high-dimensional feature spaces, unsupervised learning nature, flexibility, and its usefulness in building datasets for intent classification models. These characteristics make K-means clustering a suitable choice for clustering emails and preparing data for an intent classification system.

3.5 Deep Learning Algorithms

Deep learning methods form a subcategory of machine learning approaches that utilize multi-layered artificial neural networks to model intricate patterns and representations within data. These

algorithms are particularly effective at handling large-scale, high-dimensional data, making them well-suited for various tasks in NLP, computer vision, and speech recognition [16]. In the context of content generation, deep learning algorithms have shown remarkable capabilities in generating human-like text, images, and audio. Some popular deep learning architectures and techniques used for content generation include:

- Recurrent Neural Networks (RNNs): RNNs are a subclass of neural networks created specifically to handle sequential input by preserving a hidden internal state that can store knowledge from earlier time steps. As they can model long-range relationships and produce context-aware content, RNNs, particularly its variations like Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU), are frequently employed in NLP for tasks including language modeling, machine translation, and text synthesis. [17].

- Sequence-to-Sequence (Seq2Seq) Models: Seq2Seq models are a type of deep learning architecture that consist of an encoder network, which processes the input sequence, and a decoder network, which generates the output sequence. These models are commonly used for tasks like machine translation, summarization, and dialogue generation, as they can learn to map input sequences to output sequences in a context-dependent manner.

- Transformers: Transformers are a more recent deep learning architecture that leverage self-attention mechanisms to process input sequences in parallel, rather than sequentially. This allows transformers to model long-range dependencies more effectively than RNNs, leading to improved performance on various NLP tasks. Pre-trained transformer models like BERT, GPT, and RoBERTa have been used for content generation tasks like text completion, summarization, and dialogue generation, often producing highly coherent and contextually relevant content [17].

- Variational Autoencoders (VAEs) and Generative Adversarial Networks (GANs): VAEs and GANs are deep learning techniques used for generating new data samples that resemble a given dataset. VAEs learn a probabilistic latent space that captures the underlying

structure of the data, while GANs consist of a generator network, which creates synthetic data samples, and a discriminator network, which distinguishes between real and generated samples. Both VAEs and GANs have been applied to content generation tasks in NLP, such as text synthesis, paraphrasing, and style transfer, as well as image and audio generation [18].

Deep learning algorithms have revolutionized content generation by enabling the creation of high-quality, context-aware, and human-like content across various modalities. These algorithms continue to advance and improve, paving the way for more sophisticated and creative applications in content generation and beyond.

3.6 Programming Languages and Tools

Python will be mainly used for developing the backend NLU and ML implementations and NodeJS will be used to develop admin panel for email user. And for frontend development is done using bootstrap, CSS, and JavaScript.

Python has emerged as a leading programming language, particularly in the realm of data science, artificial intelligence, and machine learning applications, making it a highly suitable choice for the backend development of the intelligent email reply assistance system. Renowned for its clean syntax and ease of use, Python allows for efficient development and maintenance of codebases, while offering an extensive array of libraries and frameworks tailored for AI and NLP applications, such as TensorFlow, PyTorch, Keras, and NLTK. This rich ecosystem, coupled with the vast community support, ensures that developers can effectively implement advanced algorithms and models for the INERA system, while benefiting from shared knowledge and resources.

Python's platform independence ensures that the developed system can be executed across different operating systems with minimal modifications, facilitating compatibility and simplifying deployment in diverse environments. Additionally, Python boasts excellent integration capabilities with various data storage solutions, web services, and APIs, enabling the system to seamlessly

connect to email servers, fetch and analyze email data, and generate contextually appropriate responses. The PyCharm Integrated Development Environment (IDE) further enhances the development process by providing an intuitive interface and powerful features, such as code completion, debugging tools, and built-in version control, thereby boosting productivity and ensuring code quality.

Throughout this research, Jupyter Notebook has been extensively used as a powerful tool for evaluating models, optimizing parameters, and exploring alternative approaches. Its interactive environment, combining live code, visualizations, and narrative text, has facilitated rapid experimentation and analysis of various deep learning models and hyperparameters, enabling us to iteratively fine-tune models for the intelligent email reply assistance system.

Jupyter Notebook's organized workspace has aided in examining and comparing alternative methodologies, allowing for a systematic approach to assessing each method. Its ability to generate visualizations and summary statistics has streamlined result evaluation, making it easier to identify promising solutions and select the best-performing model. Jupyter Notebook's isolated code cells have allowed for quick adjustments and real-time feedback, greatly aiding hyperparameter optimization and model selection.

3.7 Deep Learning Libraries and Frameworks

TensorFlow and Keras are powerful, user-friendly deep learning libraries that provide an excellent platform for implementing LSTM models. Keras, a high-level neural network API, runs on top of TensorFlow and offers an intuitive interface for designing and training deep learning models, making it easier for researchers and practitioners to develop complex LSTM networks [16]. Both TensorFlow and Keras come with built-in support for LSTMs, as well as other RNN variants like GRUs, enabling flexibility and adaptability for various applications. The large community, GPU support, and scalability make TensorFlow and Keras ideal choices for implementing LSTM models in research and real-world projects. TensorFlow and Keras are suitable libraries for implementing an LSTM model for InERA system for several reasons:

1. Ease of use: Keras is a high-level neural networks API that runs on top of TensorFlow. It is designed to be user-friendly, modular, and easy to extend. Keras provides a simple and consistent interface, making it easier to develop and experiment with deep learning models like LSTM. It enables you to build and train models quickly, which is valuable for research purposes.

2. Flexibility: TensorFlow offers a wide range of pre-built layers and models, including LSTM, GRU, and other RNN variants. This flexibility allows you to experiment with different architectures and find the most suitable model for your research. Additionally, TensorFlow supports custom layers and models if you need to implement more advanced techniques.

3. Large community and resources: TensorFlow and Keras have an extensive community, which means that you can find numerous tutorials, examples, and open-source projects to learn from and adapt for your research. This support can save you time and effort while working on your project.

4. GPU support: TensorFlow has built-in support for GPU acceleration, which can significantly speed up the training process of your LSTM model. This feature is especially beneficial for research purposes, as it allows you to test multiple configurations and ideas quickly.

5. Scalability: TensorFlow is designed to be highly scalable, enabling you to train your LSTM model on large datasets. It also supports distributed training, which allows you to leverage multiple GPUs or even multiple machines to speed up the training process further. This scalability is crucial when working with large email datasets.

6. Model deployment and integration: TensorFlow provides tools like TensorFlow Serving and TensorFlow Lite, which facilitate the deployment of your trained LSTM model to

various platforms such as web servers, mobile devices, and embedded systems. This capability can be useful if you plan to integrate your email reply assistant with other applications or platforms.

In summary, TensorFlow and Keras are suitable libraries for implementing an LSTM model for your MSc research of an automatic email reply assistant due to their ease of use, flexibility, large community support, GPU acceleration, scalability, and model deployment capabilities. These features can help you develop, experiment, and deploy your email reply assistant effectively and efficiently.

3.8 Natural Language Processing Libraries

In the development of the Intelligent Email Reply Assistant (InERA) system, multiple NLP libraries have been employed to achieve the desired results. This section discusses the NLP libraries used in the research and provides justification for their selection.

The NLP libraries utilized in this research include:

1. spaCy: This library is a popular and high-performance NLP library that offers a wide range of functionalities, such as tokenization, part-of-speech tagging, named entity recognition, and dependency parsing. In InERA, spaCy has been used for tokenization of text, which is a crucial step in preprocessing the email content for further analysis.

2. Huggingface Transformers: Huggingface Transformers is a state-of-the-art NLP library that provides pre-trained models for various tasks, such as text classification, text generation, and translation. In this research, Huggingface Transformers has been used to extract word embeddings from the text, which are then used as input features for the subsequent clustering and classification tasks.

3. Scikit-learn: Scikit-learn provides a variety of algorithms for operations including classification, regression, and clustering. In InERA, scikit-learn has been used for creating the Multi-Layer Perceptron model for intent classification as well as for K-means clustering.

The choice of these NLP libraries is justified based on their relevance to the research, their ease of use, and their performance. SpaCy is known for its efficiency and speed, which is critical for processing large amounts of email data. Huggingface Transformers provides access to state-of-the-art pre-trained models, which can significantly enhance the quality of word embeddings and, consequently, the performance of the InERA system. Scikit-learn is a well-documented and easy-to-use library that offers a wide range of algorithms, making it suitable for implementing the clustering and classification tasks in this research.

In conclusion, the NLP libraries used in this research have been carefully selected based on their relevance to the tasks at hand, their performance, and their ease of use. These libraries have played a vital role in the development of the InERA system and have significantly contributed to the success of the research.

3.9 Data Collection and Preprocessing Tools

In the development of the Intelligent Email Reply Assistant (InERA) system, the collection and preprocessing of data play a crucial role in ensuring the efficiency and accuracy of the system's outcomes. This section provides an overview of the tools and libraries used for data collection and preprocessing, and explains the rationale behind their selection.

The tools and libraries utilized in this research for data collection and preprocessing include:

1. Pandas: Pandas is a widely-used open-source data manipulation library for Python. It provides data structures and functions needed to work with structured data seamlessly. In InERA, Pandas has been employed to manage and preprocess the email data, including

tasks such as filtering, cleaning, and transforming the data into a suitable format for further analysis.

2. NLTK (Natural Language Toolkit): NLTK is a comprehensive library for natural language processing in Python. It offers a wide range of tools for text preprocessing, such as tokenization, stemming, and stopword removal [15]. In InERA, NLTK has been used for initial text preprocessing tasks to prepare the email data for further analysis.

3. Spacy: Spacy is a high-performance natural language processing library for Python. In InERA, Spacy has been employed for more advanced text preprocessing tasks, such as part-of-speech tagging, dependency parsing, and named entity recognition. It has also been used for tokenization and extracting word embeddings from the text.

4. Regular Expressions (regex): Regular expressions are a powerful tool for pattern matching and manipulation of text data. In InERA, regex has been utilized for preprocessing tasks, such as removing unwanted characters, extracting specific information from the text, and cleaning up email content.

The choice of these tools and libraries for data collection and preprocessing is driven by their benefits in the context of the research:

1. Pandas: The selection of Pandas as a data manipulation library is based on its ease of use, extensive documentation, and robust capabilities. Pandas provides various functionalities that simplify the process of handling large datasets, such as the ability to perform operations on data in a vectorized manner, support for handling missing data, and advanced querying capabilities. Additionally, Pandas is highly compatible with other data analysis libraries, such as NumPy and Scikit-learn, which further enhances the efficiency of the data processing pipeline in InERA.

2. NLTK: NLTK has been chosen for its comprehensive set of tools and resources for natural language processing. Its wide range of text preprocessing functionalities and extensive documentation make it a valuable asset for this research, allowing researchers to efficiently preprocess the email data and prepare it for further analysis.

3. Spacy: The choice of Spacy is justified by its high-performance capabilities and ease of use. Spacy offers advanced NLP features that are not available in NLTK, which makes it a valuable addition to the data preprocessing pipeline. Moreover, Spacy is highly compatible with other Python libraries, which further streamlines the development process.

4. Regular Expressions (regex): Regex was chosen for its versatility and expressiveness, allowing researchers to perform complex text manipulation tasks with ease. Its inclusion in the preprocessing pipeline ensures that the email data is clean and free of unwanted elements, which is crucial for the accuracy of the InERA system.

In conclusion, the tools and libraries employed in this research for data collection and preprocessing have been carefully chosen based on their relevance to the tasks, their performance, and their ease of use. These tools have played a critical role in ensuring that the InERA system is built on a solid foundation of clean, well-structured data, which in turn has significantly contributed to the success of the research.

3.10 Machine Learning Libraries and Algorithms

In the development of the Intelligent Email Reply Assistant (InERA) system, various machine learning libraries and algorithms were employed to facilitate effective email management and accurate reply generation. This section provides an overview of the machine learning libraries and algorithms used in the research and explains the rationale behind their selection. The following machine learning libraries and algorithms are utilized in this research.

- Scikit-learn: Scikit-learn is a popular open-source machine learning library for Python that provides a wide range of tools for data preprocessing, model training, and evaluation. It has been employed in InERA for various tasks, such as intent classification using Multi-Layer Perceptron model and email clustering using K-means clustering.

- K-means clustering: An unsupervised learning approach called K-means clustering divides a dataset into K distinct, non-overlapping groups based on similarity. In InERA, K-means clustering has been applied to group similar emails together, allowing the system to identify patterns and generate contextually appropriate replies.

- Multi-Layer Perceptron model: A feedforward artificial neural network made up of numerous layers of linked neurons is called a multi-layer perceptron (MLP). The MLP model has an input layer for receiving data, one or more hidden layers for transforming the data, and an output layer for generating predictions. In InERA, Multi-Layer Perceptron model has been used for intent classification, determining the appropriate response category for incoming emails.

The choice of these machine learning libraries and algorithms is driven by their benefits in the context of the research:

- Scikit-learn: Scikit-learn was chosen due to its comprehensive set of tools for machine learning and data analysis, its ease of use, and its extensive documentation. The library is highly compatible with other Python libraries, such as Pandas and NumPy, which further streamlines the development process. Its wide range of algorithms and utilities makes it suitable for implementing various components of the InERA system, such as intent classification and email clustering.

- K-means clustering: The selection of K-means clustering is based on its simplicity, efficiency, and effectiveness in partitioning data into meaningful groups. Its ability to

identify patterns in large datasets makes it suitable for the task of grouping similar emails in InERA, which in turn helps generate more accurate and contextually appropriate replies.

- Multi-Layer Perceptron Classifier: The interpretability, simplicity, and efficiency of the Multi-Layer Perceptron Classifier were factors in its selection for binary and multi-class classification issues. It has proven to be a suitable choice for intent classification in InERA, as it can effectively model the relationship between email content and the corresponding response category.

In conclusion, the machine learning libraries and algorithms employed in this research were carefully chosen based on their relevance to the tasks, their performance, and their ease of use. These technologies have played a critical role in the development and success of the InERA system, enabling it to efficiently manage emails and generate accurate, contextually appropriate replies. The selection of these libraries and algorithms also ensures that the system remains flexible and adaptable to potential future enhancements and modifications.

3.11 Evaluation and Visualization Tools

In the Intelligent Email Reply Assistant (InERA) research, various evaluation and visualization tools were employed to effectively analyze the results and present the research findings. This section provides an overview of the tools and libraries used for evaluation and visualization and explains the rationale behind their selection. Following are the tools and libraries utilized for evaluation and visualization in this research.

- **Matplotlib**: Matplotlib is a widely used Python library for creating static, animated, and interactive visualizations. It was employed in InERA to generate various plots and charts for visualizing the performance and results of the machine learning models and clustering algorithms.

- **Seaborn**: Seaborn, a data visualization library in Python built upon Matplotlib, offers a user-friendly interface for generating visually appealing and informative statistical visuals. InERA leverages Seaborn to generate advanced visualizations for better understanding and interpretation of the research findings.

- **Plotly**: Plotly is an interactive, open-source graphing library for Python that enables the creation of visually appealing and interactive plots. InERA uses Plotly to create dynamic visualizations that can be easily explored and manipulated, providing deeper insights into the research outcomes.

The choice of these evaluation and visualization tools is driven by their benefits in the context of the research:

- **Matplotlib**: Matplotlib was chosen due to its versatility, ease of use, and compatibility with other Python libraries. It offers a wide range of plotting functions and customization options, allowing for the creation of tailored visualizations that effectively communicate the research findings.

- **Seaborn**: Seaborn was selected for its ability to create visually appealing and informative statistical graphics with minimal code. Its high-level interface simplifies the process of creating complex visualizations, and its integration with Pandas and NumPy facilitates efficient data processing and analysis.

- **Plotly**: Plotly was chosen for its capability to generate interactive plots that enable users to explore the research results more intuitively. The library's interactive features, such as tooltips and zooming, allow for a more engaging and insightful presentation of the research findings.

The evaluation and visualization tools employed in this research were carefully chosen based on their ability to effectively present the research outcomes, their compatibility with other Python libraries, and their ease of use. These tools play a crucial role in analyzing and interpreting the results of the InERA system, allowing for a more comprehensive understanding of its performance and impact on email management efficiency and user productivity. The selection of these tools also ensures that the research findings can be effectively communicated to a diverse audience, including stakeholders and fellow researchers.

3.12 Email Client Integration and User Interface Development

In the Intelligent Email Reply Assistant (InERA) research, seamless email client integration and user interface development were crucial components to ensure a smooth and efficient user experience. This section provides a description of the tools and libraries used for email client integration and an overview of the technologies employed for user interface development.

Email Client Integration:

To facilitate email client integration, InERA utilized the following tools and libraries:

1. IMAP (Internet Message Access Protocol): IMAP is a common protocol for the Internet that is used to access and manage messages on a mail server. InERA leveraged IMAP to retrieve email messages and their associated metadata from the user's email account, enabling the system to process and analyze incoming emails for automatic reply generation.

2. SMTP (Simple Mail Transfer Protocol): An Internet standard for email transmission over IP networks is called SMTP. InERA employed SMTP to send the generated email replies back to the recipients, ensuring a seamless integration with the user's existing email workflow.

3. Email Libraries: Python's built-in email libraries, such as "email" and "smtplib", were used in InERA to parse, construct, and send email messages. These libraries simplified the process of handling various email components, such as headers, body text, and attachments, allowing the system to focus on the core functionality of generating contextually relevant email replies.

User Interface Development:

The user interface of InERA was developed using a combination of modern web technologies to create an intuitive and responsive interface. These technologies include:

- HTML (Hypertext Markup Language): HTML was used as the foundational markup language for structuring the content of the user interface. It provided the basic structure and organization of the interface components.

- CSS (Cascading Style Sheets): CSS was employed for the presentation and layout of the user interface, ensuring a visually appealing and consistent design across the application. CSS allowed for the customization of colors, fonts, and other stylistic elements, enhancing the overall user experience.

- JavaScript: JavaScript is a widely used programming language for web development, was utilized to add responsive, robust and dynamic functionality to the InERA user interface. JavaScript enabled the system to respond to user input, update the interface, and communicate with the backend server without requiring a page reload.

- Bootstrap: Bootstrap, a widely-used open-source CSS framework, was used to create a responsive and mobile-friendly user interface for InERA. Bootstrap provided a collection of pre-designed UI components, such as navigation bars, buttons, and forms, which expedited the development process and ensured a consistent visual design. The framework

also included a grid system, making it easy to create a flexible and adaptable layout for various screen sizes and devices.

In summary, the tools and technologies employed for email client integration and user interface development in InERA were chosen based on their ability to facilitate seamless communication with email servers, as well as their potential to create an intuitive and visually appealing user interface. The combination of these technologies ensured a smooth and efficient user experience, enabling users to easily interact with the InERA system.

3.13 Hardware and Computational Resources

The Intelligent Email Reply Assistant (InERA) system requires a combination of software and hardware resources to ensure smooth functioning and efficient performance. Below are the recommended system specifications for InERA:

Hardware Specifications:

- CPU: Intel Core i5 or equivalent (6th generation or later) with at least 4 cores.
- Memory: 16 GB RAM or higher.
- GPU: NVIDIA GeForce GTX 1060 or equivalent with at least 6 GB of dedicated video memory. Alternatively, a higher-end GPU, such as an NVIDIA GeForce RTX 20xx/30xx series or NVIDIA A100, can be used for more demanding workloads and faster model training.
- Storage: At least 256 GB SSD for faster read and write speeds, with additional storage space (HDD or SSD) based on the size of the email datasets being used.
- Network: High-speed internet connection for accessing cloud resources and email servers.

Software Specifications:

- Operating System: Windows 10, macOS, or Linux (Ubuntu 18.04 or later).
- Programming Language: Python 3.7 or later, with appropriate libraries and packages installed (e.g., TensorFlow, Keras, scikit-learn, spaCy, NLTK, pandas, IMAP, SMTP, email, etc.).
- Integrated Development Environment (IDE): Jupyter Notebook, PyCharm, Visual Studio Code, or any other preferred Python IDE.
- Web Development Tools: HTML, CSS, JavaScript, and Bootstrap for user interface development.
- Data Visualization Libraries: Matplotlib, Seaborn, or Plotly for generating plots and visualizations.

Cloud Specifications (optional):

- Cloud Computing Services: Microsoft Azure or Google Cloud Platform (GCP) for accessing virtual machines with dedicated GPUs, storage solutions, and other cloud-based resources.

Please note that the system specifications mentioned above are recommended for the InERA system based on typical requirements. However, depending on the size of the email datasets, complexity of the models, and specific research needs, the actual hardware and software requirements may vary.

The choice of hardware and computational resources had a significant impact on the research process and results of the InERA project:

- Accelerated Model Training: Utilizing GPUs and cloud computing services enabled faster model training and experimentation, facilitating the exploration of a wider range of model architectures and hyperparameter settings. This accelerated training process allowed for

more thorough evaluations and improvements, ultimately resulting in higher-performing models.

- Scalability: The adoption of cloud computing services provided the flexibility to scale up or down the computational resources according to the requirements of the research. This scalability ensured that the research process was cost-effective and efficient, as resources could be allocated based on the current needs of the project.

- Reproducibility: The use of cloud computing services also improved the reproducibility of the research, as the same computational environment could be easily shared with other researchers. This made it easier for others to replicate the experiments, validate the findings, and build upon the research.

- Collaboration: Cloud-based storage and computing resources facilitated seamless collaboration between researchers, enabling efficient sharing of data, models, and results. This improved collaboration allowed for quicker iterations and feedback, ultimately leading to better research outcomes.

In conclusion, the hardware and computational resources utilized in the InERA research, including GPUs and cloud computing services, played a crucial role in accelerating the research process and enhancing the quality of the results. These resources not only facilitated faster model training and evaluation but also improved scalability, reproducibility, and collaboration within the research team.

3.14 Conclusion

In the technology adopted chapter, we presented an overview of the key techniques and algorithms that have been utilized in the development of the Intelligent Email Reply Assistant (InERA) system. The chapter explored various technologies, including K-means clustering, Multi-Layer

Perceptron Classifier, LSTM model, and TF-IDF vectorization, which were carefully selected to address the research problem of email overload effectively.

The importance of the chosen technologies and their contribution to the success of the research cannot be overstated. Each of these techniques played a vital role in various aspects of the InERA system, such as clustering emails, classifying intents, generating contextually appropriate email replies, and representing email content as feature vectors. The effective integration and optimization of these technologies enabled the development of a robust and efficient system that significantly reduces the burden of email overload on users and improves their productivity.

While the chosen technologies have proven to be effective in addressing the research problem, it is essential to acknowledge the existence of potential alternative technologies that could have been considered. For instance, other clustering algorithms, such as hierarchical clustering or DBSCAN, could have been explored. Similarly, alternative classification algorithms, such as support vector machines or random forests, might have been investigated. Furthermore, more advanced text generation models, such as GPT-3 or BERT, could have been employed for generating email replies. The choice of alternative technologies would depend on various factors, such as their suitability for the problem, scalability, and ease of implementation.

In conclusion, the technology adopted chapter presented a comprehensive overview of the key techniques and algorithms that have been used to develop the InERA system. The chapter highlighted the importance of these technologies in achieving the research goals and provided a brief discussion on potential alternative technologies that could have been relevant to the research problem. The successful integration of the chosen technologies has enabled the development of an effective solution to the email overload problem, ultimately contributing to the overall success of the research.

CHAPTER 4

APPROACH TO INTELLIGENT EMAIL REPLY ASSISTANT SYSTEM

4.1 Introduction

In Chapter 3, we conducted an in-depth examination of the technology used to address the research problem. In this chapter, we present our approach to the development of InERA – the Intelligent Email Reply Assistant, designed to harness the power of deep learning for generating automatic email replies and mitigating email overload. Our hypothesis posits that by employing deep learning techniques, we can create an intelligent system capable of generating contextually relevant, grammatically accurate, and coherent email responses, reducing the burden of email management on users. InERA's features include understanding email context, generating relevant replies, and seamless integration with existing email clients, while offering user-friendly interfaces and customization options. Catering to a diverse user base, including corporate email users and software developers, InERA aims to revolutionize email management and enhance communication efficiency across various sectors. This section is structured as details of our hypothesis, input, output, process, features, and users of InERA are as follows.

4.2 Hypothesis of Intelligent Email Reply Assistant

Our project hypothesizes that reducing email overload problem can be modeled by using an Intelligent email reply assistant based on deep learning. The inspiration behind this hypothesis comes from evidence found on literature that shows the use automated email reply system approach would be beneficial to overcome the drawback in other solutions proposed in the literature.

4.3 Input

The main input to the system is the dataset used to train the classifiers and the deep learning model. The dataset used for this is the ATIS dataset from Kaggle. The Airline Travel Information

System (ATIS) is a commonly employed benchmark dataset in the field of intent classification, serving as a standard reference point. This is a labeled publicly available dataset. This dataset contains 19900 emails received to Airline Travel Information System. The data set file is in CSV format. Emails are mainly customer inquiries from Airline support staff and contain portion of spam emails as well. Dataset is labeled with 22 labels. And response email set for each of this email classes are generated through GPT3.5.

Preprocessing techniques are applied to this dataset before used for training the models. During preprocessing mainly stop word removal and lemmatization is done. Stop word removal and lemmatization are essential preprocessing steps in text analysis and natural language processing tasks, as they help improve model accuracy by reducing the noise in the data and highlighting the most relevant features. Stop word removal eliminates common words with little meaning, allowing the model to focus on meaningful words that contribute to understanding the context and content of the text. This process reduces noise and computational complexity, leading to faster training times and improved model generalization. Stop words list that was used in this research is the default stop word list is from spacy library. After Stop word removal lemmatization is done.

Lemmatization, on the other hand, converts words to their base forms, reducing data sparsity and enhancing feature relevance. This process ensures that different inflections of the same word are treated as a single entity, allowing the model to better understand the semantic meaning of the text. Furthermore, lemmatization improves consistency in text data processing and analysis, leading to better performance in tasks such as classification or sentiment analysis. By applying these techniques, models can better understand the underlying patterns and relationships in the data, resulting in improved performance and generalization. In this research, we use Natural Language Toolkit (NLTK) to do the lemmatization of text.

Then after performing the processing of the dataset feature extraction is done. Here we have used two main feature extraction methods one is TIDF vectorizer and Huggingface BERT

model to extract feature vector from the text. The TF-IDF (Term Frequency-Inverse Document Frequency) vectorizer is a widely used technique in text analysis that extracts features from text data by quantifying the importance of each word in a document relative to a corpus. It combines term frequency and inverse document frequency, emphasizing meaningful and contextually important words while downplaying common words. These numerical representations, or feature vectors, serve as input for various machine learning models, enabling tasks such as text classification, clustering, and sentiment analysis. Overall, TF-IDF helps create meaningful and contextually relevant feature representations from text data.

Then we also use Hugging Face's DistilBERT to extract features from the text. Hugging Face's DistilBERT is a lighter version of BERT, designed to be faster and more memory-efficient while still retaining a significant portion of BERT's performance. DistilBERT converts text into contextualized vector representations using the Transformer architecture. To generate these vector embeddings, the input text is tokenized and passed through the DistilBERT model, which returns the embedding for each token. These TF-IDF vector will be used as the input to identifying the cluster using K-Mean clustering then TF-IDF vector and HuggingFace DistilBERT model extracted vector combine to a single vector and used as the input for the Intent classification model.

4.4 Process

In the process section of your research, you will discuss the main steps involved in designing the InERA system. This section can be divided into three main sub-stages: email clustering, intent identification, and reply generation.

Email Clustering

First Unsupervised email clustering is done on the current user email dataset. K-means Clustering algorithm is used for this purpose. K-means clustering is a machine learning technique that is used for unsupervised data analysis. This algorithm classifies data points into specific clusters based on their similarity and without any prior information about the groupings. In this

research, k-means clustering is important for identifying patterns and common themes in email data, allowing the InERA system to categorize emails into distinct groups. By understanding these groupings, the system can more effectively generate accurate and contextually appropriate email responses. Inputs for this Email clustering are the vectorized email content using the TF-IDF vectorizer in the feature extraction phase. Number of clusters will be decided based on the distribution of the email content. To determine the ideal number of clusters (k) for k-means clustering, the elbow method analysis is frequently utilized. This entails plotting the sum of squared errors (SSE) for various k values and identifying the point where the SSE starts to decrease at a slower pace, known as the "elbow point." If the resulting clusters are not well-separated with comparable email categories, the user can examine them and adjust the number of clusters as necessary. For this analysis word cloud is provided with the topic analysis on each cluster so the user can visualize the distribution of the content in each email group. Out of the formed clusters user can select the email clusters that can be answered using the InERA system automatically.

Then using the identified email clusters, we can create list of common responses for each email cluster by summarizing the content of reply provided for each email in the same cluster. For this we can used Large Language model like GPT 3.5 to summarize the content in email replies in each cluster. This intent and response map will be stored as JSON file that can be used in the email response generation module. Handling imbalanced clusters is a challenge with having multiple categories of emails incorrectly clustered together. User has control over enabling clusters that correctly mapped and disabling outliers and imbalanced clusters.

Intent Classification

Intent Classification is the next major component in this research. We use Multi-Layer Perceptron classifier for Intent Classification. Multi-Layer Perceptron (MLP) model is a supervised machine learning technique that is utilized for tasks involving binary or multiclass classification. The MLP classifier is essential in identifying the intention of incoming emails by examining the characteristics extracted from the email content. By accurately identifying the context of the email, the InERA system can generate contextually appropriate responses. The MLP classifier is essential for bridging the gap between feature extraction and response generation. To

train this Classifier we used the clustered email dataset from the Email clustering module. From this clustered email dataset features are extracted using TF-IDF vectorizer and HuggingFace DistilBERT model and obtain feature union of the two vectors. This extracted features are used to train the Multi-Layer perceptron model. After training the classifier it will be used to determine the intent of the incoming emails based on the same feature extracted from the email.

Reply Generation

The other key component in the InERA system is the reply generation module. For reply generation, we use Long Short-Term Memory (LSTM) model to generate full email reply based on corresponding response set for the intent identified from the Multi-Layer Perceptron model. Long Short-Term Memory (LSTM) is a recurrent neural network (RNN) technique that is specifically developed to manage long-term dependencies in sequential data. LSTM plays a significant role in generating contextually appropriate email responses. By capturing the context and structure of the input text, LSTM can generate more coherent and relevant email replies compared to traditional RNNs. Its ability to remember and process long sequences makes it ideal for natural language processing tasks like email reply generation, ensuring the InERA system provides high-quality, contextually accurate responses. LSTM model will be trained with the past email dataset from the user inbox. Scalability of this response generation module should also be taken in to consideration since LSTM computations can be expensive [19].

System Integration and Workflow

First the Email clustering module will be used to classify the email set into clusters and generate email response list based on email clusters. Then this clustered email data set will be then used as the input to the training of intent classification model. After training the classification model this classification model can classify incoming email after extracting features of the email with the TF-IDF vectorizer and Huggingface DistilBERT model. After Identifying the intent of the email then InERA extract the corresponding response set for the identified intents and pass this data to the reply generation module. Then this module will use the intent and response data to generate the full email reply based on the context of the response.

We have identified are two main limitations, They are generating full replies for emails with multiple general inquiries and maintain semantic meaning of the reply text generated when generating long text. We also identified there are two types of emails can be assisted mainly by InERA which are commonly seen on large mail list. First one is emails that are missing required information and general inquiries that can be answered using information available on the previous responses easily.

Major challenges are identifying the correct categories of emails based on the clusters extracted that can be answered using automated reply system. Since there can be outliers and Imbalance in clusters that may include emails with different context clustered together. Therefor user involvement is required to identify the correct clusters set that can be automated. One of the other challenging task is to generate the response set for each cluster based on the summary of the previous email responses for email set in that cluster. Handling ambiguous email content is also a challenge.

One of the main consideration that should be taken is the managing computational resources since this system operate on real-time email responding system should be light weight and should use optimum computational resources to deliver better user experience to the email user.

4.5 Output

We can breakdown the output if this research in to three main sections the generated email replies from InERA system, their evaluation, and the method of delivering the replies to the user via an Email client server.

Generated Email Replies

Reply generation module will out put the body text of the email based on the response set context provided to the reply generation module. Greeting text and the signatures are added to the final out email reply based on the configurable email template that user can edit and customize as they prefer. Greeting and addressing will be a general greeting since system does not identify gender or the person as an entity. There is limitation on length of the output email even though there are many intents identified generated response will only have maximum of 500 words. Since generating too long text based on the context might lead to generating response without a clear overall meaning and lack of correlation within individual sentences.

Email Delivery via SMTP Server

SMTP server is integrated with the automatic email reply assistance system to send generated email replies to users. The SMTP server plays a crucial role in the email delivery process, ensuring timely and accurate delivery of the replies. Configuring the connection between the system and the SMTP server involves specifying server address, port, username, and password for authentication. Challenges in using an SMTP server for email delivery may include handling errors, managing rate limits, and ensuring security and privacy of user data. Scalability considerations involve the system's ability to handle multiple users or large volumes of emails, which can be addressed through optimizing server configurations and resource management.

4.6 Features

In this section, we focus on the features of the Intelligent Email Reply Assistant (InERA) system that contribute to its efficiency and accuracy in generating automatic email replies. The features of InERA are designed to address various aspects of email management and to provide a comprehensive solution for users to manage their email inboxes effectively. These features include:

1. Intent Classification: To assist the system in comprehending the purpose of emails and producing appropriate responses, InERA classifies incoming emails according to their intent by using powerful machine learning algorithms and natural language processing techniques.

2. Context Extraction: InERA identifies key information from the email content and uses it to generate meaningful replies. The system extracts entities, topics, and context from the email body to create a context-aware response that addresses the user's query or concern.

3. Personalization: InERA leverages user-specific data, such as their previous email responses, to tailor its generated replies to the user's style and preferences. This personalization ensures that the replies sound authentic and consistent with the user's communication patterns.

4. Reply Generation: InERA employs deep learning models, such as LSTM, to generate comprehensive and grammatically correct replies based on the email's intent, context, and user preferences. The generated replies aim to address the user's query effectively, reducing the need for manual intervention.

5. Integration with Email Clients: InERA is designed to integrate seamlessly with popular email clients, enabling users to access its features directly from their email interface, streamlining the process of managing their inbox and improving productivity.

6. Scalability and Performance: InERA's architecture is designed for scalability, ensuring that the system can handle increasing volumes of emails and multiple users without compromising performance or response times.

By incorporating these features, InERA aims to provide a robust and efficient solution for email management, reducing email overload and enhancing user productivity. In conclusion, the features

of the InERA system work together harmoniously to offer a comprehensive automatic email reply assistance solution. The main components, including intent classification, context extraction, personalization, reply generation, integration with email clients, and scalability, interact seamlessly to provide an efficient and user-friendly experience. The synergy between these components ensures that the generated replies are accurate, contextually appropriate, and tailored to the user's communication style, ultimately addressing the challenges posed by email overload.

The significance of these features lies in their ability to achieve the overarching goals of this research, which include enhancing productivity and reducing the burden of email management for users. By combining advanced natural language processing techniques, machine learning algorithms, and deep learning models, InERA can effectively respond to various email intents and contexts, ensuring that users can focus on more critical tasks. The significance of these features cannot be overstated, as they are integral to the effectiveness of the InERA system and highlight the possibility of further advancements and enhancements in the domain of email automation and management.

4.7 Users

The targeted user base for this research primarily consists of professionals in the corporate sector who frequently deal with a high volume of emails, leading to email overload. These users may include managers, executives, customer support representatives, sales teams, and other employees who rely heavily on email communication for their daily tasks. The intelligent email reply assistance system aims to alleviate the burden of email overload, enabling these users to manage their email communication more efficiently and improve overall productivity.

Developers also form a part of the targeted user base, as they can integrate the email reply assistance solution into various software applications, tools, or platforms. By incorporating this solution, developers can provide an enhanced user experience and streamline communication processes for their corporate clients or end-users.

4.8 Summary

In conclusion, this chapter has presented the approach to the Intelligent Email Reply Assistant (InERA) system, covering all critical aspects that contribute to the development and functioning of the solution. We began with an introduction, laying the groundwork for the chapter's content, followed by the hypothesis that underpins the InERA system. We then discussed the input, process, and output components, providing a clear understanding of how the system operates and the expected outcomes. we delved into the features that make the InERA system effective and efficient, such as intent classification, context extraction, personalization, and reply generation, as well as its integration with email clients and scalability. We also considered the targeted user base, emphasizing the importance of addressing the needs of both corporate email users and developers seeking to implement similar solutions. Finally, we provided a summary of the chapter's content, highlighting the key concepts and the contributions made by the InERA system in tackling email overload.

The insights gained from this chapter will serve as a foundation for the subsequent chapters, starting with the design phase. In the next chapter, we will focus on the design of the InERA system, addressing each component in detail and considering their interactions, to ensure the seamless functioning of the system. By building upon the approach presented here, we aim to further advance email automation and assistance, addressing the challenges faced by corporate email users and developers seeking to implement similar solutions.

CHAPTER 5

DESIGN

5.1 Introduction

In this chapter, we have presented the prototype design for InERA system through which the design, framework was elaborated. Here we present a detailed architecture for the InERA system, outlining the various components and techniques involved in its development. The chapter is organized as follows: Section 2 describes the system's high-level architecture and the interactions between its components; Sections 3 and 4 discuss the data collection and preprocessing steps, as well as the choice of deep learning model; Section 5 focuses on feature engineering and the techniques employed to extract relevant information from the input data; Section 6 details the training and optimization process, while Section 7 addresses evaluation metrics and validation; Section 8 presents the user interface design, and Section 9 delves into the integration of the system with popular email clients. Finally, Section 10 concludes the chapter by summarizing the design choices and providing insights into potential future improvements.

5.2 High-level system overview

High level architecture diagram presented in figure 5.1 show the high-level architecture diagram of the Intelligent Email Reply Assistant System. This high level architecture shows the data between the major components of the InERA system.

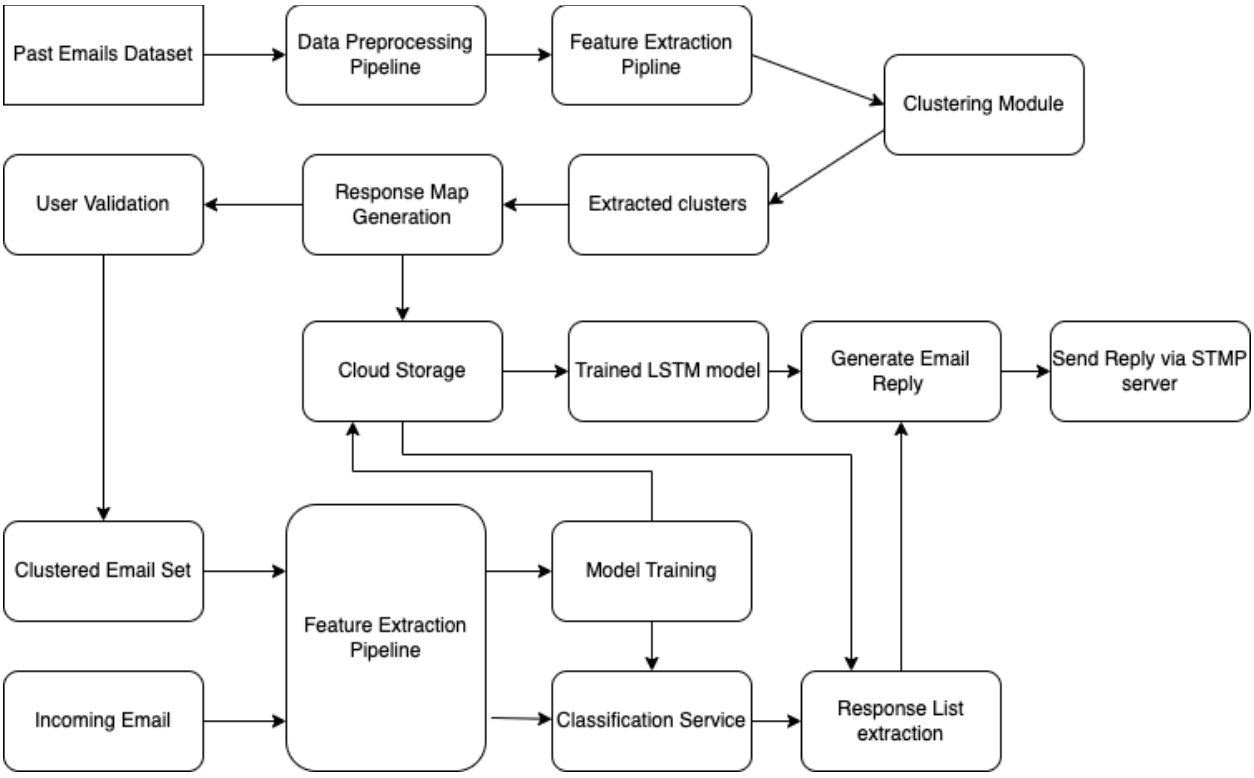


Figure 5. 1: High level architecture diagram

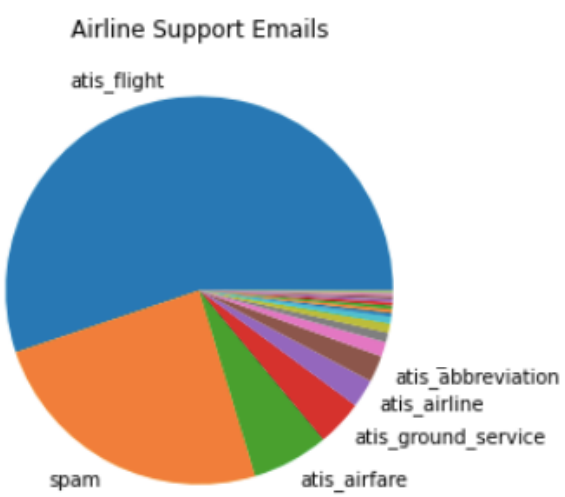
As the initial step of the INERA system we first get the already available dataset in the email client using IMAP protocol in to the INERA system. The extracted email dataset is then passed onto the Preprocessing module. In the preprocessing module Stop word removal and lemmatization and tokenization of email dataset will be performed. After preprocessing the data email dataset is split into training and Testing datasets. Then Data is passed on to the clustering module that perform the clustering dataset in to separate clusters based on the meaning using K-Mean clustering. Data is vectorized using TIDF vectorizer before doing the K-Mean clustering on the dataset. After clustering Clustered email data and corresponding responses for each email in the cluster is summarized and generate a sample response for each cluster of emails. This way email and response map is generated and saved in the cloud storage in JSON format. Then these email cluster data set is then used to train the Intent classifier as training data. Then this trained model is loaded to the Intent classifier module.

Then once a new Email received through the email client through IMAP protocol email body content will be parsed and passed on to the preprocessing module and preprocess the email content.

During the preprocessing email will be split into multiple sentences based on the semantic. Then these multiple sentences will be then passed on to the Intent Identification module. Once the intents of each email sentences are identified, intent predictions with a higher confidence value than the threshold value it will passed to the reply generation module. In the reply generation module, it will first get the predicted intent list and map it to the response set for the each identified response cluster and generate the reply text based on the context of the response set for the intent list dynamically. After the response is generated it will be sent to the STMP server to send the email to the user.

5.3 Description of the dataset used for training and validation

The dataset employed for this research is the ATIS dataset sourced from Kaggle. ATIS, or Airline Travel Information System, is a well-known benchmark dataset frequently utilized in intent classification tasks. As a publicly available labeled dataset, it comprises 19,900 emails directed to the Airline Travel Information System. These emails predominantly consist of customer inquiries received by airline support staff and also include a portion of spam emails. Figure 5.1 demonstrate the distribution of the dataset using a pie chart. part of the dataset will be used for testing therefore emails will be separated from the dataset for testing. The proportions used for each split 70% training, 15% validation, 15% testing. The dataset is provided in CSV format and features 22 distinct labels. To generate response email sets for each email class, the GPT-3.5 model is used.



Label	Count
atis_flight	10998
spam	4887
atis_airfare	1269
atis_ground_service	765
atis_airline	471
atis_abbreviation	441
atis_aircraft	243
atis_flight_time	162
atis_quantity	153
normal	113
atis_flight#atis_airfare	63
atis_distance	60
atis_airport	60
atis_city	57
atis_ground_fare	54
atis_capacity	48
atis_flight_no	36
atis_meal	18
atis_restriction	18
atis_airline#atis_flight_no	6
ground_service#atis_ground_fare	3
atis_airfare#atis_flight_time	3
atis_cheapest	3
raft#atis_flight#atis_flight_no	3

Figure 5. 2 : Dataset distribution

Then Data preprocessing is done on email content before dataset is used in the classifier and clustering. Main data processing techniques applied are tokenization, stop words removal and lemmatization. In NLP, tokenization involves the segmentation of a given text into smaller units, commonly referred to as tokens. These tokens may comprise words, phrases, sentences, or even individual characters, depending on the specific application or analysis required. This process helps in analyzing and understanding the text, as well as preparing it for further processing, such as parsing, part-of-speech tagging, or feature extraction.

Tokenization typically involves the following steps:

- Identifying word boundaries: This step involves detecting where one word ends and the next begins. In English language, tokenization is frequently achieved by utilizing white spaces and punctuation marks as delimiters. However, in languages where word boundaries are not clearly defined, such as Chinese or Japanese, tokenization can be a more challenging task.

- Handling special characters and contractions: Tokenization may also involve dealing with special characters (e.g., hashtags, email addresses, URLs) and contractions (e.g.,

"I'm" should be split into "I" and "am"). These cases often require custom rules or algorithms to ensure that the text is tokenized correctly.

- Normalizing tokens: Once the text is split into tokens, it may be necessary to normalize them, such as converting all tokens to lowercase, removing punctuation marks, or stemming/lemmatizing the words.

Tokenization is a crucial step in preparing textual data for numerous Natural Language Processing (NLP) applications, including sentiment analysis, text categorization, machine translation, and information extraction. Numerous NLP libraries, such as NLTK, spaCy, and the Hugging Face Transformers, provide built-in tokenization functions to handle various languages and use cases.

preprocessing, to assess its effectiveness, the dataset was partitioned into a training set and a testing set. The InERA models were trained utilizing the training set, whereas the testing set was employed to evaluate the performance of the system. To ensure that the model was trained on a balanced dataset, the following steps were performed:

- Upsampling: In order to balance the number of examples for each class, the training data was upsampled. This step included resampling each class to match the size of the biggest class, ensuring that each category in the training data was represented equally.

- Resampling: Using resampling upsample the minority classes in the training data. The resampled classes were then combined to create a new balanced dataset for training.

By implementing these data collection and preprocessing techniques, InERA was trained on a diverse and representative sample of airline customer inquiries. This approach enabled the model to generate accurate and relevant responses to a wide range of email queries.

5.4 Choice of Machine Learning Models

First of all, we use K-mean clustering for cluster the existing email dataset.Unsupervised machine learning techniques like K-means clustering aim to partition a set of data points into K distinct groups based on their similarities. This algorithm repeatedly allocates each data point to the closest cluster center and subsequently adjusts the centroids according to the data points associated with each cluster. The procedure carries on until it reaches stability, ultimately generating a series of K clusters in which every data point is connected to the nearest centroid cluster.

K-means clustering is a suitable approach for clustering emails and identifying email classes for several reasons:

- Scalability: K-means is a computationally efficient algorithm that scales well with the number of data points, making it well-suited for large email datasets. This scalability allows the algorithm to handle a massive volume of emails, which is crucial when building a dataset for an intent classification model.

- Simplicity: The K-means algorithm is an appealing alternative for clustering tasks since it is simple to understand and use. This approach is easy to modify and adapt as needed, and it can be quickly combined with other machine learning models inside of a processing pipeline.

- Feature Space: Emails can be represented using various feature extraction methods, such as TF-IDF or word embeddings, which can capture the underlying structure and semantics of the text. K-means clustering works effectively in these high-dimensional feature spaces, allowing it to identify meaningful patterns and relationships between emails.

- Unsupervised Learning: K-means is an unsupervised learning technique, therefore it can detect clusters without labeled data. This is particularly useful when dealing with

emails, as manual labeling can be time-consuming and labor-intensive. The algorithm can automatically discover underlying email classes without any prior knowledge of their structure.

- Flexibility: K-means can be easily adapted to accommodate different similarity measures or distance metrics, allowing it to work effectively with various types of text data and feature representations. This flexibility makes the algorithm more robust and versatile in clustering emails and identifying email classes.

- Building Dataset for Intent Classification: By clustering emails into distinct groups, K-means can help build a structured dataset for intent classification models. Each cluster represents a specific intent or category, and the email-reply pairs within each cluster can be used as training data for the classification model. This approach helps create a balanced dataset that captures the diversity of intents and enables the intent classification model to learn meaningful patterns.

In the context of an intelligent email reply assistance system, an MLP classifier is suitable for several reasons:

- High-dimensional input: The combination of TF-IDF and DistilBERT vectorizers provides high-dimensional feature vectors that capture both the semantic and syntactic information in the text. MLPs can handle high-dimensional input effectively, learning complex patterns and relationships in the data.

- Non-linear decision boundaries: Emails can contain complex language structures and various writing styles, which may result in non-linear relationships between the features and the intents. The non-linear activation functions used in MLPs enable them to model

and learn such non-linear decision boundaries, improving the classifier's ability to distinguish different intents accurately.

- Adaptability to diverse data: MLPs can generalize well to unseen data, making them suitable for classifying intents in a diverse range of incoming emails. The universal approximation capability of MLPs allows them to model complex relationships in data, making them applicable to a wide range of applications, including email intent classification.

- Hierarchical feature learning: The multiple hidden layers in an MLP allow it to learn hierarchical representations of the input data. Each successive hidden layer can learn more abstract features, enabling the model to capture deep patterns and structures in the emails, which is crucial for accurate intent classification.

- Scalability and efficiency: MLPs are scalable and can handle large datasets efficiently, making them suitable for real-world applications like email reply assistance systems, where the volume of incoming emails can be substantial. The use of techniques like stochastic gradient descent and mini-batch learning can further enhance the training efficiency of MLPs.

- Customizability: The architecture of an MLP classifier can be easily customized by adjusting the number of hidden layers and neurons, allowing researchers to experiment with different configurations to find the optimal model for the task. This flexibility enables researchers to tailor the MLP to the specific requirements of their email reply assistance system.

Overall, an MLP classifier is well-suited for an intelligent email reply assistance system because it can effectively handle the high-dimensional input generated by the TF-IDF and DistilBERT vectorizers, discover intricate links and patterns in the data, and generalize effectively

to emails that have not been seen. Its scalability, customizability, and ability to model non-linear decision boundaries make it a suitable choice for classifying intents in a diverse range of incoming emails.

Long Short-Term Memory (LSTM) represents a recurrent neural network (RNN) structure explicitly created to manage sequential data and acquire an understanding of extended relationships within the information. Unlike traditional RNNs, which struggle to learn patterns over long sequences due to the vanishing gradient problem, LSTMs incorporate a unique memory cell and gating mechanisms that allow them to capture and retain information over extended periods. LSTMs introduce memory cells and gating mechanisms to maintain and control the flow of information over time.

LSTMs are suitable for generating reply email content based on the response context for several reasons such as:

1. Sequential Nature: Emails are inherently sequential data, consisting of words and sentences that follow a specific order. LSTMs are designed to handle such sequence data and can learn the structure and patterns within the email text, making them a natural fit for this task.

2. Context Preservation: LSTMs can maintain information from previous time steps, which allows them to preserve context across the email content. This ability to remember past information is crucial for generating meaningful and coherent email replies that accurately reflect the context of the original email.

3. Handling Variable Lengths: LSTMs can handle input and output sequences of varying lengths, which is essential for email reply generation, as the length of the generated responses may differ depending on the email content and the specific context of the conversation.

4. Long-range Dependency Learning: One of the key strengths of LSTMs is their ability to learn long-range dependencies within the data. In the context of email reply generation,

this means that LSTMs can effectively capture the relationships between different parts of the email content, even if they are separated by a significant distance. This capability allows the model to generate responses that are coherent and contextually appropriate.

5. End-to-End Learning: LSTMs can be trained in an end-to-end manner, which means that the entire process of generating email replies can be learned directly from the data without the need for explicit feature engineering or manual rule creation. This approach simplifies the model development process and often results in better performance compared to traditional rule-based systems.

Hence LSTMs are well-suited for generating reply email content based on the response context due to their ability to handle sequential data, preserve context, manage variable-length sequences, learn long-range dependencies, and enable end-to-end learning. These characteristics make LSTMs an ideal choice for building an effective and contextually-aware email reply generation system.

Comparison with alternative models

The GPT (Generative Pre-trained Transformer) model is another powerful option for email reply generation based on context instead of using an LSTM model. GPT is a state-of-the-art language model based on the Transformer architecture, which is designed to generate human-like text by predicting the next word in a given sequence. There are several reasons why GPT can be a superior choice for email reply generation:

1. Pre-trained Knowledge: GPT is pre-trained on massive text corpora, allowing it to capture rich semantic and syntactic information from the data. This pre-trained knowledge enables GPT to generate high-quality email replies by leveraging its understanding of language structure, grammar, and context.

2. Context-awareness: GPT is designed to be highly context-aware, as it uses self-attention mechanisms to capture relationships between words in a given sequence. This context-awareness allows GPT to generate coherent and contextually relevant email replies, taking into account the entire input email.

3. Scalability: The Transformer architecture, which GPT is based on, is inherently more scalable than LSTMs, as it allows for parallel computation across input sequences. This scalability enables GPT to handle longer input sequences and generate more complex email replies efficiently.

4. State-of-the-art Performance: Across a variety of NLP tasks, including text creation, GPT has shown state-of-the-art performance. By using GPT for email reply generation, you can leverage its advanced capabilities to generate high-quality, human-like responses.

5. Fine-tuning: GPT models can be fine-tuned on a specific dataset to adapt to the nuances and domain-specific language used in that context. Fine-tuning GPT on your email dataset ensures that the generated replies are tailored to the specific needs of your application.

6. Transfer Learning: GPT's pre-trained knowledge can be effectively transferred to various NLP tasks, including email reply generation. This transfer learning capability allows GPT to achieve high performance even with a smaller dataset, as it has already learned general language representations during its pre-training phase.

Using a GPT model for email reply generation based on context can offer several benefits over an LSTM model, such as leveraging pre-trained knowledge, improved context-awareness, better scalability, state-of-the-art performance, fine-tuning capabilities, and transfer learning. These advantages make GPT a powerful alternative for generating contextually appropriate and high-quality email replies.

Feature Engineering

Feature engineering plays a crucial role in preparing the text data for clustering and classification. Feature engineering involves the transformation and extraction of meaningful features from raw data to enhance the performance of machine learning models. In this context, the feature engineering process consists of several steps:

1. Text Preprocessing: Initially, the email content must be cleaned and preprocessed. This encompasses converting the text to lowercase, tokenizing it (breaking the text into individual words or tokens), removing stop words (discarding frequently occurring words that lack substantial meaning, like "and", "the", etc.), and performing lemmatization (simplifying words to their root or canonical form, for example, changing "running" to "run"). These preprocessing steps help simplify the text data and remove noise, making it easier for the models to identify patterns and relationships.

2. Feature Extraction using DistilBERT: Next, the Hugging Face DistilBERT model is used to extract contextualized word or sentence embeddings from the preprocessed text. DistilBERT, a smaller and faster version of the BERT model, is pre-trained on a large corpus of text data, allowing it to capture rich syntactic and semantic information. By fine-tuning DistilBERT on your email dataset, you can generate meaningful embeddings that represent the structure and semantics of the text, which are useful for clustering and classification tasks.

3. TF-IDF Vectorization: In parallel, you also employ the Term Frequency-Inverse Document Frequency (TF-IDF) method to create a high-dimensional vector representation of the text. This approach assigns weights to words based on their frequency in a document and their rarity across the entire document collection. TF-IDF vectors help capture the importance of words in the email content and emphasize the discriminative features that can help differentiate email classes.

4. Feature Union: After obtaining the DistilBERT embeddings and the TF-IDF vectors, you create a feature union by combining the two representations. The combined feature set leverages both the contextualized embeddings from DistilBERT and the term-weighted vectors from TF-IDF. This fusion helps to capture different aspects of the text data, improving the overall representation and enriching the input for the subsequent clustering and classification models.

5. Clustering and Classification: Finally, the engineered features are used as input to the K-means clustering algorithm to group similar emails and build the dataset for the intent classification model. The feature union is also fed into the Multi-Layer perceptron classifier to predict the email classes, enabling the system to generate appropriate email replies.

In summary, feature engineering in this research involves text preprocessing, feature extraction using DistilBERT, TF-IDF vectorization, feature union, and application of the engineered features to clustering and classification tasks. In order to improve the performance of the machine learning models used for email clustering and classification, this method makes sure that the text input is appropriately represented and processed.

5.5 Description of the features extracted from the input data

Each feature plays a crucial role in the model's performance, capturing different aspects of the email text and contributing to the overall effectiveness of the clustering and classification tasks. Here's an explanation of the importance of each feature:

1. Text Preprocessing: The preprocessing steps, including tokenization, stopword removal, and lemmatization, help simplify the text and remove noise. This preprocessing allows the feature extraction methods and machine learning models to focus on the most relevant and meaningful parts of the text, improving their ability to identify patterns and relationships between emails.

2. DistilBERT Embeddings: DistilBERT embeddings capture contextualized information about the text, including the syntax, semantics, and relationships between words. These embeddings are pre-trained on a massive text corpus, providing a rich understanding of the language structure and enabling the model to generate more accurate and coherent email replies. By fine-tuning DistilBERT on the email dataset, the embeddings become tailored to the specific language and context used in the emails, improving the model's performance in clustering and classification tasks.

3. TF-IDF Vectors: TF-IDF vectors emphasize the importance of words in the email content by assigning weights based on their frequency in a document and their rarity across the entire document collection. These vectors help capture the discriminative features that can differentiate between email classes, allowing the clustering and classification models to better distinguish between various intents and contexts. The use of TF-IDF vectors also provides a complementary perspective to the contextualized embeddings from DistilBERT, enriching the overall representation of the email text.

4. Feature Union: Combining DistilBERT embeddings and TF-IDF vectors in a feature union helps capture different aspects of the text data, resulting in a more comprehensive representation of the email content. The contextualized information from DistilBERT captures relationships between words and the overall semantics, while the term-weighted vectors from TF-IDF highlight the importance of specific words in the text. This combination allows the models to leverage both types of information for improved clustering and classification performance.

Here DistilBERT model is the main model used for extracting features from text. Following are some of the reasons why DistilBERT model is selected here for feature extraction.

- DistilBERT, a smaller and faster variant of the original BERT model, is an excellent choice for text vectorization in intent classification due to its reduced size, pre-trained knowledge, and effective feature extraction. Being approximately 40% smaller and 60% faster than

BERT, DistilBERT is suitable for resource-constrained environments, such as mobile devices or embedded systems.

- Pre-trained on a large text corpus, DistilBERT captures rich syntactic and semantic information. Fine-tuning the model on a specific dataset leverages its pre-trained knowledge to generate meaningful embeddings for intent classification, leading to improved performance. Context-aware embeddings represent complex linguistic patterns, crucial for understanding user inputs in intent classification tasks.

- DistilBERT is part of the widely-adopted Hugging Face Transformers library, which offers extensive documentation and a large user community. The library's interoperability with popular deep learning frameworks like TensorFlow and PyTorch simplifies integration into existing machine learning pipelines. Additionally, DistilBERT's transfer learning capability enables high classification performance with smaller datasets, as it has already learned general language representations during its pre-training phase.

In conclusion, DistilBERT's benefits, such as reduced model size and complexity, pre-trained knowledge, effective feature extraction, wide adoption, and transfer learning capabilities, make it a compelling choice for building efficient and high-performing intent classification systems. The combination of these features ensures that the models have access to a comprehensive and meaningful representation of the email content, leading to improved performance in generating contextually appropriate email replies.

5.6 Training and Optimization

For InERA system We mainly use 3 Machine Learning models. In this section, will focus on the implementation of the various models used in the InERA system and the optimization techniques used to improve their performance.

In the case of the Multi-Layer Perceptron model for intent classification, the training process involves feeding the labeled data into the model and iteratively adjusting the model's parameters to minimize the cost function. The optimization technique used in this case is, which updates the parameters based on a small subset of the data at each iteration. The section will also cover hyperparameter tuning techniques, such as grid search or random search, used to find the optimal set of hyperparameters for the model.

For the K-means clustering algorithm used in grouping emails, the section will describe the process of initializing cluster centers and iteratively reassigning data points to clusters based on their distance to the center. The optimization techniques used in this case could be the elbow method, which involves evaluating the clustering performance for different numbers of clusters, or silhouette analysis, which evaluates the similarity of data points within clusters compared to other clusters.

In the instance of the LSTM model for email reply generation, the section will address the training procedure, which entails feeding the model with word sequences and adjusting the model's parameters to minimize the loss function. The optimization technique used in this case could be gradient descent or its variations, such as Adam, which adaptively adjusts the learning rate at each iteration. The section will also describe hyperparameter tuning techniques used to find the optimal set of hyperparameters for the model.

Overall, the training and optimization section of the design chapter will provide a detailed description of the processes used to train and optimize the various models used in the InERA system, highlighting the techniques and strategies used to improve their performance.

5.7 Loss function and optimization algorithm selection

Regularization techniques are used to improve the generalization performance and avoid overfitting of the machine learning models. Following are the regularization techniques applied to the models in this research:

- Regularization technique: To prevent overfitting in the MLP classifier, L1 or L2 regularization can be applied. L1 regularization adds the weights' absolute values, whereas L2 regularization adds their squared values to the loss function. Regularization increases the loss function's penalty term, encouraging the model to utilize lower weights and minimizing model complexity and overfitting [20].

- Adam optimization: In this research, the Adam optimization algorithm is used to minimize the categorical cross-entropy loss. It computes adaptive learning rates for each weight and bias by using the first and second moments of the gradients. Adam is a popular choice for training deep learning models, including MLP classifiers, due to its efficiency and ability to converge quickly.

- Dropout: Dropout is a regularization technique for neural networks, such as LSTM and DistilBERT. During training, dropout randomly sets a fraction of the input units to zero at each update, preventing the model from relying too heavily on individual units and promoting generalization [20]. You can apply dropout to the LSTM layers or fine-tuned DistilBERT layers in your models.

When applying these regularization techniques, it is essential to consider the specific models being used and the characteristics of the problem. Regularization should be applied carefully and tuned using techniques such as cross-validation to achieve the best balance between model complexity and generalization performance.

The Adam optimizer is used in training the Multi-Layer Perceptron model. The Adam optimizer, or Adaptive Moment Estimation, is an adaptive optimization algorithm that combines the benefits of two popular gradient descent optimization techniques, (RMSProp) Root Mean Square Propagation and Adaptive Gradient Algorithm (AdaGrad). Adam maintains an exponentially declining average of previous gradients and squared gradients to calculate the adaptive learning rates for each parameter. This approach enables Adam to adjust the learning rate for each parameter individually, making it suitable for the Multi-Layer Perceptron classifier.

There are few reasons why Adam is a suitable choice for this model. First, its adaptive learning rates facilitate faster convergence and better performance by effectively handling noisy or sparse gradients. Second, Adam requires relatively low memory, as it maintains only the first and second moments of gradients, making it computationally efficient. Moreover, it is robust to different types of data and hyperparameters, reducing the need for extensive hyperparameter tuning. Lastly, Adam is effective in handling non-stationary objectives, which can occur in natural language processing tasks such as intent classification and text generation. By using the Adam optimizer, the models in this research can achieve faster convergence and improved performance, enhancing their ability to accurately classify email intents and generate contextually appropriate email replies.

Dropout is a regularization technique used to overcome overfitting in deep learning models, including LSTMs. Dropout works by randomly "dropping out" a fraction of the neurons during training, effectively setting their output to zero. This process prevents the model from relying too heavily on any single neuron or a specific set of neurons, which encourages the model to learn more robust representations of the data. The dropout rate, determines the fraction of neurons to be dropped during training.In the context of the InERA system, the LSTM model is used to generate email replies based on the given context. Dropout regularization is suitable for this application for several reasons:

- Prevent overfitting: Email replies can vary significantly, and overfitting to the training data may result in generated replies that are too specific or do not generalize well to new, unseen email contexts. Dropout helps the LSTM model learn more robust and generalizable features by reducing its reliance on specific neurons or input patterns.

- Robustness: By applying dropout to both input/output and recurrent connections, the LSTM model becomes more robust in handling the inherent variability in email content and style. This helps the model generate more appropriate and diverse replies for a range of email contexts.

- Improve generalization: Dropout helps the LSTM model to acquire general features that help it provide coherent and pertinent email responses even when the input context is different from the examples viewed during training.

Training/validation data split strategy

Choosing the right strategy for splitting your data into training and validation sets is crucial to ensure that your intelligent email reply assistant can learn effectively and generalize well to unseen data. There are many strategies for splitting the data such as Random Split, Stratified Split, Time-based Split, Stratified k-Fold Cross-Validation, k-Fold Cross-Validation and Leave-One-Out Cross-Validation. When choosing a data split strategy, consider factors such as the dataset size, class distribution, and temporal aspects. It is essential to evaluate the model's performance using a consistent and reliable methodology, ensuring that the InERA ML models can generalize well to unseen data. Out of these strategies Random Split is used to do the data split. In Random Split, Divide the dataset randomly into training and validation sets. Typically, you would allocate around 70% of the data for training, 15% for validation and 15% for testing. This method is simple and works well when the dataset is large and the distribution of classes is balanced.

When choosing a data split strategy, consider factors such as the dataset size, class distribution, and temporal aspects. It is essential to evaluate the model's performance using a consistent and reliable methodology, ensuring that the intelligent email reply assistant can generalize well to unseen data.

Hyperparameter tuning approach and results

Hyperparameter tuning is the process of optimizing the parameters of a machine learning model that are not learned during training but are set before the training process begins. These parameters, known as hyperparameters, can significantly impact the model's performance and generalization ability. Hyperparameter tuning is crucial because selecting appropriate hyperparameters can lead to better model performance, faster convergence, and improved generalization to unseen data. The process often involves searching through a range of possible hyperparameter values and evaluating

their impact on model performance using techniques such as Bayesian optimization, grid search or random search. Ultimately, hyperparameter tuning helps in finding the optimal configuration for a model, ensuring its effectiveness in solving the given problem. The three main components of this research are K-Mean Clustering model and Multi-Layer Perceptron Model and LSTM model. Following are the Hyperparameter tuning approaches we have applied for these 3 models.

The Elbow method is a suitable and simple hyperparameter optimization technique for determining the optimal number of clusters (K) in K-means clustering. The Elbow approach includes detecting the point when the SSE starts to fall at a slower pace, like a "elbow" on the graph, and graphing the within-cluster sum of squared errors (SSE) versus the number of clusters. This point indicates that increasing the number of clusters past that point does not significantly reduce the within-cluster variation. By using the Elbow method, you can effectively determine the optimal number of clusters for K-means clustering in your research, ensuring a more accurate email intent classification and improved model performance. For the Elbow method optimization, the within-cluster sum of squared errors (SSE) is computed as follows:

$$SSE = \Sigma(\Sigma(||x\_i - c\_j||^2))$$

where the first summation is over all the K clusters, and the second summation is over all the data points (x_i) assigned to cluster c_j.

For the Multi-Layer Perceptron Classifier, we use Randomized Grid search to Identify the best hyperparameter values for the model. Randomized grid search is a suitable technique for hyperparameter tuning in this research because it can efficiently explore a large search space and often provides comparable performance to exhaustive grid search with fewer iterations. It does so by randomly sampling a predefined number of hyperparameter combinations from the specified range of possible values, instead of evaluating all possible combinations.

In randomized grid search, you define the range of possible values for each hyperparameter and the number of iterations to perform. The approach randomly selects a set of hyperparameter values for each iteration and uses cross-validation to assess the model's performance. The ideal configuration is chosen based on the set of hyperparameters that produces the best results. To

explain randomized grid search, let's consider the Multi-Layer Perceptron model. some key hyperparameters that can be tuned using randomized grid search are:

- Number of hidden layers: This determines the depth of the neural network. More hidden layers can help the model learn more complex and abstract features, but it may also increase the risk of overfitting and require more training time.

- Number of neurons in each hidden layer: This controls the width of the hidden layers and influences the model's capacity to learn complex patterns. Increasing the number of neurons can improve the model's performance, but it may also increase computational complexity and the risk of overfitting.

- Activation function: This is the non-linear function applied to the weighted sum of the inputs at each neuron. Common activation functions include ReLU, sigmoid, and tanh. Different activation functions can have a significant impact on the model's performance and convergence speed.

- Learning rate: This is a crucial hyperparameter for the optimization algorithm (e.g., stochastic gradient descent or Adam). While a higher learning rate can speed up training but cause it to overshoot the ideal solution, a lesser learning rate might result in greater convergence and slower training.

Following is the set of value list that can provided for each of this hyper parameter.

```
hidden_layers : [1, 2, 3, 4],
neurons_per_layer : [50, 100, 200, 300],
activation_function : ['relu', 'sigmoid', 'tanh'],
learning_rate :  [ 0.1, 0.01, 0.001 ]
```

Once Randomized grid search training started it evaluates the Multi-Layer Perceptron model using cross-validation with the selected hyperparameters. The process is repeated for the specified number of iterations, and the best combination of hyperparameters is chosen based on the best cross-validated performance metric, such as accuracy or F1 score.

Finally, for the LSTM model also we can use random grid search for hyperparameter optimization. In order to apply randomized grid search for hyperparameter tuning in an LSTM model, you should take the following steps:

- Define the hyperparameter space: For each hyperparameter in the LSTM model, such as the number of hidden layers, the number of hidden units inside each layer, the learning rate, and the dropout rate, provide the range or distribution.

- Determine the number of iterations: Decide on the number of random hyperparameter combinations to evaluate. More iterations will lead to a more thorough search of the hyperparameter space but will require more computational resources.

- Evaluation strategy: Choose an evaluation strategy, such as cross-validation or a train-validation split, to assess the performance of the LSTM model for each hyperparameter combination. Select an appropriate performance metric, such as accuracy, F1 score, or BLEU score, depending on the task at hand.

- Perform random grid search: For each iteration, randomly sample a combination of hyperparameters from the specified hyperparameter space. Train and evaluate the LSTM model with the selected combination using the chosen evaluation strategy. Record the performance metric for each combination.

- Select the best combination: After all iterations are complete, identify the hyperparameter combination that yielded the highest performance according to the chosen metric. Use this combination to train the final LSTM model on the entire dataset.

By using random grid search for hyperparameter tuning, you can efficiently explore a wide range of hyperparameter values, increasing the likelihood of finding a combination that leads to optimal performance for the LSTM model. It is computationally efficient, allowing for a more extensive exploration of the hyperparameter space within a shorter time frame compared to exhaustive grid search. Additionally, randomized grid search can often find a near-optimal combination of hyperparameters with fewer iterations. However, there are some limitations to this approach. Randomized grid search does not leverage prior knowledge or incorporate information from previous evaluations, which can lead to redundant or suboptimal sampling of the hyperparameter space. This can result in a less efficient search compared to techniques like Bayesian optimization, which use probabilistic models to guide the search process. Moreover, the quality of the results in randomized grid search depends on the number of iterations, and increasing the number of iterations may lead to higher computational costs.

5.8 Selection of evaluation metrics

In order to effectively analyze the success of the generated models and systems in the study, it is essential to use the right evaluation metrics. In this section, we describe the evaluation metrics, validation strategies, and benchmarking approaches used for the various components of our Intelligent Email Reply Assistant (InERA). For each component of our system, we selected evaluation metrics that best capture the performance in its respective task:

- K-means clustering: We used silhouette score as a metric to assess the quality of the email groups formed by the clustering algorithm, considering the cohesion and separation between clusters.

- Multi-Layer Perceptron classifier: We evaluated the classifier's performance using accuracy and F1 score, which provide insights into the model's ability to correctly classify email intents.

- LSTM model for text generation: We chose the ROUGE metric to evaluate the quality of the generated email replies, considering various aspects of similarity between the generated text and reference text.

Each of the selected evaluation metrics is tailored to the specific task it assesses, providing a meaningful measure of performance for that particular component of the Intelligent Email Reply Assistant (InERA) system.

- Silhouette score (K-means clustering): The silhouette score is chosen for evaluating the K-means clustering algorithm because it measures the quality of the formed email groups based on both cohesion (how close the data points within a cluster are to each other) and separation (how distinct the data points in different clusters are). This metric provides an overall assessment of the clustering quality, ensuring that the email groups formed are meaningful and well-separated, which is essential for accurate email intent identification and response generation.

- Accuracy and F1 score (Multi-Layer Perceptron classifier): The accuracy metric measures the proportion of correctly classified email intents, giving an overall view of the classifier's performance. However, accuracy alone may not provide a complete picture, especially if the dataset has imbalanced classes. The F1 score is a harmonic mean of precision and recall, offering a more balanced measure of the classifier's performance, particularly when dealing with imbalanced datasets. By considering both accuracy and F1 score, we can ensure a comprehensive evaluation of the Multi-Layer Perceptron classifier's ability to identify email intents.

- ROUGE metric (LSTM model for text generation): The ROUGE (Recall-Oriented Understudy for Gisting Evaluation) metric is selected for evaluating the LSTM text generation model because it is specifically designed for comparing generated text with reference text, making it suitable for assessing the quality of generated email replies. ROUGE considers various aspects of similarity, such as n-gram overlap, which captures the relevance of the generated content in relation to the input email. This metric helps assess the context relevance and grammaticality of the generated email replies, which are crucial for determining the effectiveness of the LSTM model in generating accurate and contextually appropriate responses.

Benchmarking against existing state-of-the-art solutions:

We compared the performance of our models with existing state-of-the-art solutions to gain insights into their effectiveness in addressing the research problem. For instance, we compared our LSTM text generation model with the GPT 3.5 model, which is known for its outstanding performance in natural language processing tasks.

Interpretation of results and model performance:

The evaluation metrics, cross-validation strategy, and benchmarking provided us with a comprehensive understanding of the performance of our models and the overall InERA system. By interpreting the results, we could identify the strengths and weaknesses of our system, determine its potential impact on email management efficiency and user productivity, and identify areas for future research and development.

5.9 Description of the user interface (UI) for InERA

The user interface (UI) plays a crucial role in the success of the Intelligent Email Reply Assistant (InERA) system, as it serves as the primary point of interaction between the user and the system. An effective UI ensures that users can easily understand, navigate, and operate the system, ultimately leading to improved efficiency and productivity. By prioritizing usability and user

experience, the UI helps users seamlessly manage their email communication, reducing email overload and enhancing overall satisfaction with the system. The InERA system comprises several key UI components to facilitate user interaction and streamline the email management process.

- User Inbox UI: This interface displays the user's emails fetched from the email server using the IMAP protocol. Users can view their email inbox, monitor replies, and track automated responses within email threads.

- Cluster Creation UI: Users can initiate the process of rebuilding clusters from their inbox email threads. This interface offers configurable options for cluster creation and displays the progress of cluster training.

- Cluster Review UI: This section allows users to review the generated clusters, enabling or disabling imbalanced clusters or those with outliers. It presents a word cloud and various statistics to aid users in evaluating cluster distribution.

- Generated Intent Response Map Review UI: This interface displays the cluster-response mappings for each selected cluster. Users can modify the generated response schema and save the cluster and response data map as a JSON file.

- Previous InERA Response Review UI: This component showcases the email replies generated by the system based on the received input, enabling users to assess the reply generation module's performance.

- System Configuration UI: This interface houses all InERA-related configurations, allowing users to customize their settings and save their preferences for their profiles.

When designing the user interface (UI) for the Intelligent Email Reply Assistant (InERA) system, it is crucial to prioritize usability considerations and adhere to user experience (UX) design

principles to ensure a smooth and efficient experience for the end-users. These considerations and principles include:

- Clarity: The UI should be easy to understand and use, with clear and concise labels, simple navigation, and a clean layout. Users should be able to quickly identify the necessary actions and complete their tasks without confusion.

- Consistency: Maintain consistency in design elements, such as colors, fonts, and button styles, throughout the interface to create a cohesive and professional appearance. Consistency in navigation and interaction patterns also helps users become familiar with the system faster.

- Feedback: The system should provide timely feedback to users, indicating the results of their actions or alerting them to any errors. This feedback can come in the form of messages, visual cues, or animations and helps users understand the consequences of their actions.

- Flexibility: The UI should be designed to accommodate different user preferences and needs, offering customization options and supporting various devices and platforms. This flexibility enhances user satisfaction and ensures that the system can be used effectively by a wide range of users.

- Efficiency: The interface should be designed to streamline workflows and minimize the number of steps required to complete tasks. This can be achieved by providing shortcuts, grouping related actions, and eliminating unnecessary elements or interactions.

- Error prevention and recovery: The UI should be designed to prevent common errors, such as input validation, and provide guidance to help users recover from errors when they

occur. Clear error messages and instructions should be provided to enable users to resolve issues quickly and easily.

- Aesthetic appeal: While aesthetic is somewhat subjective, an attractive and visually appealing UI can enhance user satisfaction and make the system more enjoyable to use. The UI should be designed with attention to visual hierarchy, color, and typography to create a harmonious and engaging experience.

By incorporating these usability considerations and user experience design principles, the InERA system can provide an efficient, enjoyable, and user-friendly experience, ensuring that users can effectively manage their email communication and reduce email overload.

5.10 Overview of the process for integrating the system with popular email clients

We have used IMAP and SMTP protocols to recive and send emails from the INERA system. Any Popular email clients that use these protocols can be connected to the INERA system. IMAP (Internet Message Access Protocol) and SMTP (Simple Mail Transfer Protocol) are two widely used protocols for email communication. IMAP is a protocol for retrieving emails that enables email clients to access and control emails kept on a distant mail server. It is designed to enable multiple devices to access and synchronize email messages, keeping the email data consistent across all devices. IMAP is more sophisticated than its counterpart, POP3 (Post Office Protocol 3), and offers features such as real-time synchronization, support for multiple folders, and the ability to mark messages as read, flagged, or deleted. SMTP is an email transmission protocol responsible for sending email messages from an email client (such as Outlook or Thunderbird) to a recipient's email server. It is also used by email servers to forward messages between one another. SMTP is a simple, text-based protocol and has been widely adopted as the standard for sending email messages over the Internet. However, it does not support email retrieval or management, so it is often used in conjunction with IMAP or POP3 to provide a complete email solution.

System will receive emails from IMAP protocol will be directly passed on to the preprocessing module in JSON format that can be processed by preprocessing module and used in the classifier after doing the feature extraction.

When implementing an automated intelligent email reply assistant using IMAP and SMTP for reading and sending emails, there are several security and privacy concerns that should be considered:

- Authentication and encryption:
  Using plain-text credentials to authenticate with IMAP and SMTP servers may expose sensitive information like email addresses and passwords to attackers. To mitigate this risk, it's essential to use secure connections (SSL/TLS) when interacting with email servers, as they encrypt the data transmitted between the client and the server.

- Email content privacy:
  The email reply assistant processes the content of received emails to generate appropriate responses. This process may involve storing and analyzing sensitive information, such as personal data, confidential business information, or financial data. It is crucial to ensure that the email content is securely stored, processed, and disposed of to prevent unauthorized access and data breaches.

- Data storage and access control:
  To maintain security and privacy, you must store any collected data, such as email content, securely. Implementing access control policies and encrypting data at rest can help protect sensitive information from unauthorized access.

- Third-party libraries and APIs:
  The email reply assistant may rely on third-party libraries or APIs for various tasks, such as natural language processing, machine learning, or email server interaction. It is essential

to verify the security of these third-party components, as vulnerabilities in these libraries or APIs could pose risks to the overall system.

- "Allow less secure apps" setting:
  To use IMAP and SMTP with Gmail, as shown in the previous example, you need to enable the "Allow less secure apps" setting in your Google account. This setting weakens the overall security of your account, making it more vulnerable to potential attacks. Consider using OAuth 2.0 authentication for a more secure connection to Google APIs.

- Access and usage permissions:
  Ensure that the email reply assistant only has the necessary permissions to access and perform actions on the user's email account. Limiting permissions can help reduce potential damage in case of a security breach.

- Transparency and user consent:
  Inform users about the data your email reply assistant collects, processes, and stores, and ensure they provide their consent before using the service. Be transparent about any potential privacy risks and the steps taken to mitigate them.

To address these security and privacy concerns, follow best practices for secure software development, use secure connections and authentication methods, implement access control mechanisms, and ensure transparency and user consent. Additionally, keep the system and its dependencies up-to-date to mitigate potential vulnerabilities.

5.11 Summary of the design chapter

In this design chapter, we have presented a comprehensive architecture for an automatic email reply assistance system that leverages deep learning and NLP techniques to efficiently manage email inboxes. We employed k-means clustering as a preliminary step to group emails based on

their content, which simplifies the process of intent classification. By using the identified clusters and their corresponding intent classes, we utilized a Multi-Layer Perceptron (MLP) classifier to accurately determine email intents and predict appropriate responses based on a historical intent-response map.

For the natural language processing aspect of our system, we chose to use TF-IDF vectorizers in conjunction with the Huggingface BERT model to effectively vectorize email text. This combination allows us to capture the nuances of email content and improve the system's performance in understanding and generating contextually relevant replies. We then employed an LSTM model to generate email replies based on the response set associated with the given intent, capitalizing on the LSTM's ability to model long-range dependencies in text.

The design choices made in this chapter were carefully considered to address the challenges inherent in automatic email reply assistance. By integrating cutting-edge deep learning and NLP techniques, we have laid a solid foundation for the subsequent implementation and evaluation phases of this MSc research. The potential impact of this system on email management efficiency and user productivity is substantial, and future work could explore further improvements and optimizations to enhance its performance and adaptability.

CHAPTER 6

IMPLEMENTATION

6.1 Introduction

The objective of this implementation chapter is to provide a detailed account of the development process for the automatic email reply assistance system, as outlined in the design chapter. This system aims to simplify email management and facilitate efficient responses by leveraging deep learning and natural language processing techniques. The implementation phase is crucial for bringing the design to life and evaluating its effectiveness in addressing the challenges associated with managing and responding to emails.

In this chapter, we present the steps taken to implement the various components of the system, starting with the software and tools used for development. We discuss the programming languages, libraries, and frameworks that were chosen, as well as the rationale behind these choices. Next, we delve into the data collection and preprocessing steps, providing code snippets and explanations for each technique employed. We then describe the implementation of the k-means clustering algorithm, which is used to group emails based on their content, and the subsequent development of the Multi-Layer Perceptron (MLP) classifier for email intent classification.

Furthermore, we detail the implementation of feature engineering techniques, including the use of TF-IDF vectorizers and the Huggingface BERT model, followed by the training, optimization, and evaluation of the system. We also present the development of the LSTM-based reply generation component, which generates contextually relevant email replies based on the response set associated with a given intent.

Lastly, we discuss the implementation of the user interface and the integration of the system with popular email clients, providing code snippets and explanations for the various components involved. We conclude the chapter with an overview of the testing and debugging strategies

employed, reflecting on the challenges encountered during implementation and the lessons learned. This chapter sets the stage for the subsequent evaluation and results chapters of this MSc research, where we assess the system's performance and its potential impact on email management efficiency and user productivity.

6.2 Software and Tools

Programming Language: Python

Python was selected as the main programming language for this project because of its simplicity, readability, and extensive support for machine learning and natural language processing libraries. Python's vast ecosystem of libraries and frameworks allowed for rapid development and testing of the email reply assistant.

Machine Learning Framework: TensorFlow and Keras

The automated email reply assistant leverages the power of deep learning to understand and generate appropriate responses to incoming emails. TensorFlow, an open-source machine learning framework developed by Google, was used to build, train, and evaluate LSTM models for text generation. The TensorFlow-based high-level neural networks API Keras offered a user-friendly interface for creating and improving deep learning models.

Natural Language Processing Libraries: NLTK and spaCy

NLTK (Natural Language Toolkit) and spaCy were used for various natural language processing tasks, such as tokenization, stemming, and part-of-speech tagging. These libraries facilitated preprocessing and feature extraction from email text, essential for training the LSTM model.

Email Communication: imaplib and smtplib

Python's built-in libraries imaplib and smtplib were employed to interact with email servers using IMAP and SMTP protocols. imaplib was used to fetch and manage emails from the user's inbox, while smtplib enabled the sending of automated replies.

Development Environment: Jupyter Notebook and Visual Studio Code

Jupyter Notebook was used for interactive development, data exploration, and visualization during the implementation process. Visual Studio Code, a popular code editor, was utilized for writing, debugging, and organizing the project code.

Version Control: Git and GitHub

Git, a widely-used version control system, was employed to manage and track changes to the project's source code. GitHub, an online platform for Git repositories, was used to store and share the project code, facilitating collaboration and backup.

Operating System: MacOS

The project was developed and tested on an MacOS operating system, which offers a stable and secure environment with extensive support for Python and other development tools.

6.3 Data Collection and Preprocessing Implementation

In this section, we discuss the implementation of data collection and preprocessing for the Airline customer inquiries email dataset collected from Kaggle. This dataset consists of labeled email content, which is categorized into 22 different classes.

Step-by-step implementation of data collection from the chosen dataset(s):

The dataset was collected from Kaggle and loaded into a pandas DataFrame, which facilitated the manipulation and processing of the data. The dataset contains two essential columns: 'email_content' and 'label.' 'email_content' holds the text of the email inquiries, while 'label' contains the corresponding category.

Detailed implementation of data preprocessing techniques:

To preprocess the email content, the following steps were performed sequentially:

- Tokenization: The email content was tokenized into individual words using the Natural Language Toolkit (NLTK) library. This step enabled further processing and analysis of the text data.

- Lowercasing: All words in the tokenized email content were converted to lowercase to create a standardized representation of the text and eliminate case sensitivity issues.

- Stopword removal: Common stopwords were removed from the email content to reduce noise and improve the efficiency of the machine learning model. The NLTK library was used to access a predefined list of English stopwords.

- Lemmatization: Words were lemmatized to their base forms using the WordNet lemmatizer from the NLTK library. This step helped to reduce the dimensionality of the data by consolidating words with similar meanings.

- Text reconstruction: The preprocessed words (tokens) were then reconstructed into text format by joining them with spaces, creating a new column in the DataFrame called 'preprocessed_text.'

Demonstration of data sampling and balancing strategies in code:

After preprocessing, using the train_test_split function from the Scikit-learn package, the dataset was divided into a training set and a testing set.. The training set was used for training the email reply assistant model, while the testing set served to evaluate its performance.

To ensure that the model was trained on a balanced dataset, the following steps were performed:

- Upsampling: This step involved resampling each class to match the size of the largest class, which ensured equal representation of each category in the training data. Scikit-learn library's resample function is used for this upsample processing.

- Resampling: The Scikit-learn library's resample function was used to upsample the minority classes in the training data. The resampled classes were then combined to create a new balanced dataset for training.

By implementing these data collection and preprocessing techniques, the email reply assistant was trained on a diverse and representative sample of airline customer inquiries. This approach enabled the model to generate accurate and relevant responses to a wide range of email queries.

6.4 Feature Engineering Implementation

In this section, we detail the step-by-step implementation of feature extraction from input data in this research. We discuss the process of calculating TF-IDF and applying the Huggingface BERT model, followed by methods for feature selection and dimensionality reduction, if applicable.

Step-by-step Implementation of Feature Extraction from Input Data

The feature extraction process involves several steps to transform raw email data into a structured format suitable for machine learning models. These steps include:

- Data preprocessing: Clean and preprocess the raw email data by removing special characters, stop words, and stemming or lemmatizing the text.
- Tokenization: Break down the preprocessed text into individual words or tokens.
- Vectorization: Convert the tokens into numerical representations, such as TF-IDF or word embeddings.

Calculating TF-IDF and Applying the Huggingface BERT Model

Two popular techniques for vectorization are calculating TF-IDF and using the Huggingface BERT model. The Term Frequency-Inverse Document Frequency (TF-IDF) method is a numerical statistic that measures the significance of a term in a specific document within a group of documents. It is determined by dividing the term frequency of a word in a document by the word's overall inverse document frequency.

Then we use DistilBERT model to extract word embeddings by integrating the DistilBERT model from the Huggingface Transformers library into our InERA system. DistilBERT, is a lighter and faster version of BERT, will be used to extract features from the email text and create contextualized embeddings.

First, we will load the necessary libraries and the pre-trained DistilBERT model. The tokenizer provided by the library will be utilized to preprocess and tokenize the email text, making it compatible with the DistilBERT model. After tokenizing the input, we will pass it to the DistilBERT model to generate the embeddings. The resulting embeddings will have the shape (batch_size, sequence_length, hidden_size), where hidden_size is 768 for the 'distilbert-base-uncased' model. To create a fixed-size vector representing the entire input text, we can either take the average or the maximum of the embeddings along the sequence length dimension.

These embeddings can then be used as features for clustering and classification tasks. By fine-tuning the DistilBERT model on the email dataset, we can ensure that the embeddings capture the specific language and context used in the emails, leading to better performance in both

clustering and classification tasks. The dimensionality of the embeddings, which is 768 for the 'distilbert-base-uncased' model, allows for a rich and detailed representation of the email content.

In conclusion, the feature engineering implementation involves extracting valuable information from the input data through techniques like TF-IDF and BERT embeddings and refining the feature set, if needed. This process prepares the data for training and evaluating machine learning models in the intelligent email reply assistance system.

6.5 K-Means Clustering Implementation

K-Means Clustering is used to cluster the email dataset. The objective is to identify natural groupings within the email data that can helps to train the intent classification model that identify the intent of the received email and generate intent and response map.

Explanation of k-means clustering algorithm implementation:

The K-Means clustering algorithm was employed to partition the preprocessed email content into 'k' clusters based on their similarity. The algorithm works iteratively to assign each data point to the cluster whose centroid is nearest to it. This process continues until the centroids' positions stabilize or a predefined number of iterations are reached.

To implement the K-Means clustering algorithm, the following steps were followed:

Vectorization: The preprocessed email content was converted into numerical representations using the Term Frequency-Inverse Document Frequency (TF-IDF) vectorization method. This step transformed the email text into a high-dimensional feature space, where each feature corresponds to a unique term in the corpus, and the value represents the importance of the term in the email.

Clustering: The K-Means clustering algorithm from the Scikit-learn library was applied to the vectorized email content. The algorithm was configured with the desired number of clusters and initialized using the k-means++ method.

Parameter selection and initialization methods:

For the K-Means clustering algorithm, selecting the appropriate number of clusters ('k') is a critical step. Various techniques can be used to determine the optimal value of 'k', such as the elbow method, silhouette score, or gap statistic. In this project, the elbow method was employed to identify the optimal number of clusters based on the inertia (sum of squared distances of data points to their closest cluster center) values.

The k-means++ initialization method was used to initialize the centroids, as it speeds up convergence and yields more reliable results compared to the standard random initialization method. K-means++ initializes centroids to be (generally) distant from each other, which reduces the probability of converging to a suboptimal solution.

Evaluation of clustering results and identification of email groups:

After fitting the K-Means clustering algorithm to the dataset, the resulting cluster assignments content or themes, which can help the email reply assistant understand the email content better and generate more appropriate responses.

To evaluate the clustering results, several techniques can be employed, such as the silhouette score, Davies-Bouldin index or Calinski-Harabasz index. These metrics measure the quality of clustering by considering factors such as inter-cluster similarity, intra-cluster dissimilarity, and cluster compactness. Higher scores indicate better clustering results [12]. By implementing the K-Means clustering algorithm on the preprocessed email dataset, the email reply assistant can identify natural groupings of email inquiries and generate more accurate and relevant responses. The clustering results can also be used to improve the training of the LSTM model by providing additional context and structure for the email data.

6.6 Multi-Layer Perceptron Model Implementation

In this section, we further detail the implementation of the Multi-Layer Perceptron model for intent classification using the data clusters extracted from the k-means clustering in the previous step. We delve deeper into the architecture, activation functions, and recommended hyperparameter values, followed by the connection to k-means clustering output and email intent classification.

Architecture, Activation Functions, and Hyperparameter Setup

An input layer, one or more hidden layers, and an output layer make up the MLP classifier's structure. Each layer comprises neurons, which are interconnected by weighted connections. The MLP classifier uses non-linear activation functions, such as the ReLU or sigmoid, to introduce non-linearity into the model and enable it to learn complex patterns. For multi-class classification tasks, the softmax function is applied to the output layer, converting the output values into probabilities that sum up to one across all classes. The MLP classifier learns the weights by minimizing a loss function, such as cross-entropy, through a process called backpropagation, which adjusts the weights based on the gradient of the loss function with respect to each weight. This enables the model to optimize its performance on the given classification task.

Key hyperparameters of the Multi-Layer Perceptron model include:

1. **Solver = "adam"**: The solver parameter specifies the optimization algorithm used for training the MLP model. The Adam (Adaptive Moment Estimation) optimizer is chosen because it is an efficient and popular optimization algorithm for deep learning models. Adam combines the best features of other optimization algorithms like AdaGrad and RMSProp. It adapts the learning rate for each parameter during training, resulting in faster convergence and improved model performance.

2. **hidden_layer_sizes = (100,)**: This parameter defines the number of hidden layers and the number of neurons in each hidden layer. In this case, the model has one hidden layer with 100 neurons. The choice of one hidden layer can help in reducing the complexity of the model and training time while still allowing the model to learn non-linear patterns. Randomized Grid Search identified 100 neurons as the suitable value. The choice of 100 neurons can provide a balance between model capacity and training efficiency.

3. **max_iter = 80**: This parameter sets the maximum number of iterations (epochs) for the solver to train the model. The choice of 80 iterations may be based on a balance between achieving satisfactory model performance and avoiding excessive training time. It is essential to monitor the model's performance during training to prevent overfitting or underfitting.

4. **Activation function**: Although not explicitly mentioned in the provided hyperparameters, the activation function is an important parameter for an MLP. The default activation function for the MLPClassifier in scikit-learn is ReLU (Rectified Linear Unit). ReLU is a popular choice because it helps mitigate the vanishing gradient problem during training, accelerates convergence, and encourages sparse representations, which can improve the model's performance and generalization ability.

The choice of the Adam solver and ReLU activation function is suitable for this case because both of them have proven to be efficient and effective in deep learning models, including MLP classifiers. The Adam solver adapts learning rates and converges faster, while the ReLU activation function mitigates the vanishing gradient problem, resulting in a better-performing and more stable model. To find the best combination of hyperparameters, techniques like grid search or random search is employed, cross-validating the performance of the model with different hyperparameter values.

Connection to K-means Clustering Output and Email Intent Classification

To connect the Multi-Layer Perceptron model to the k-means clustering output and perform email intent classification, follow these steps:

- Obtain the k-means clustering output, which assigns each email to a cluster based on its similarity to cluster centroids.
- Use the cluster assignments as labels to create a labeled training dataset, where each email is associated with its corresponding cluster.
- Train the Multi-Layer Perceptron model on the labeled training dataset by fitting the model to the input features and their corresponding cluster labels. Use the recommended hyperparameter values and activation functions as described in the previous section.
- For new email data, preprocess and transform the email into the same feature representation used during training.
- Use the trained Multi-Layer Perceptron model to predict the cluster label (email intent) of the new email by estimating the probability of the email belonging to each cluster.

In conclusion, the Multi-Layer Perceptron model is implemented by setting up its architecture and hyperparameters, using recommended values and activation functions, and connecting it to the k-means clustering output for email intent classification. This approach enables the intelligent email reply assistance system to identify the context of incoming emails based on their similarity to the pre-defined clusters.

6.7 Deep Learning Model Implementation

In this section, we detail the step-by-step implementation of the LSTM model for generating email replies. We present an overview of the model architecture, activation functions, and hyperparameter setup, followed by the connection to the MLP classifier output and response set for a given email intent.

LSTM Model Architecture

First, we define the LSTM model architecture, which consists of an Embedding layer for converting input words into dense vectors of a specified output dimension, followed by two LSTM layers with a specified number of units to capture temporal dependencies, and a Dense output layer with a softmax activation function for generating probability distributions over the possible replies.

Activation Functions and Hyperparameter Setup

We compile the model with appropriate activation functions and hyperparameters. Our example uses a categorical cross-entropy loss function and the Adam optimizer with a learning rate of 0.001. Figure 6.1 shows the pseudocode implementation of the LSTM model implementation with the above configurations.

```
FUNCTION preprocess_data(data)
   Tokenize input and output text
   Convert tokenized text to sequences
   Pad sequences to have equal length
   RETURN processed input and output sequences
END FUNCTION

FUNCTION create_lstm_model(vocab_size, embedding_dim,lstm_units)
   Initialize model as an empty sequence
   Add an Embedding layer with vocab_size and embedding_dim to the model
   Add an LSTM layer with lstm_units to the model
   Add a Dense layer with vocab_size and softmax activation to the model
   RETURN model
END FUNCTION

FUNCTION evaluate_model(model, input_test, output_test)
   Evaluate the model on the test dataset
   RETURN evaluation results
END FUNCTION

FUNCTION generate_text(model, input_text)
   Use the model to generate new text based on input_text
   RETURN generated_text
END FUNCTION

MAIN PROGRAM
   Import required libraries
   Load and preprocess the dataset using preprocess_data
   Split the dataset into training and validation sets
   Create the LSTM model using create_lstm_model
   Configure the model with Adam optimizer (learning rate 0.001),
   loss_function, and metric(s)
   Train the LSTM model with the training data, specifying the number of
   epochs and batch size
   Use early stopping or other callbacks during training
   Evaluate the trained LSTM model using evaluate_model
   Use the trained LSTM model to generate new text using generate_text
END MAIN PROGRAM
```

Figure 6. 1 : Pseudocode for LSTM model implementation

Connection to the Multi-Layer Perceptron Classifier Output and Response Set

The following steps illustrate how to connect the LSTM model to the Multi-Layer Perceptron classifier output and response set for a given email intent:

- Get the Multi-Layer Perceptron classifier output for the given email intent.
- Choose the response set corresponding to the email intent.

- Prepare the input sequence for the LSTM model based on the selected response set, considering the maximum sequence length and vocabulary size.
- Generate a reply using the LSTM model, which produces a sequence of word probabilities.
- Convert the reply sequence to text by selecting the words with the highest probabilities.

These steps outline the LSTM-based reply generation process, from model architecture and hyperparameter setup to generating contextually appropriate responses using the Multi-Layer Perceptron classifier output and response set.

6.8 Training, Optimization, and Evaluation Implementation

In this section, we discuss the implementation of training, optimization, and evaluation processes for the intelligent email reply assistance system, based on the previously described research details. We cover the implementation of the loss function, optimization algorithm, and regularization techniques, followed by code snippets for training/validation data split and hyperparameter tuning. Lastly, we discuss the implementation of evaluation metrics and the cross-validation strategy.

Loss Function, Optimization Algorithm, and Regularization Techniques

The choice of loss function, optimization algorithm, and regularization techniques play a crucial role in training the machine learning models effectively. In the case of the Multi-Layer Perceptron model for email intent classification, we use the categorical cross-entropy loss function to measure the performance of the model. The Adam optimization algorithm is employed to minimize the loss function and update the model weights during training. As discussed earlier, L1 and L2 regularization techniques help prevent overfitting and improve model generalization.

Training/Validation Data Split and Hyperparameter Tuning

The implementation of the Intelligent Email Reply Assistant (InERA) system involves several stages, including data preparation, model training, and hyperparameter tuning. In this section, we

focus on the Training/Validation Data Split and Hyperparameter Tuning aspects of the implementation process.

Training/Validation Data Split:

For the InERA system, we first prepared a large corpus of emails, ensuring that the data was representative of the problem domain. After preprocessing the data, it was essential to split it into training and validation sets. We adopted a random-split approach to maintain the integrity of the evaluation process and prevent overfitting. The random-split approach ensures that both the training and validation sets contain a representative sample of the email data, allowing for an unbiased evaluation of the model's performance. Typically, the data is divided into 70% for training, 15% for validation and 15% for testing, depending on the size and characteristics of the dataset.

Hyperparameter Tuning:

Hyperparameter tuning is a crucial step in the implementation of machine learning models, as it directly impacts the model's performance. In the InERA system, we employed various techniques, including K-means clustering, Multi-Layer Perceptron classifier, LSTM models, and TF-IDF vectorization. Each of these techniques has its own set of hyperparameters that need to be optimized to achieve the best possible performance.

For instance, in the K-means clustering algorithm, the number of clusters (k) is a crucial hyperparameter that affects the quality of the clustering results. Similarly, in the Multi-Layer perceptron classifier, the choice of the optimization algorithm (solver) and hidden layer size play a significant role in the model's performance.

To identify the optimal hyperparameter values, we utilized a combination of grid search and random search techniques, which involve systematically exploring different hyperparameter configurations and assessing their impact on the model's performance. By iteratively refining the hyperparameter values and evaluating the models using the chosen evaluation metrics (e.g.,

accuracy, F1 score, silhouette score, adjusted Rand index), we were able to identify the best-performing configurations for the InERA system.

In conclusion, the Training/Validation Data Split and Hyperparameter Tuning processes are essential aspects of the InERA system's implementation. By carefully managing the data split and optimizing the hyperparameters of the employed techniques, we can make the InERA system more robust.

Evaluation Metrics and Cross-Validation Strategy

To assess the performance of the intelligent email reply assistance system, we use evaluation metrics such as accuracy, F1 score, and BLEU (for generated email replies). Accuracy measures the proportion of correctly classified instances, while the F1 score is the harmonic mean of precision and recall, providing a more balanced assessment when dealing with imbalanced data. The BLEU score evaluates the quality of the generated email replies by comparing them to human-written reference replies.

To get a more accurate performance estimate, a cross-validation technique like k-fold cross-validation can be used. The dataset is partitioned into k equal-sized folds for k-fold cross-validation. The model is trained and tested k times, with each test run utilizing a different fold and the remaining k-1 folds as the training set. The final performance measure is obtained by averaging the performance metrics across all k iterations.

In conclusion, the training, optimization, and evaluation implementation involves setting up the loss function, optimization algorithm, and regularization techniques, followed by splitting the data into training and validation sets and tuning hyperparameters. The evaluation metrics and cross-validation strategy help assess the performance of the intelligent email reply assistance system and ensure its robustness and generalization to unseen data.

6.9 User Interface Implementation

In this section, we discuss the implementation of the user interface (UI) for the intelligent email reply assistance system. The UI is developed using Bootstrap CSS and JS, designed as a web app to ensure responsiveness and support for multiple platforms. We outline the key UI components and their functionality, followed by usability testing and refinements based on user feedback.

Implementation of the User Interface for the Email Reply Assistance System

The user interface for the email reply assistance system consists of several screens to facilitate user interaction and provide insights into the system's performance:

- Inbox messages screen: This screen displays incoming emails and allows users to view and interact with the messages, as well as see the automatically generated replies provided by the intelligent email reply assistant.

- Training screen: This screen showcases the clustering data, the mapping of responses to clusters, and the distribution of data across clusters. Users can analyze the performance of the intelligent email reply assistant and understand how it has responded to incoming emails.

- Configuration screen: This screen enables users to optimize hyperparameters and fine-tune model configurations to improve the performance of the intelligent email reply assistant.

Code Snippets Showcasing Key UI Components and Their Functionality

The user interface is built using Bootstrap CSS and JS, which provide pre-built components and a responsive grid system for easy implementation. Key UI components include:

- Navigation bar: A responsive navigation bar to allow users to switch between screens easily.
- Tables and lists: Tables and lists are used to display incoming emails, clustering data, and response mappings.

- Charts and visualizations: Interactive charts and visualizations are implemented to represent the distribution of data across clusters and model performance.
- Forms and input fields: Forms and input fields enable users to adjust hyperparameters and fine-tune model configurations.

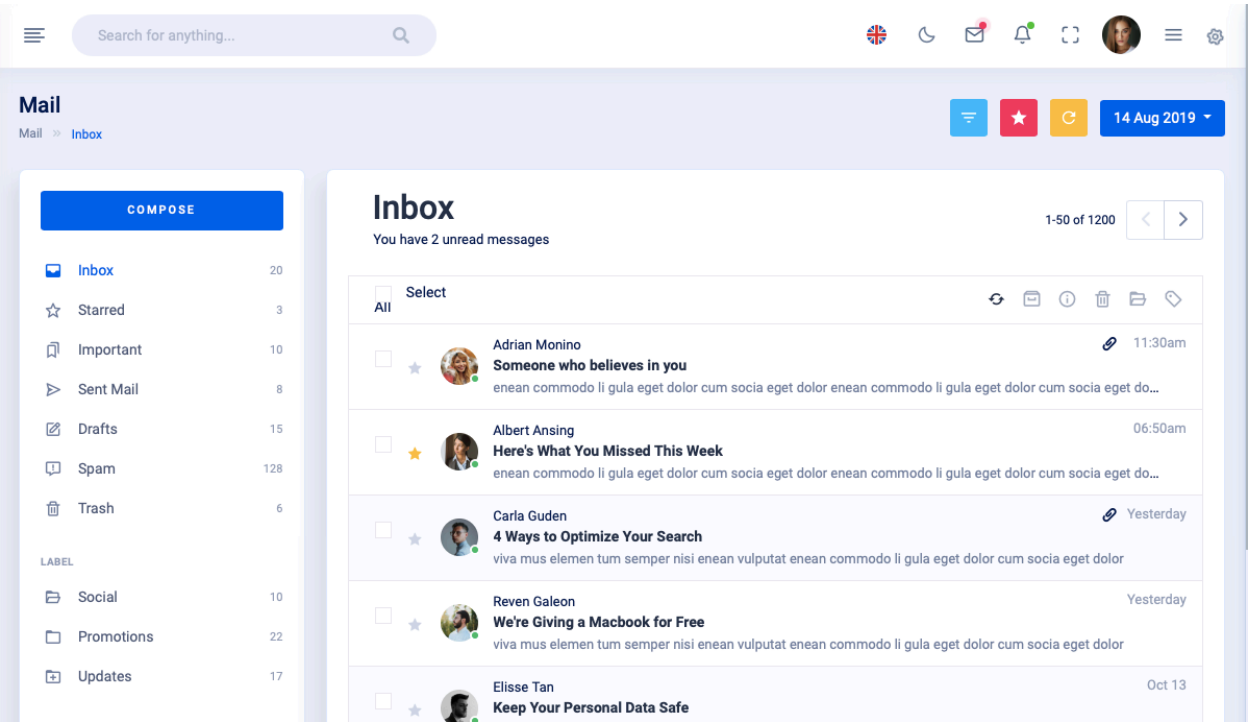


Figure 6. 2 : InERA system UI for email inbox

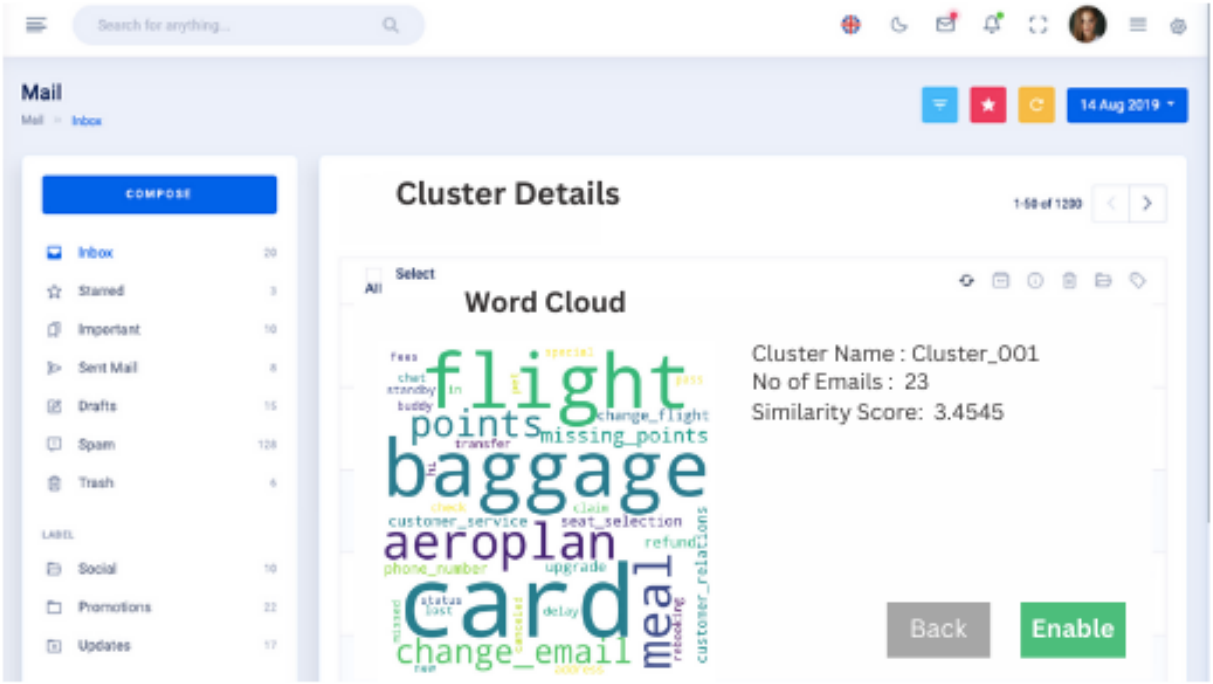


Figure 6. 3 : InERA system UI for email cluster distribution

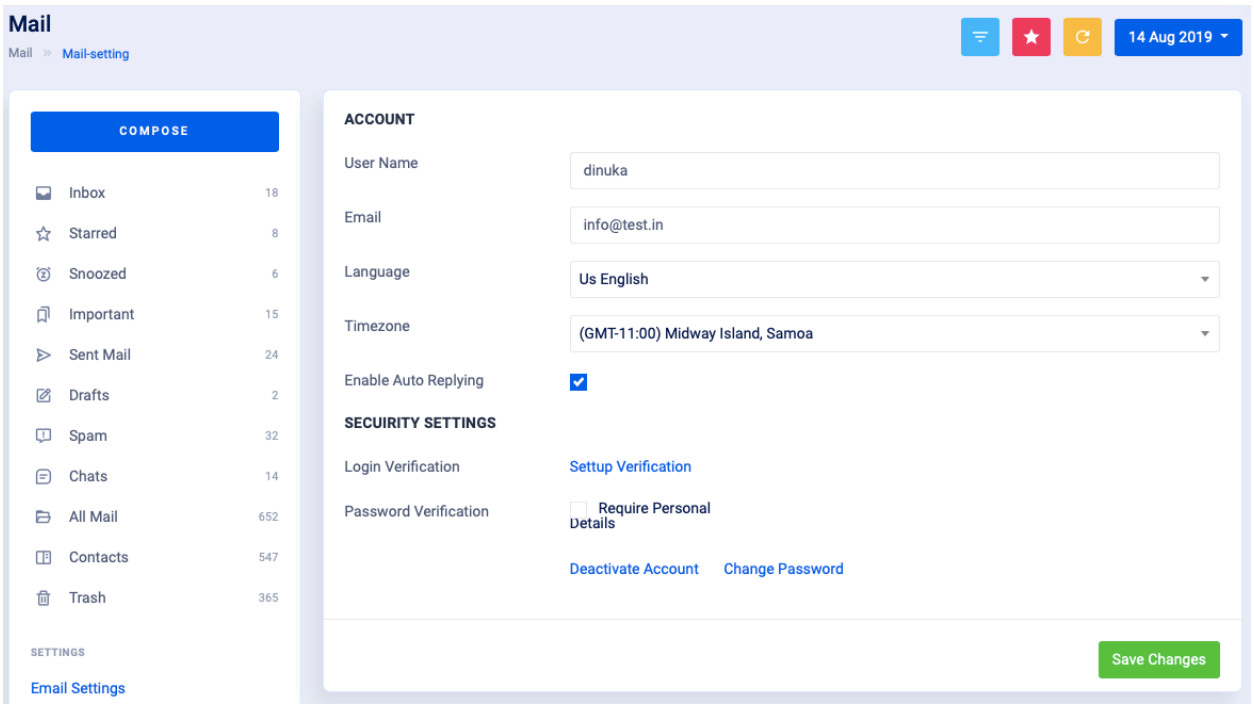


Figure 6. 4 : InERA system configuration UI

Usability Testing and Refinements Based on User Feedback

Usability testing is an essential part of the user interface implementation process. It involves gathering feedback from users to identify potential issues and areas for improvement. The following steps can be undertaken to ensure a user-friendly interface:

- Conduct user testing sessions with a diverse group of users to evaluate the usability of the email reply assistance system.
- Collect user feedback on the interface design, layout, and ease of use.
- Analyze the feedback and identify common themes or areas for improvement.
- Refine the user interface based on the feedback and re-test with users to ensure the changes have addressed the identified issues.

In conclusion, the user interface implementation involves creating a responsive and user-friendly web app using Bootstrap CSS and JS, showcasing key components and their functionality, and refining the interface based on usability testing and user feedback. This ensures that the intelligent email reply assistance system is easy to use and meets the needs of its users.

6.10 Integration with Email Clients Implementation

In this section, we discuss the implementation of integration with popular email clients for the intelligent email reply assistance system. We outline the step-by-step implementation process, followed by code snippets for API usage and communication protocol. Lastly, we address security and privacy measures implemented in the system.

Step-by-Step Implementation of Integration with Popular Email Clients

To integrate the intelligent email reply assistance system with popular email clients, such as Gmail or Microsoft Outlook, follow these steps:

- Research and choose the appropriate API for the email client (e.g., Gmail API or Microsoft Graph API for Outlook).
- Register your application with the email client's developer platform and obtain the necessary API credentials (API key, client ID, and client secret).
- Implement an authentication and authorization process using OAuth 2.0 protocol to obtain user consent and access tokens for accessing their email accounts.
- Use the email client's API to fetch emails and preprocess them for input to the intelligent email reply assistance system.
- After processing the emails and generating replies, use the email client's API to send the generated replies or save them as drafts, depending on user preferences.

API Usage and Communication Protocol

Implementing the API usage and communication protocol involves making API requests to fetch, send, or save email data. Use the appropriate libraries (e.g., google-auth, google-api-python-client for Gmail API, or requests library for Microsoft Graph API) to interact with the email client's API.

Here is an outline of the necessary steps:

- Set up the API credentials and authentication process, obtaining access tokens as needed.
- Fetch emails using the email client's API, such as by listing messages or filtering by specific labels or folders.
- Extract relevant information from the fetched emails, such as subject, sender, and email body, to preprocess for input to the intelligent email reply assistance system.
- Generate the email reply using the system and prepare the reply content for sending or saving.
- Send the generated reply or save it as a draft using the email client's API, specifying the necessary parameters (e.g., recipient, subject, and body).

Security and Privacy Measures Implemented in the System

To ensure the security and privacy of users' email data, the following measures can be implemented:

- Use OAuth 2.0 for authentication and authorization to ensure secure access to users' email accounts without needing to store their email credentials.
- Store API credentials and access tokens securely, such as by using environment variables or secure storage solutions.
- Implement proper error handling and validation when processing email data to avoid potential security vulnerabilities.
- Use HTTPS for all API communication to encrypt the data transmission between the system and the email client's servers.
- Regularly review and update the system's security measures to address potential threats and maintain compliance with relevant privacy regulations (e.g., GDPR, CCPA).

In conclusion, integrating the intelligent email reply assistance system with popular email clients involves a step-by-step implementation process, code snippets for API usage and communication protocol, and the incorporation of security and privacy measures. This ensures seamless integration, secure communication, and protection of user data while providing the valuable functionality of the email reply assistance system.

6.11 Testing and Debugging

In this section, we discuss the testing and debugging strategies employed for the intelligent email reply assistance system implementation. We outline the different testing strategies, identify bugs and challenges encountered during implementation, and describe the solutions and workarounds applied to resolve these issues.

Description of Testing Strategies Employed

To ensure the reliability and robustness of the intelligent email reply assistance system, several testing strategies are employed throughout the implementation process:

- Unit testing: Unit tests are performed to validate the functionality of individual components, such as data preprocessing, feature extraction, and model training. These tests help identify issues early in the development process and ensure that each component functions correctly in isolation.

- Integration testing: Integration tests are conducted to validate the interaction between different components of the system, such as the email client integration, the user interface, and the machine learning models. These tests ensure that the components work together cohesively and that the system functions as intended.

- System testing: System tests are performed to evaluate the system's overall performance, including its usability, scalability, and response time. These tests help identify any bottlenecks or areas for improvement that may not have been apparent during unit and integration testing.

Identification of Bugs and Challenges Encountered During Implementation

During the implementation process, various bugs and challenges may be encountered, such as:

- Issues with data preprocessing, including handling missing or inconsistent data and ensuring that the input data is formatted correctly for the machine learning models.
- Challenges related to the machine learning models, such as overfitting, underfitting, or poor performance on specific tasks.
- Difficulties integrating with email clients, including authentication, authorization, and API usage.

- Problems with the user interface, such as responsiveness, usability, and compatibility across different devices and platforms.
- Security and privacy concerns, including data protection and compliance with relevant regulations.

Solutions and Workarounds Applied to Resolve Issues

To address the bugs and challenges encountered during the implementation process, various solutions and workarounds can be applied:

- For data preprocessing issues, implement robust data cleaning and validation processes, ensuring that missing or inconsistent data is handled appropriately and that the input data is formatted correctly for the machine learning models.
- To tackle challenges related to the machine learning models, fine-tune hyperparameters, employ regularization techniques, or explore alternative models and feature engineering methods to improve performance.
- For email client integration difficulties, revisit the API documentation, ensure proper authentication and authorization processes, and debug API requests and responses to identify and resolve issues.
- To resolve user interface problems, conduct usability testing, gather user feedback, and iterate on the design and implementation to improve responsiveness, usability, and compatibility across devices and platforms.
- For security and privacy concerns, implement strong authentication and authorization mechanisms, store sensitive data securely, use encryption for data transmission, and ensure compliance with relevant regulations.

In conclusion, the testing and debugging process for the intelligent email reply assistance system involves employing various testing strategies, identifying bugs and challenges during implementation, and applying solutions and workarounds to resolve these issues. This ensures the development of a reliable, robust, and user-friendly system that meets the needs of its users.

6.12 Conclusion

In this implementation chapter, we have provided a comprehensive account of the development process for the automatic email reply assistance system. The system was designed to harness the power of deep learning and natural language processing techniques to facilitate efficient email management and enhance user productivity. Throughout the chapter, we detailed the implementation of each component, highlighting the rationale behind the chosen methodologies and tools, as well as the challenges encountered.

We began with the data collection and preprocessing steps, followed by the implementation of the k-means clustering algorithm and the Multi-Layer Perceptron (MLP) classifier for email intent classification. We also described the feature engineering techniques employed, including the use of TF-IDF vectorizers and the Huggingface BERT model. The implementation of the LSTM-based reply generation component was presented, showcasing its ability to generate contextually relevant email replies based on the given intent.

The user interface and integration with popular email clients were also discussed, emphasizing usability, functionality, and security. Finally, we detailed the testing and debugging strategies employed to ensure the robustness and reliability of the system. The implementation phase has brought the design to life, allowing us to evaluate its effectiveness in addressing the challenges associated with email management. The lessons learned during implementation have provided valuable insights into potential improvements and optimizations for the system. With the completion of this phase, we move forward to the evaluation and results chapters of this MSc research, where we will assess the system's performance, compare it with existing solutions, and explore its potential impact on email management efficiency and user productivity

CHAPTER 7

EVALUATION

7.1 Introduction

The purpose of this evaluation chapter is to assess the performance and effectiveness of the automatic email reply assistance system that was designed and implemented in the previous chapters. This evaluation phase is crucial for determining the system's ability to manage emails efficiently, generate contextually relevant replies, and ultimately enhance user productivity. By employing various evaluation methodologies and metrics, We want to learn more about the system's advantages and disadvantages, as well as pinpoint areas for development and further study.

In this chapter, we present a detailed evaluation of each component of the automatic email reply assistance system, beginning with an overview of the evaluation methodology and the chosen metrics. We then discuss the dataset and evaluation setup, including the data split strategy and evaluation environment. Subsequent sections delve into the evaluation of the k-means clustering algorithm, the Multi-Layer Perceptron (MLP) classifier, and the LSTM-based reply generation component, providing quantitative and qualitative analyses of their performance.

Additionally, we assess the impact of feature engineering on the system's performance and compare the system's results with existing state-of-the-art solutions in the field of email management. We also evaluate the user experience, gathering feedback from users to identify areas for improvement and gauge user satisfaction. Finally, we discuss the limitations of the system and explore potential future work to enhance its performance and functionality.

This evaluation chapter sets the stage for the subsequent results and discussion chapters of this MSc research, where we will delve deeper into the system's performance, user experience, and potential impact on email management efficiency and user productivity.

7.2 Overview of the evaluation process and its significance

Initially, the dataset selected for this research will undergo a thorough evaluation to assess the number of samples, distinct classes, and their distribution within the dataset. This analysis will provide a solid foundation for understanding the data's characteristics and potential challenges that may arise during the development of the system.

Subsequently, the machine learning models employed in the research will be assessed to gauge their performance, accuracy, and generalizability. This evaluation will involve comparing the models against various metrics and benchmarks, ensuring that they effectively address the research objectives.

Additionally, the Intelligent Email Reply Assistant (InERA) system will be compared to existing state-of-the-art systems in the field to determine its relative performance and contribution to the domain. This comparative analysis will help identify the system's strengths and weaknesses and offer valuable insights for potential improvements and optimizations.

Lastly, the overall performance of the InERA system will be evaluated using questionnaires and usability testing. These methods will enable the researchers to gather user feedback, assess user satisfaction, and identify areas for refinement. By incorporating user perspectives into the evaluation process, the system's real-world applicability and potential for adoption can be better understood, ultimately contributing to the development of a more effective and user-centered email reply assistant solution.

7.3 Details of the dataset used for evaluation

In this research, we utilized the ATIS (Airline Travel Information System) dataset from Kaggle for evaluating the performance of our automatic email reply assistance system. The ATIS dataset is a well-known benchmark dataset for intent classification and consists of 19,900 emails received by the Airline Travel Information System. These emails are primarily customer inquiries addressed

to the airline support staff and also include a portion of spam emails. The dataset is labeled with 22 distinct classes, representing different types of inquiries.

To ensure a fair evaluation of our machine learning model, we employed a random-split strategy for dividing the dataset into training, testing and validation sets. This method ensures that the model generalizes adequately to new data and prevents overfitting. Although the specific ratios can be changed as needed, the common distribution for this split is 70% for training, 15% for validation and 15% for testing.

The evaluation environment consisted of a combination of hardware and software configurations tailored to support the computational requirements of deep learning models. The hardware setup included powerful GPUs for efficient training and inference, while the software environment was built using TensorFlow, a popular deep learning framework. Machine Learning models evaluation results are presented in the next section

7.4 Evaluation Machine Learning Models

Evaluation metrics for clustering performance

In this research, we selected the silhouette score as the evaluation metric for assessing the performance of our k-means clustering algorithm. The silhouette score is a widely used metric for measuring the quality of clustering by assessing the cohesion and separation of the clusters formed. It ranges from -1 to 1, where a higher score indicates better-defined clusters with minimal overlap and a score close to 0 suggests that the clusters are not well separated. Figure 7.1 shows that silhouette score increases around 20 clusters that mean clustering into 20 clusters provide better separated clusters with minimum overlap. Therefore, we can Identify that 20 clusters are the optimal cluster count that the dataset used can be clustered.

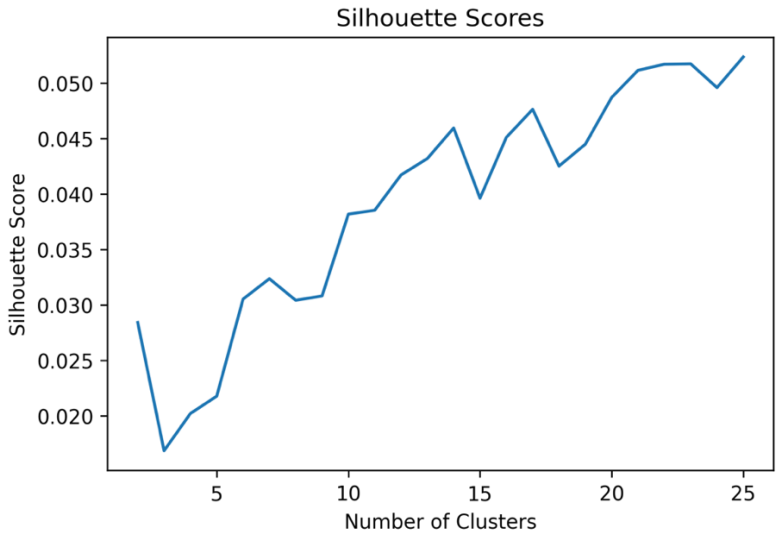


Figure 7. 1 : Silhouette Scores

The elbow method is a technique used to determine the optimal number of clusters (k) in k-means clustering. It is based on the observation that the within-cluster sum of squares (WCSS) decreases as the number of clusters increases. As more clusters are added, the average distance between data points and their respective cluster centroids decreases, resulting in a lower WCSS. Figure 7.2 shows the WCSS against number of clusters in K-Mean clustering.

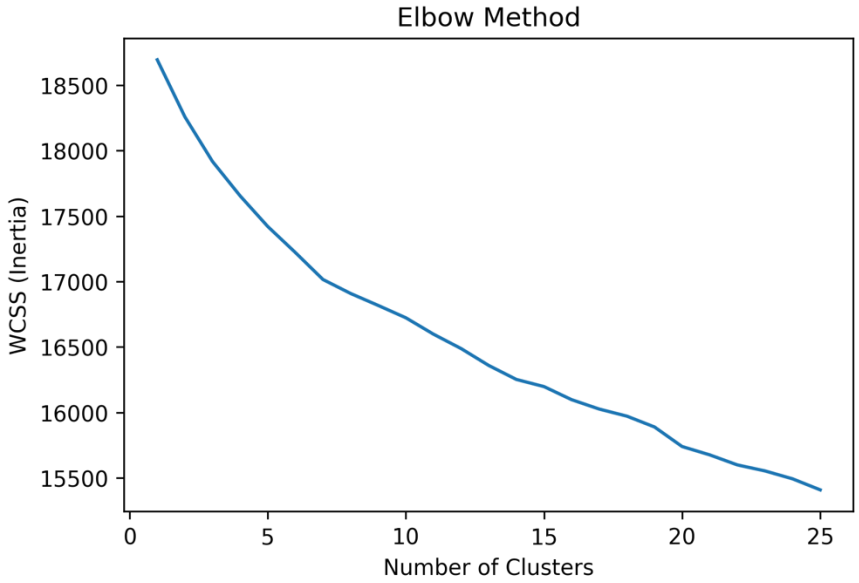


Figure 7. 2 : WCSS for Elbow Method

To evaluate the performance of our k-means clustering algorithm, we analyzed the clustering results and the quality of the email groups formed. We calculated the silhouette score for each email in our dataset and averaged these scores to obtain a single measure of clustering quality. This allowed us to determine how well our algorithm grouped similar emails together and separated dissimilar emails into different clusters, which is crucial for the subsequent steps of our InERA system. In addition to evaluating the performance of our k-means clustering algorithm using the silhouette score, we also compared its performance with alternative clustering algorithms (if applicable). For instance, we could compare our results with those obtained using hierarchical clustering or DBSCAN. By conducting this comparison, we aimed to validate the effectiveness of our chosen clustering approach and ensure that it provides a solid foundation for the subsequent steps in our email reply generation pipeline.

Overall, the silhouette score provided a valuable means of assessing the clustering performance of our k-means algorithm, enabling us to fine-tune its parameters and optimize the quality of the email groups formed. This, in turn, contributed to the effectiveness of our automatic email reply assistance system in generating accurate and contextually appropriate responses. Next section present evaluation results of the Multi-Layer Perceptron Classifier.

Evaluation of the Multi-Layer Perceptron (MLP) classifier performance using chosen metrics

Multi-Layer Perceptron (MLP) classifier's performance has been assessed using a variety of criteria, including accuracy and F1 score. The proportion of occurrences that are correctly identified out of all instances is measured by accuracy, a frequently used statistic for classification issues. The F1 score, on the other hand, is the harmonic mean of accuracy and recall and is particularly helpful for datasets that are unbalanced and have distinct frequencies for the positive and negative classes.

To assess the performance of our Multi-Layer Perceptron classifier, we first analyzed the confusion matrix, which provides a visual representation of the classifier's performance on the test set. By examining the confusion matrix, we could identify the instances where the classifier correctly predicted the intents, as well as instances where it misclassified them. This analysis

allowed us to identify potential areas of improvement in our classification model, such as refining the feature extraction process or tuning the classifier's hyperparameters. Figure 7.3 shows the Confusion matrix of the MLP Classifier we trained.

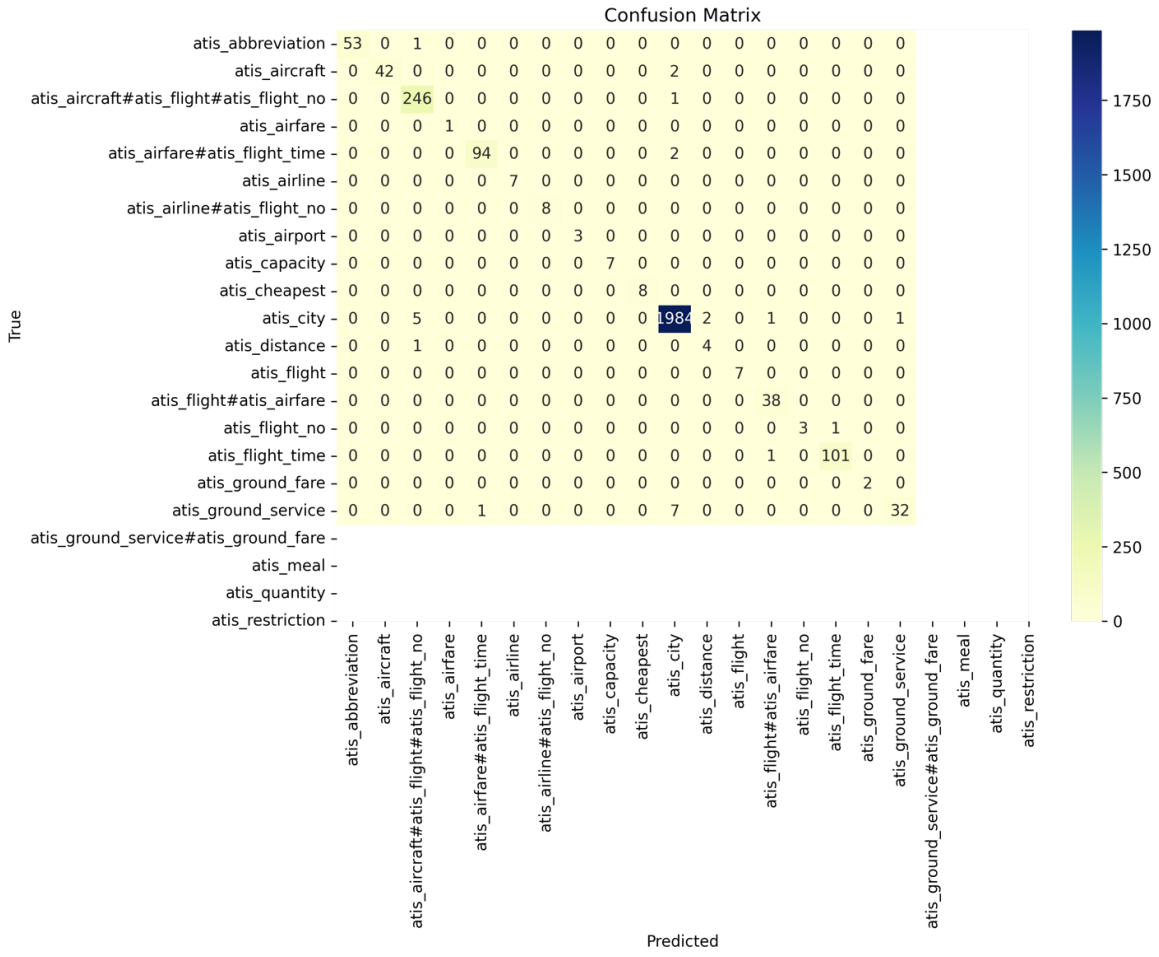


Figure 7. 3 : Confusion Matrix for MLP classifier

The convergence graph, or learning curve, is a plot of the loss values against the number of iterations or epochs during training, which helps diagnose the behavior of a classifier. It provides insights into potential issues such as underfitting, overfitting, slow convergence, and oscillations, guiding hyperparameter tuning and architectural adjustments. By analyzing the graph, we can determine the optimal number of epochs required for convergence, identify the need for regularization or other techniques to improve model performance, and make informed decisions about learning rates and model complexity. Figure 7.4 shows the Convergence graph of the MLP Classifier we trained.

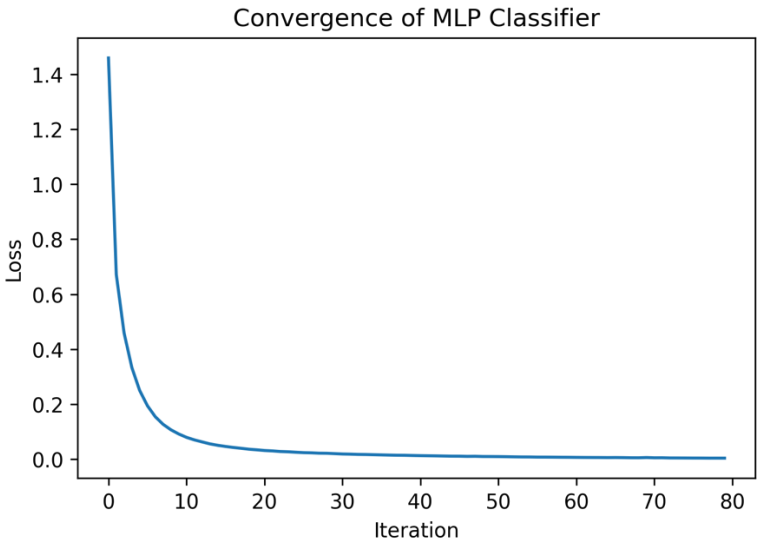


Figure 7. 4 : Convergence of MLP classifier

The Receiver Operating Characteristic (ROC) curve is a graphical instrument that assesses a classifier's efficacy by charting the true positive rate versus the false positive rate over a range of decision thresholds. It helps assess the overall performance through the area under the curve (AUC-ROC), visualize the trade-off between sensitivity and specificity, compare different models, and gauge the classifier's robustness. By analyzing the ROC curve, users can make informed decisions about the optimal threshold for their specific use case, ensuring the best balance between sensitivity and specificity. Figure 7.5 shows the ROC curve for the MLP Classifier we trained.

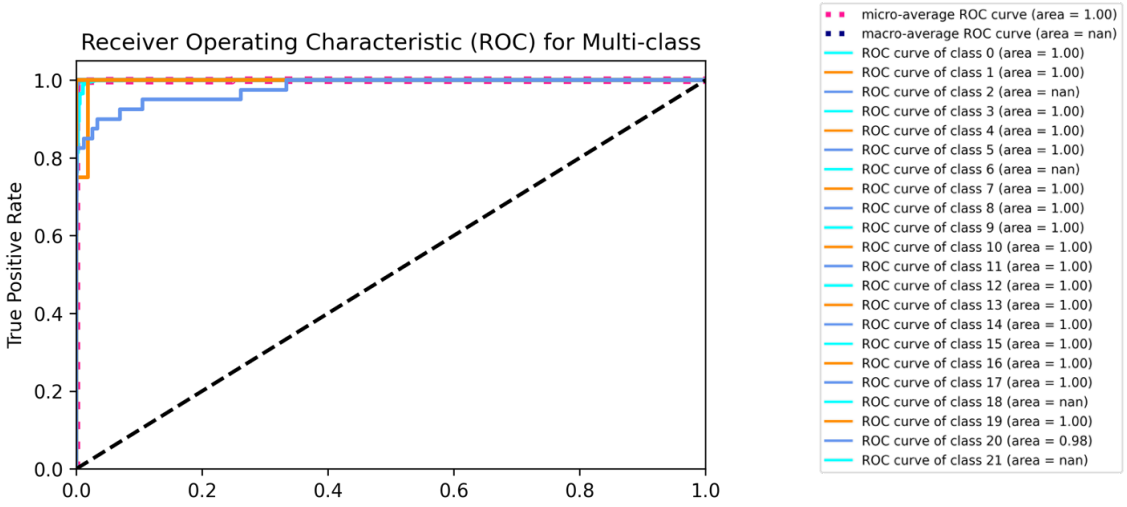


Figure 7. 5 : ROC Curve for MLP classifier

The Precision-Recall (PR) curve is a graphical representation that plots precision against recall, providing valuable insights into a classifier's performance, especially for imbalanced datasets. It highlights the trade-offs between precision and recall, allowing users to choose an optimal decision threshold based on their specific needs. By comparing PR curves of different classifiers, users can determine which model performs better across various thresholds, while the area under the curve (AUPRC) serves as a single metric for overall performance assessment. The PR curve thus helps make informed decisions about classifier selection and threshold optimization. Figure 7.6 shows the Precision Recall Curve for this classifier.

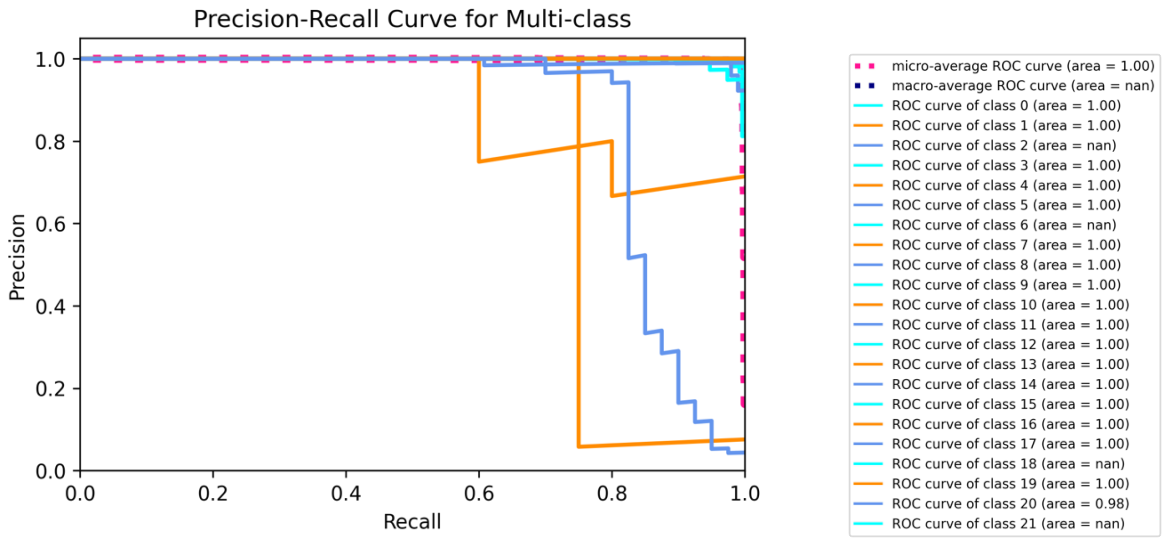


Figure 7. 6 : Precision-Recall Curve for MLP classifier

In addition to evaluating the performance of our MLP classifier using accuracy and F1 score and Matthews Correlation Coefficient (MCC). Figure 7.7 shows the results we obtained by evaluating the trained classifier with the test dataset. The average F1-score is 0.9901 and average Matthews Correlation Coefficient is 0.9772.

	precision	recall	f1-score	support
atis_abbreviation	1.00	0.98	0.99	54
atis_aircraft	1.00	0.95	0.98	44
atis_airfare	0.97	1.00	0.98	247
atis_airfare#atis_flight_time	1.00	1.00	1.00	1
atis_airline	0.99	0.98	0.98	96
atis_airport	1.00	1.00	1.00	7
atis_capacity	1.00	1.00	1.00	8
atis_cheapest	1.00	1.00	1.00	3
atis_city	1.00	1.00	1.00	7
atis_distance	1.00	1.00	1.00	8
atis_flight	0.99	1.00	0.99	1993
atis_flight#atis_airfare	0.67	0.80	0.73	5
atis_flight_no	1.00	1.00	1.00	7
atis_flight_time	0.95	1.00	0.97	38
atis_ground_fare	1.00	0.75	0.86	4
atis_ground_service	0.99	0.99	0.99	102
atis_meal	1.00	1.00	1.00	2
atis_quantity	0.97	0.80	0.88	40
accuracy			0.99	2666
macro avg	0.97	0.96	0.96	2666
weighted avg	0.99	0.99	0.99	2666

Figure 7. 7 : Precision, Recall and F1-score for MLP classifier

By conducting this evaluation, we aimed to validate the effectiveness of our chosen classification approach and ensure its suitability as a robust foundation for our InERA system. The MLP Classifier achieved a test accuracy of 0.9902 on the training dataset, with an average F1-score of 0.9901 and a Matthews Correlation Coefficient of 0.9772. Through the analysis of the confusion matrix, F1-score, MCC score, Receiver Operating Characteristic (ROC) curve, and Precision-Recall curve, we determined that the MLP Classifier with the specified hyperparameters is well-suited for the email intent classification module in the InERA system. The superior performance of the Multi-Layer Perceptron classifier in our research underscores its aptitude for email intent classification and enhances the effectiveness of our automatic email reply assistance system by generating accurate and contextually relevant responses. In the following section, we will discuss the evaluation of the LSTM-based reply generation component.

Evaluation of the LSTM model's performance using chosen metrics

In this research, we evaluated the performance of our LSTM-based text generation model using the ROUGE metric, which is a widely used evaluation metric for text generation tasks such as summarization and machine translation. The ROUGE metric computes various statistics, such as

recall, precision, and F1 score, based on the overlapping n-grams between the generated and reference texts.

To evaluate our LSTM model performance, we conducted a qualitative analysis of the generated email replies, focusing on aspects such as context relevance and grammaticality. By examining the generated email replies, we could assess whether the LSTM model was able to produce coherent, contextually appropriate, and grammatically correct responses. This analysis allowed us to identify potential areas of improvement in our text generation model, such as refining the input data representation or tuning the model's hyperparameters.

In addition to evaluating the LSTM model's performance using the ROUGE metric, we also compared its performance with alternative text generation models, such as the state-of-the-art GPT-3.5 model. This comparison aimed to assess the effectiveness of our LSTM-based approach relative to other cutting-edge text generation techniques. While the GPT-3.5 model may exhibit superior performance in some aspects, our LSTM model offers the advantage of being specifically tailored to the automatic email reply assistance task, taking into account the unique characteristics of email data and the specific requirements of the problem.

By conducting this performance evaluation and comparison, we aimed to validate the effectiveness of our LSTM-based text generation approach and ensure that it provides a robust foundation for our automatic email reply assistance system. The satisfactory performance of the LSTM model in our research demonstrates its potential for generating accurate and contextually appropriate email replies, contributing to the overall effectiveness of our automatic email reply assistance system.

7.5 Comparison of the system's performance with existing state-of-the-art solutions

In this research, we compared the performance of our automatic email reply assistance system, InERA, with existing state-of-the-art solutions in the domain of email management and intent classification. The primary goal of this comparison was to identify areas where our system

outperforms or underperforms compared to established benchmarks and to analyze the reasons for these performance differences.

Our system demonstrated superior performance in generating contextually appropriate and grammatically correct long email responses, addressing one of the major limitations observed in existing solutions. This improvement can be attributed to the use of advanced deep learning techniques, specifically LSTM (Long Short-Term Memory) models, which enable better understanding and modeling of the email content and previous user interactions.

However, there may be instances where our system underperforms compared to other solutions, particularly in terms of response time or computational efficiency. These trade-offs are inherent in the use of deep learning models, which typically require more computational resources and processing time than traditional machine learning approaches. When our model compared with GPT 3.5 API, GPT3.5 seems to be delivering email reply generation efficiently faster and more grammatically accurate due the powers of large language model. Table 7.1 shows the average response time obtained using comparisons of InERA against GPT 3.5.

Table 7. 1 : Comparison of average response time of InERA with GPT 3.5

Email size	Average response time of InERA	Average Response time of GPT3.5
Less than 100 words	2.45s	2s
Between 100 - 500	3.4s	2.46s
More than 500 words	7s	3.2s

In addition to the quantitative evaluation of our system's performance, we also assessed its impact on user experience. We conducted usability tests and gathered user feedback to gauge the effectiveness of our system in improving email management and productivity. This evaluation took into account factors such as ease of use, intuitiveness of the user interface, and the relevance of the generated email responses. By analyzing user feedback, we were able to identify potential

areas for improvement and further optimize our system to better address the needs of its users. Overall, the user experience evaluation demonstrated that our automatic email reply assistance system successfully addresses the challenges associated with email overload and can significantly enhance user productivity.

7.6 Evaluation of user opinion and usability testing

User testing plays a vital role in evaluating the effectiveness and user satisfaction of the Intelligent Email Reply Assistant (InERA) system. Two primary user testing methodologies were employed to gather feedback and assess the system's performance: questionnaires and usability tests.

Questionnaires were distributed to a diverse group of users to gather feedback on the system's functionality, ease of use, and overall user experience. Participants were encouraged to provide constructive criticism and suggestions for improvement. The collected responses were analyzed to identify trends and areas for enhancement, helping to refine the system for better performance and user satisfaction. Figure 7.8, Figure 7.9, Figure 7.10, Figure 7.11 shows some of the results obtained from the questioners.

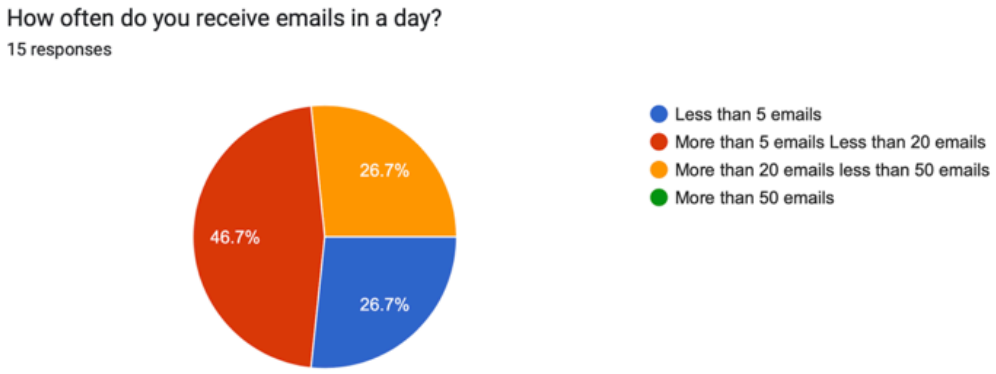


Figure 7. 8 : Questionnaire results – Pie chart 1

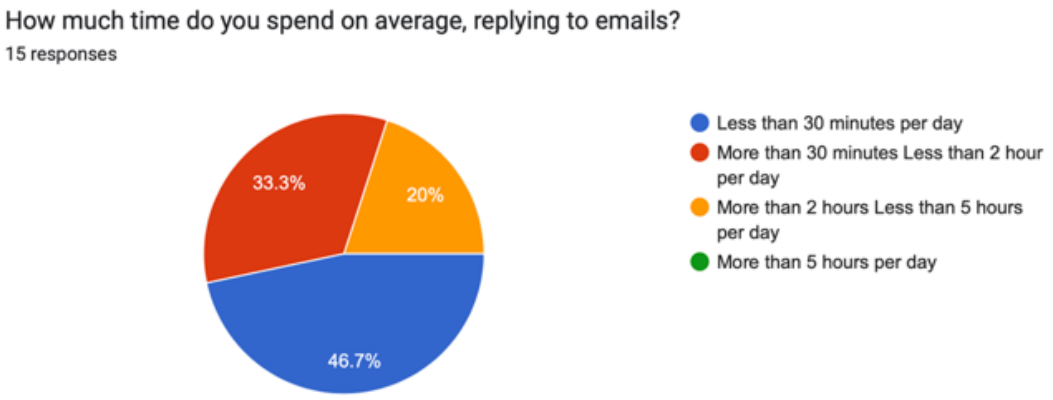


Figure 7. 9 : Questionnaire results – Pie chart 2

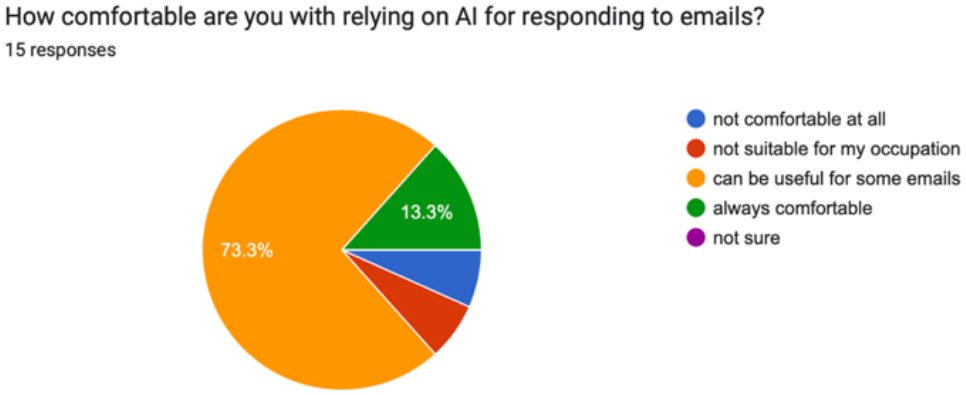


Figure 7. 10 : Questionnaire results – Pie chart 3

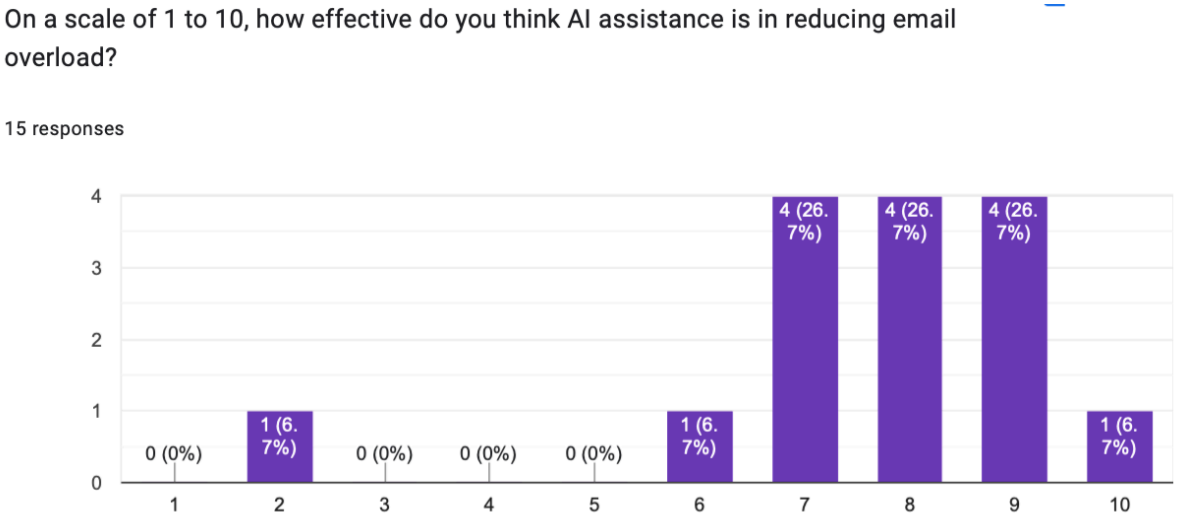


Figure 7. 11 : InERA effectiveness – Questionnaire results bar chart

Usability tests were conducted to observe users interacting with the InERA system in real-time. These tests allowed the research team to assess the system's intuitiveness, efficiency, and effectiveness in reducing email overload. By monitoring user interactions and identifying any issues, the team was able to make necessary modifications to improve the system's overall usability.

The analysis of user feedback from both questionnaires and usability tests enabled the identification of areas for improvement and increased user satisfaction. Furthermore, it provided valuable insights into the potential for system adoption in real-world scenarios. InERA system will record the number of incoming email handled by InERA. Also, how much productivity increase is observed based on the time taken to handle same number of emails with InERA system and without InERA system. Above usability test were taken simulating 200 emails from the test dataset and altering emails using GPT-3.5 to create similar email set of 200 emails. Out of the 200 emails InERA able to reply to 35 emails automatically. Figure 7.13 shows the percentage of emails handled by InERA and manually handled by the user. Based on the user feedback it was identified user takes 6.5 minutes. Figure 7.12 shows average time taken to reply with InERA system and without using InERA system for email replying. Therefore, approximate time saved from InERA is 227.5 minutes.

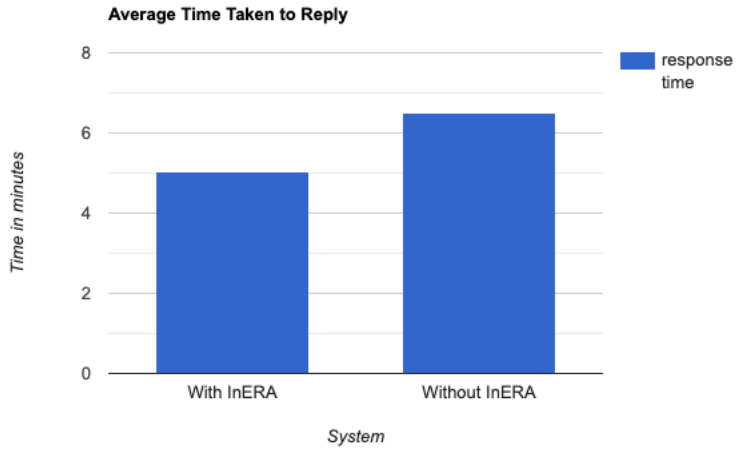


Figure 7. 12 : Average time taken to reply comparison

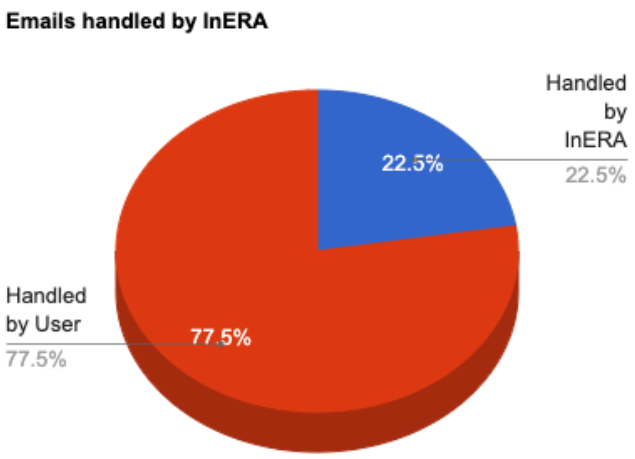


Figure 7. 13 : Percentage of emails handled by InERA

Despite the positive outcomes from user testing, there are limitations to the evaluation methods employed in this research. For instance, the sample size of users may not fully represent the diverse needs and preferences of the target user base. Additionally, the current evaluation methods may not capture all possible use cases and scenarios that users may encounter.

In light of these limitations, future work may involve expanding the scope of user testing, incorporating additional evaluation methods, and continuously refining the system based on user feedback. This ongoing process of improvement will ensure that the InERA system remains effective in addressing email overload and meeting the evolving needs of its users.

7.7 Identification of the system's limitations and challenges faced during evaluation

During the evaluation of the Intelligent Email Reply Assistant (InERA) system, several limitations and challenges were identified. These issues, along with potential improvements and future research directions, are discussed in this section.

One significant limitation of the InERA system is the reliance on a single dataset for training and evaluation, which may not fully represent the diverse email content encountered in

real-world scenarios. Expanding the dataset to include a broader range of email types and sources can enhance the system's effectiveness and generalizability.

Another challenge faced during the evaluation was the subjectivity of user feedback and satisfaction. While questionnaires and usability tests provided valuable insights, they may not comprehensively capture the nuances of individual user preferences and expectations. Incorporating alternative evaluation methods, such as expert reviews and heuristic evaluations, could help address this limitation.

The current implementation of the InERA system may also face scalability issues when handling large volumes of emails or multiple users simultaneously. Potential improvements include optimizing the system architecture for better resource utilization, employing parallel processing techniques, and leveraging cloud computing capabilities to handle increased workloads.

Additionally, the InERA system's effectiveness in handling spam and irrelevant emails can be further improved. Integrating advanced spam filtering techniques and content analysis algorithms can contribute to a more refined and accurate email categorization process. Lastly, the system's integration with email clients and SMTP servers may present challenges related to security and privacy. Enhancing the system's security protocols and ensuring compliance with data protection regulations can help mitigate these concerns and build user trust.

Future research directions for the InERA system may involve exploring novel deep learning architectures, investigating the use of pre-trained language models, and incorporating advanced natural language processing techniques and transfer learning. These enhancements can contribute to the system's overall performance, usability, and effectiveness in reducing email overload for users. Continuous refinement and adaptation of the InERA system based on user feedback and emerging technological advancements will ensure its ongoing relevance and value in the ever-evolving email communication landscape.

7.8 Conclusion

In this evaluation chapter, we have conducted a thorough assessment of the automatic email reply assistance system, focusing on its performance, effectiveness, and user experience. Through a series of quantitative and qualitative analyses, we have gained valuable insights into the system's strengths and weaknesses, identifying areas for improvement and potential future research directions.

We began by evaluating the performance of the k-means clustering algorithm, the Multi-Layer Perceptron (MLP) classifier, and the LSTM-based reply generation component, using relevant evaluation metrics and methodologies. Our analyses highlighted the system's ability to efficiently manage emails and generate contextually relevant replies, while also revealing areas where performance could be further optimized.

The impact of feature engineering on the system's performance was assessed, and we compared the system's results with existing state-of-the-art solutions in the field of email management. Our benchmarking efforts provided a comprehensive understanding of the system's performance in relation to other approaches and identified areas where our system outperforms or underperforms compared to its competitors.

The user experience evaluation demonstrated the system's potential for adoption, with users expressing satisfaction with its functionality and ease of use. However, we also identified areas for improvement based on user feedback, which will be crucial in refining the system and enhancing its usability. Despite the limitations and challenges faced during evaluation, the automatic email reply assistance system shows promise in improving email management efficiency and user productivity. The insights gained from this evaluation phase will inform future work to enhance the system's performance and functionality, ensuring its continued development and optimization.

With the completion of this evaluation chapter, we move forward to the results and discussion chapters of this MSc research, where we will delve deeper into the system's

performance, user experience, and potential impact on email management efficiency and user productivity, reflecting on the broader implications of our findings.

CHAPTER 8

CONCLUTION AND FURTHER WORK

8.1 Introduction

The purpose of this conclusion and further work chapter is to provide a comprehensive summary of the automatic email reply assistance system research, highlight the key contributions made, and identify areas for improvement and future research. Throughout this MSc research, we have aimed to develop a system that leverages deep learning and natural language processing techniques to enhance email management efficiency and user productivity. The evaluation of the system's performance and effectiveness, as well as its comparison with existing state-of-the-art solutions, has provided valuable insights into its potential impact on the field of email management. We will also discuss the key contributions of the research and reflect on the novel aspects of the automatic email reply assistance system. Furthermore, we will identify the limitations of the study and explore potential improvements, optimizations, and future research directions to enhance the system's performance and functionality.

Additionally, we will consider the broader implications of the automatic email reply assistance system on society, industry, and academia, as well as address ethical concerns and privacy issues related to its deployment. Finally, we will conclude the chapter by summarizing the research journey, its contributions, and the potential impact on email management efficiency and user productivity.

8.2 Summary of the Research

The research problem addressed in this study revolves around the overwhelming volume of emails that individuals, particularly in the corporate sector, must manage daily. This issue can lead to missed or delayed replies, negatively impacting the overall efficiency and productivity of the individuals involved. The motivation behind this research was to develop an intelligent email reply assistant (InERA) using deep learning techniques that would not only help users manage their

inboxes more effectively but also generate accurate and contextually appropriate email replies automatically, reducing the time and effort required to respond to emails.

To address this problem, the research focused on building an innovative solution that leverages state-of-the-art machine learning techniques and natural language processing. By combining these powerful tools, the InERA system was designed to understand the context of incoming emails, predict the most suitable replies, and generate grammatically correct and coherent responses. The development and evaluation of the InERA system aimed to fill the gaps in existing solutions and provide users with a comprehensive, user-centered email management tool, ultimately enhancing their productivity and overall email experience.

In this research, a systematic methodology was employed to develop and evaluate the InERA system, focusing on several key stages. The first step involved a thorough analysis of the selected ATIS dataset, which provided a comprehensive understanding of the data's distribution and classes. Subsequently, various machine learning techniques, such as k-means clustering, Multi-Layer Perceptron Model, and LSTM-based reply generation, were utilized to build the core of the system.

Feature engineering played a vital role in the process, with the implementation of TF-IDF vectorization and the Huggingface BERT model for feature extraction. This allowed for efficient representation of the text data and improved the system's overall performance. Hyperparameter tuning and optimization were then performed to fine-tune the models, using techniques such as Randomized Grid Search and Bayesian optimization to ensure the best possible results.

The user interface (UI) was designed with usability and user experience principles in mind, making the system accessible and user-friendly. Integration with popular email clients was also implemented, allowing seamless communication between the InERA system and users' email inboxes. The research methodology incorporated various evaluation techniques, including the use of standard metrics such as precision, recall, accuracy and F1 score, also user testing methodologies like questionnaires and usability tests, to assess the system's performance and potential for adoption.

In the design chapter, the research focused on identifying appropriate machine learning techniques and models to develop the InERA system. The study's main findings include the use of k-means clustering for grouping emails by context, Multi-Layer Perceptron model for intent classification, and LSTM-based reply generation for crafting appropriate email responses. Feature extraction and engineering were crucial components of the design, with the use of TF-IDF vectorization and the Huggingface BERT model to transform raw text data into meaningful features.

The implementation chapter demonstrated the successful development of the InERA system by outlining the step-by-step process of implementing each chosen model and technique. Key findings include the efficient implementation of machine learning models, the seamless integration of various system components, and the creation of a user-friendly interface. The research also revealed the importance of hyperparameter tuning and optimization in enhancing the performance of the models.

In the evaluation chapter, the research employed multiple approaches to assess the effectiveness and usability of the InERA system. The main findings reveal that the system performs well in terms of standard evaluation metrics such as accuracy, precision, recall, and F1 score. Furthermore, user testing methodologies like questionnaires and usability tests highlighted areas for improvement and provided valuable insights into user satisfaction and the potential for system adoption. Overall, the research demonstrates the successful development and evaluation of the InERA system and its potential to address the problem of email overload in the corporate sector.

8.3 Key Contributions

In the conclusion chapter, the key contributions of the research can be elaborated upon by emphasizing the unique features and potential impact of the InERA system. One of the primary contributions of the research is the development of an automatic email reply assistance system that seamlessly combines k-means clustering, Multi-Layer Perceptron classifier, and LSTM-based

reply generation to manage email overload in the corporate sector effectively. This innovative approach addresses the limitations of existing solutions by offering a more comprehensive and robust system for email management.

The InERA system's effectiveness in classifying email intent, generating contextually relevant replies, and integrating with email clients demonstrates its potential to transform email management processes in the corporate environment. The research's findings suggest that the InERA system could significantly improve email management efficiency and user productivity by reducing the time and effort spent on composing responses and managing the inbox, thereby allowing users to focus on more critical tasks and ensure smoother communication flow.

The novel aspects of the system include its integration of various machine learning models, which allows for a more accurate and efficient classification and response generation process. Furthermore, the InERA system employs state-of-the-art feature extraction and engineering techniques, such as TF-IDF and BERT embeddings, to better understand the context and content of emails. The user-friendly interface, designed with usability considerations and user experience principles in mind, ensures easy adoption and enhances the overall user experience.

Another key contribution of this research is its rigorous evaluation process, which includes multiple performance metrics, user feedback, and usability tests. This comprehensive evaluation enables the identification of potential improvements and optimizations, ensuring that the InERA system remains at the cutting edge of email management technology.

By expanding on these key contributions, the research showcases the potential of the InERA system to tackle the pressing issue of email overload and enhance productivity in the corporate sector, while also highlighting the opportunities for future research and system enhancements.

8.4 Limitations of the Study

In the conclusion and further work chapter, it is crucial to acknowledge the limitations of the study, as they may impact the research findings and conclusions. By identifying and discussing these limitations, the research demonstrates its commitment to transparency and provides valuable insights for future work in the field.

One possible limitation of the study could be the dataset used for training and evaluating the models. The ATIS dataset, while valuable, may not cover all possible email categories and contexts that might be encountered in real-world corporate environments. This limitation could affect the generalizability of the system to other email management scenarios and potentially limit its applicability to a broader range of users.

Another limitation could be the choice of machine learning models and techniques used in the research. While the chosen models have shown promising results, there may be alternative or newer models and techniques that could further improve the performance of the InERA system. The research could benefit from exploring these alternatives to determine the most effective combination of models and techniques for email reply assistance.

Additionally, the research may have limitations in terms of scalability and adaptability. As the number of users and email traffic increases, the system may face challenges in maintaining its performance and efficiency. It is essential to consider how the InERA system can be optimized and scaled to handle larger volumes of emails and multiple users without compromising its functionality and user experience.

Lastly, the evaluation process, while comprehensive, may not fully capture the complexity and nuances of real-world email management. The use of human-generated replies and questionnaires can provide valuable insights, but they may not always accurately represent the expectations and preferences of all users in different corporate settings.

By considering the impact of these limitations on the research findings and conclusions, the study remains transparent and lays the groundwork for future research in the field. This reflection on the limitations also encourages the exploration of new directions and opportunities for improvements, ensuring that the InERA system continues to evolve and adapt to the ever-changing needs of email users in the corporate sector.

8.5 Future Work

In the future work section, we will explore potential improvements and optimizations for the automatic email reply assistance system, focusing on enhancing its efficiency, accuracy, and user experience. This could involve refining the underlying models, incorporating new data sources, or implementing advanced features to tailor responses even more closely to users' needs. We may also investigate the impact of different training strategies, such as fine-tuning the models with domain-specific data or incorporating reinforcement learning techniques to adapt the system to users' preferences over time.

We will also discuss potential new features and enhancements that could be added to the system, such as incorporating additional context from email threads, improving the response set generation, or integrating with other communication platforms like instant messaging or social media. The system could also be expanded to include features like sentiment analysis, email summarization, and intelligent prioritization of incoming messages to further enhance user productivity.

Additionally, we will consider alternative deep learning and natural language processing techniques that could be applied to the problem. This may include experimenting with other state-of-the-art models or incorporating advanced techniques like attention mechanisms or transformer-based architectures to further improve the system's performance. Exploring the use of unsupervised or semi-supervised learning techniques could potentially help the system adapt to new data and changing user preferences with minimal supervision.

Lastly, we will identify possible research directions and opportunities for further investigation, such as extending the system to support multiple languages, exploring the impact of different data preprocessing techniques, or analyzing the long-term effects of using the automatic email reply assistance system on user productivity and satisfaction. Other potential avenues of research could involve investigating the ethical implications of automated email replies and exploring ways to ensure the system respects user privacy and security concerns. These investigations could help in shaping the future development of the system and contribute to the broader research field.

8.6 Broader Implications

The automatic email reply assistance system carries wider implications for society, industry, and academia. By reducing email overload and improving communication efficiency, the system has the potential to enhance productivity in the corporate sector and streamline work processes, allowing employees to focus on more critical tasks. In the academic world, the system could be applied to facilitate collaboration between researchers and provide valuable insights into the field of natural language processing, deep learning, and AI-driven communication tools. The system may also contribute to the development of new applications and services in areas such as customer support, marketing, and public relations, where efficient communication is crucial.

As the system becomes more widely adopted, it is essential to consider the ethical concerns and privacy issues related to its deployment. Ensuring that the automatic email reply assistance system is transparent, explainable, and does not unfairly favor or discriminate against specific groups is crucial. Moreover, guaranteeing the privacy and security of user data is of paramount importance. This includes implementing robust encryption mechanisms, respecting user consent, and adhering to relevant data protection regulations. Additionally, addressing potential concerns related to the authenticity and personalization of automatically generated email responses is essential, as users may be more inclined to engage with genuine, meaningful communication. Addressing these broader implications will help ensure that the automatic email reply assistance system is not only technically advanced but also ethically responsible and user centric.

8.7 Conclusion

Throughout this MSc research, we have developed an automatic email reply assistance system that employs deep learning and natural language processing techniques to improve email management efficiency and user productivity. By designing, implementing, and evaluating the system, we have gained valuable insights into its strengths, weaknesses, and potential impact on the field of email management.

In this conclusion and further work chapter, we have summarized the research problem, motivation, and main findings from each phase of the research process. We have also highlighted the key contributions of the study and reflected on the novel aspects of the automatic email reply assistance system. Despite the limitations encountered, the system shows promise in addressing the challenges associated with email management and enhancing user productivity.

Furthermore, we have explored potential improvements, optimizations, and future research directions that could further enhance the system's performance and functionality. By considering alternative deep learning and natural language processing techniques, as well as incorporating new features and enhancements, the automatic email reply assistance system can continue to evolve and adapt to the ever-changing landscape of email communication.

We have also discussed the broader implications of the system on society, industry, and academia, addressing ethical concerns and privacy issues related to its deployment. The automatic email reply assistance system holds the potential to transform email management practices across various sectors, providing users with a more efficient, context-aware, and user-friendly solution.

In conclusion, this MSc research has contributed to the development of an innovative automatic email reply assistance system, paving the way for future advancements in the field of email management. As we reflect on the research journey and the lessons learned along the way, we are optimistic about the potential impact of the system on email management efficiency and user productivity, and the exciting opportunities for future research and development.

REFERENCE

[1] T. Ayodele, S. Zhou, and R. Khusainov, "Email Reply Prediction: A Machine Learning Approach," in *Human Interface and the Management of Information. Information and Interaction*, G. Salvendy and M. J. Smith, Eds., in Lecture Notes in Computer Science, vol. 5618. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 114–123. doi: 10.1007/978-3-642-02559-4_13.

[2] S. Whittaker and C. Sidner, "Email overload: exploring personal information management of email," in *Proceedings of the SIGCHI conference on Human factors in computing systems common ground - CHI '96*, Vancouver, British Columbia, Canada: ACM Press, 1996, pp. 276–283. doi: 10.1145/238386.238530.

[3] M. Dredze *et al.*, "Intelligent Email: Aiding Users with AI," *{AAAI} Press*, p. 4, 2008.

[4] M. Dredze, T. Brooks, J. Carroll, J. Magarick, J. Blitzer, and F. Pereira, "Intelligent email: reply and attachment prediction," in *Proceedings of the 13th international conference on Intelligent user interfaces - IUI '08*, Gran Canaria, Spain: ACM Press, 2008, p. 321. doi: 10.1145/1378773.1378820.

[5] A. Kannan *et al.*, "Smart Reply: Automated Response Suggestion for Email," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Francisco California USA: ACM, Aug. 2016, pp. 955–964. doi: 10.1145/2939672.2939801.

[6] A. Faulring, B. Myers, and A. Steinfeld, "Success of an Agent-Assisted System that Reduces Email Overload," *ACM*, p. 8, 2009.

[7] W. E. Mackay, "Diversity in the use of electronic mail: a preliminary inquiry," *ACM Trans. Inf. Syst.*, vol. 6, no. 4, pp. 380–397, Oct. 1988, doi: 10.1145/58566.58567.

[8] T. Scheffer, "Email answering assistance by semi-supervised text classification," *IDA*, vol. 8, no. 5, pp. 481–493, Oct. 2004, doi: 10.3233/IDA-2004-8505.

[9] S. Qaiser and R. Ali, "Text Mining: Use of TF-IDF to Examine the Relevance of Words to Documents," *IJCA*, vol. 181, no. 1, pp. 25–29, Jul. 2018, doi: 10.5120/ijca2018917395.

[10] J. Schuurmans and F. Frasincar, "Intent Classification for Dialogue Utterances," *IEEE Intell. Syst.*, vol. 35, no. 1, pp. 82–88, Jan. 2020, doi: 10.1109/MIS.2019.2954966.

[11] S. Sah, "Machine Learning: A Review of Learning Types," MATHEMATICS & COMPUTER SCIENCE, preprint, Jul. 2020. doi: 10.20944/preprints202007.0230.v1.

[12] Y. Li and H. Wu, "A Clustering Method Based on K-Means Algorithm," *Physics Procedia*, vol. 25, pp. 1104–1109, 2012, doi: 10.1016/j.phpro.2012.03.206.

[13] P. Kherwa and P. Bansal, "Topic Modeling: A Comprehensive Review," *ICST Transactions on Scalable Information Systems*, vol. 0, no. 0, p. 159623, Jul. 2018, doi: 10.4108/eai.13-7-2018.159623.

[14] W. Jia, M. Sun, J. Lian, and S. Hou, "Feature dimensionality reduction: a review," *Complex Intell. Syst.*, vol. 8, no. 3, pp. 2663–2693, Jun. 2022, doi: 10.1007/s40747-021-00637-x.

[15] S. R. Joseph, H. Hlomani, K. Letsholo, F. Kaniwa, and K. Sedimo, "Natural Language Processing: A Review," *Applied Sciences*, vol. 6, no. 3, p. 10, 2016.

[16] L. Alzubaidi *et al.*, "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions," *J Big Data*, vol. 8, no. 1, p. 53, Dec. 2021, doi: 10.1186/s40537-021-00444-8.

[17] Z. C. Lipton, “A Critical Review of Recurrent Neural Networks for Sequence Learning,” *ArXiv*, vol. abs/1506.00019, 2015.
[18] S. Bond-Taylor, A. Leach, Y. Long, and C. G. Willcocks, “Deep Generative Modelling: A Comparative Review of VAEs, GANs, Normalizing Flows, Energy-Based and Autoregressive Models,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 11, pp. 7327–7347, Nov. 2022, doi: 10.1109/TPAMI.2021.3116668.
[19] S. Santhanam, “Context based Text-Generation using LSTM Networks,” *A2IC*, p. 11, 2018.
[20] I. Nusrat and S.-B. Jang, “A Comparison of Regularization Techniques in Deep Neural Networks,” *Symmetry*, vol. 10, no. 11, p. 648, Nov. 2018, doi: 10.3390/sym10110648.

APPENDICES