UNIVERSITETI I EVROPËS JUGLINDORE
УНИВЕРЗИТЕТ НА ЈУГОИСТОЧНА ЕВРОПА
SOUTH EAST EUROPEAN UNIVERSITY

South East European University
Faculty of Contemporary Sciences & Technologies

Intelligent Email Routing with LLMs

Prepared By:
Hamit Kamberi

Supervised By:
Prof. Dr. Mennan Selimi

A Master Thesis submitted to the Faculty of Contemporary Sciences and Technologies, SEEU in partial fulfillment of the requirements for the degree of M.Sc. in Software Engineering

Tetovo
March, 2026

Declaration

I declare that this thesis has been composed solely by myself and that it has not been submitted, in whole or in part, in any previous application for a degree. I declare that this work has been completed according to the guidelines established by the faculty and has not been submitted for any other purpose. I have duly acknowledged all the sources from which the ideas and extracts have been taken.

Hamit Kamberi

Abstract

Email classification and routing are essential yet challenging tasks in modern organizational workflows. Manual handling of incoming messages often results in delays, misclassification, and inefficiencies, particularly in multilingual and high-volume environments. This thesis introduces a three-phase intelligent email routing framework that integrates traditional machine learning techniques, semantic similarity models, and state-of-the-art large language models (LLMs) to achieve high accuracy, scalability, and cost-efficiency. In the first phase, a semantic keyword extraction pipeline is implemented using the all-MiniLM-L6-v2 SentenceTransformer model, combined with natural language processing techniques such as tokenization, stopword removal, and similarity-based categorization. This stage identifies and classifies domain-specific terms with high precision, providing a robust foundation for further semantic analysis. The second phase leverages advanced LLMs, including multiple GPT-based models (e.g., GPT-3.5, GPT-4) and LLaMA-based models (e.g., LLaMA-2, fine-tuned variants), to perform deep, context-aware interpretation of email content. These models are tested in both zero-shot and few-shot configurations to evaluate their adaptability to multilingual, multi-label, and ambiguous classification tasks. In the final phase, an automated routing engine applies decision logic, enriched with summarization techniques, to ensure that emails are directed to the correct departments. This stage also integrates human-in-the-loop review for cases with low confidence scores, thereby balancing automation with quality assurance. The system will be benchmarked against traditional rule-based and standalone machine learning classifiers, with performance evaluated using metrics such as accuracy, precision, recall, F1-score, misclassification rate, inference time, and cost per processed email. A detailed cost-performance analysis will compare cloud-based API deployments against self-hosted LLaMA inference setups, highlighting trade-offs between accuracy and operational expenses. By bridging keyword-level similarity analysis with advanced semantic reasoning, this research aims to

provide a robust, multilingual, and adaptive solution to email classification and routing. The contributions are expected to advance both academic understanding and practical implementations of AI-driven communication systems, offering a scalable blueprint for real-world deployment in diverse organizational contexts.

Апстракт

Класификацијата и пренасочувањето на е-пошта се суштински, но предизвикувачки задачи во современите организациски работни процеси. Рачното ракување со дојдовни пораки често резултира со доцнења, погрешна класификација и неефикасности, особено во повеќејазични и високо-обемни средини. Оваа теза воведува интелигентна рамка за пренасочување на е-пошта во три фази, која ги интегрира традиционалните техники на машинско учење, моделите за семантичка сличност и најсовремените јазични модели (LLMs), со цел да се постигне висока точност, скалабилност и исплатливост.

Во првата фаза се имплементира цевковод за семантичко извлекување на клучни зборови користејќи го моделот all-MiniLM-L6-v2 SentenceTransformer, комбиниран со техники за обработка на природен јазик како токенизација, отстранување на стоп-зборови и категоризација врз основа на сличност. Оваа фаза идентификува и класифицира доменски специфични термини со висока прецизност, обезбедувајќи цврста основа за понатамошна семантичка анализа.

Втората фаза ги користи напредните јазични модели, вклучувајќи повеќе GPT-базирани модели (на пр. GPT-3.5, GPT-4) и LLaMA-базирани модели (на пр. LLaMA-2, фино прилагодени варијанти), за длабока и контекстуално чувствителна интерпретација на содржината на е-поштата. Овие модели се тестираат во zero-shot и few-shot конфигурации за да се оцени нивната адаптабилност на повеќејазични, повеќе-етикетни и двосмислени задачи за класификација.

Во финалната фаза, автоматизиран мотор за пренасочување применува одлучувачка логика, збогатена со техники за сумирање, за да обезбеди дека е-поштите се насочуваат кон правилните оддели. Оваа фаза исто така интегрира човечка проверка во случаи со ниски резултати на сигурност, со што се балансира автоматизацијата со обезбедување на квалитет.

Системот ќе се споредува со традиционални класификатори базирани на правила и самостојни модели за машинско учење, а перформансите ќе се евалуираат со користење на метрики како точност, прецизност, recall, F1-score, стапка на погрешна класификација, време на инференција и трошок по обработена е-пошта. Детална анализа на трошок-перформанс ќе ги спореди cloud-базираните API имплементации со самостојно хостирани LLaMA инференциски системи, истакнувајќи ги компромисите помеѓу точност и оперативни трошоци.

Со поврзување на анализата на сличност на ниво на клучни зборови со напредно семантичко резонирање, ова истражување има за цел да обезбеди робустно, повеќејазично и адаптивно решение за класификација и пренасочување на е-пошта. Се очекува придонесите да го унапредат и академското разбирање и практичните имплементации на AI-управувани комуникациски системи, нудејќи скалабилен план за реална примена во различни организациски контексти.

Abstrakt

Klasifikimi dhe drejtimi i email-eve janë detyra thelbësore, por sfiduese, në flukset e punës të organizatave moderne. Menaxhimi manual i mesazheve hyrëse shpesh rezulton në vonesa, klasifikim të gabuar dhe mungesë efikasiteti, veçanërisht në mjedise shumëgjuhëshe dhe me volum të lartë. Ky punim prezanton një kornizë inteligjente për drejtimin e email-eve me tre faza, e cila integron teknikat tradicionale të të mësuarit makinerik, modelet e ngjashmërisë semantike dhe modelet më të avancuara gjuhësore (LLMs), për të arritur saktësi të lartë, shkallëzim dhe efikasitet në kosto.

Në fazën e parë, implementohet një linjë semantike për nxjerrjen e fjalëve kyçe duke përdorur modelin all-MiniLM-L6-v2 SentenceTransformer, të kombinuar me teknika të përpunimit të gjuhës natyrore si tokenizimi, heqja e fjalëve të zakonshme (stopwords) dhe kategorizimi sipas ngjashmërisë. Ky hap identifikon dhe klasifikon termat specifikë të domenit me saktësi të lartë, duke ofruar një bazë solide për analiza semantike të mëtejshme.

Faza e dytë përdor modelet gjuhësore më të avancuara, duke përfshirë disa modele të bazuara në GPT (p.sh. GPT-3.5, GPT-4) dhe modele të bazuara në LLaMA (p.sh. LLaMA-2, variante të përshtatura), për interpretim të thellë dhe të ndjeshëm ndaj kontekstit të përmbajtjes së email-eve. Këto modele testohen si në konfigurim zero-shot ashtu edhe few-shot për të vlerësuar aftësinë e tyre për t'iu përshtatur klasifikimit shumëgjuhësh, multi-label dhe rasteve të paqarta.

Në fazën përfundimtare, një motor automatik drejtimi aplikon logjikë vendimmarrëse, të pasuruar me teknika përmbledhjeje, për të siguruar që email-et të dërgohen në departamentet e duhura. Ky hap gjithashtu integron rishikimin nga njeriu për rastet me vlerësim të ulët të besueshmërisë, duke balancuar kështu automatizimin me sigurimin e cilësisë.

Sistemi do të krahasohet me klasifikues tradicionalë të bazuar në rregulla dhe të mësuar makinerik, me performancë të vlerësuar duke përdorur metrika si saktësia, precizioni, rikujtimi, F1-score, shkalla e klasifikimit të gabuar, koha e përpunimit dhe kostoja për email të procesuar. Një analizë e detajuar e raportit kosto-performancë do të krahasojë shërbimet cloud API me zbatimet e vetë-hostuara të inferencës LLaMA, duke theksuar kompromiset midis saktësisë dhe shpenzimeve operacionale.

Duke lidhur analizën e ngjashmërisë në nivel fjalë-kyçe me arsyetimin semantik të avancuar, ky kërkim synon të ofrojë një zgjidhje të fuqishme, shumëgjuhëshe dhe adaptive për klasifikimin dhe drejtimin e email-eve. Kontributet priten të avancojnë si kuptimin akademik, ashtu edhe zbatimet praktike të sistemeve të komunikimit të drejtuara nga AI, duke ofruar një plan të shkallëzueshëm për përdorim real në kontekste të ndryshme organizative. ...

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction and Motivation

Contents

1.1 Motivation

In today's organizations, email remains one of the main channels for communication with customers and partners. Companies receive thousands of emails daily, which need to be read, understood, and forwarded to the right department. When this process is done manually, it often causes long delays, mistakes in classification, and requires several full-time employees to handle the workload. Studies have shown that reading and forwarding a single email can take several minutes, which, when multiplied by thousands of messages, creates significant inefficiencies.

At the same time, recent progress in artificial intelligence and natural language processing has shown promising results in automating such tasks. Large Language Models (LLMs) such as GPT-4, LLaMA, and Mistral are able to understand complex, multilingual, and even ambiguous messages in ways that older rule-based or classical machine learning methods cannot. This creates a strong motivation to design a new system that combines the strengths of keyword-based methods, semantic similarity models, and LLMs in order to improve email classification and routing.

1.2 Problem Statement

The main problem organizations face is the inefficient and error-prone handling of emails. Manual processing takes too long (on average 2–5 minutes per email), and misclassification rates can reach as high as 30%. This leads to delays in communication, frustration for customers, and unnecessary workloads for employees.

Traditional methods, such as rule-based filters or classical machine learning models, also show limitations. They often fail when an email contains multiple topics (multi-label classification) or when it is written in different languages. In real-world practice, many emails belong to more than one category or need attention from different departments. Current approaches are not flexible enough to deal with this complexity.

The problem can, therefore, be summarized as follows:

- Organizations need an automated email routing system that is faster, more accurate, and scalable.

- The system must handle multi-label classification, multilingual input, and ambiguous content.
- It should also reduce the manual workload while keeping a human review option for sensitive or uncertain cases.

1.3 Research Question and Hypothesis

1.3.1 Main Research Question

The primary objective of this thesis is to investigate the effectiveness of integrating traditional machine learning techniques, NLTK-based keyword processing, and large language models (LLMs) into a unified system for automated email classification and departmental routing.

Research Question (RQ):

> How can a multi-phase classification model that combines traditional machine learning techniques, NLTK-based keyword extraction, and large language models improve email classification accuracy and departmental routing compared to traditional rule-based approaches?

1.3.2 Main Hypothesis

Hypothesis (H1):

> A multi-phase classification model that integrates machine learning methods and NLTK-based keyword extraction with large language models for semantic analysis and summarization will significantly improve the accuracy, scalability, and overall effectiveness of email classification and departmental routing compared to traditional keyword-based or rule-based systems.

1.3.3 Expected Outcomes

The underlying assumption of this hypothesis is that combining structured keyword-based filtering with the deep contextual understanding provided by large language models

enables the system to handle ambiguous, multi-topic, and semantically complex emails more effectively than single-layer approaches.

It is expected that the proposed system will achieve higher classification accuracy across departments, demonstrate improved robustness in complex linguistic scenarios, and maintain acceptable runtime performance for real-world municipal deployment. The empirical evaluation presented in Chapter 4 provides quantitative evidence to assess whether these expectations are validated.

1.4 Methodology

To address the research problem, this thesis proposes a three-phase intelligent email classification and routing framework that integrates traditional natural language processing techniques with large language models.

1.4.1 Phase 1: Keyword Extraction and Preprocessing

The first phase focuses on textual preprocessing and semantic keyword extraction. Natural language processing techniques such as tokenization, stop-word removal, and normalization are applied to structure the raw email content. Embedding models (e.g., `all-MiniLM-L6-v2`) are utilized to capture semantic similarity between messages and predefined domain categories. This stage provides structured linguistic features that serve as the foundation for downstream classification.

1.4.2 Phase 2: Context-Aware Classification with Large Language Models

The second phase introduces deep semantic classification using large language models (LLMs). Advanced transformer-based models such as GPT variants and LLaMA architectures are evaluated for their ability to understand contextual meaning and classify emails into predefined departments. Both zero-shot and prompt-engineered configurations are explored to assess model adaptability across multilingual and multi-topic inputs. Comparative experiments are conducted across multiple LLM architectures to evaluate differences in accuracy, latency, and robustness.

1.4.3 Phase 3: Automated Routing and Summarization

In the final phase, the predicted department is used by a routing engine that forwards the email to the appropriate recipient within the organization. In addition, automatic summarization is applied to generate a concise overview of the message content, enabling faster human review and improved workflow efficiency. This phase connects the classification output with practical deployment requirements.

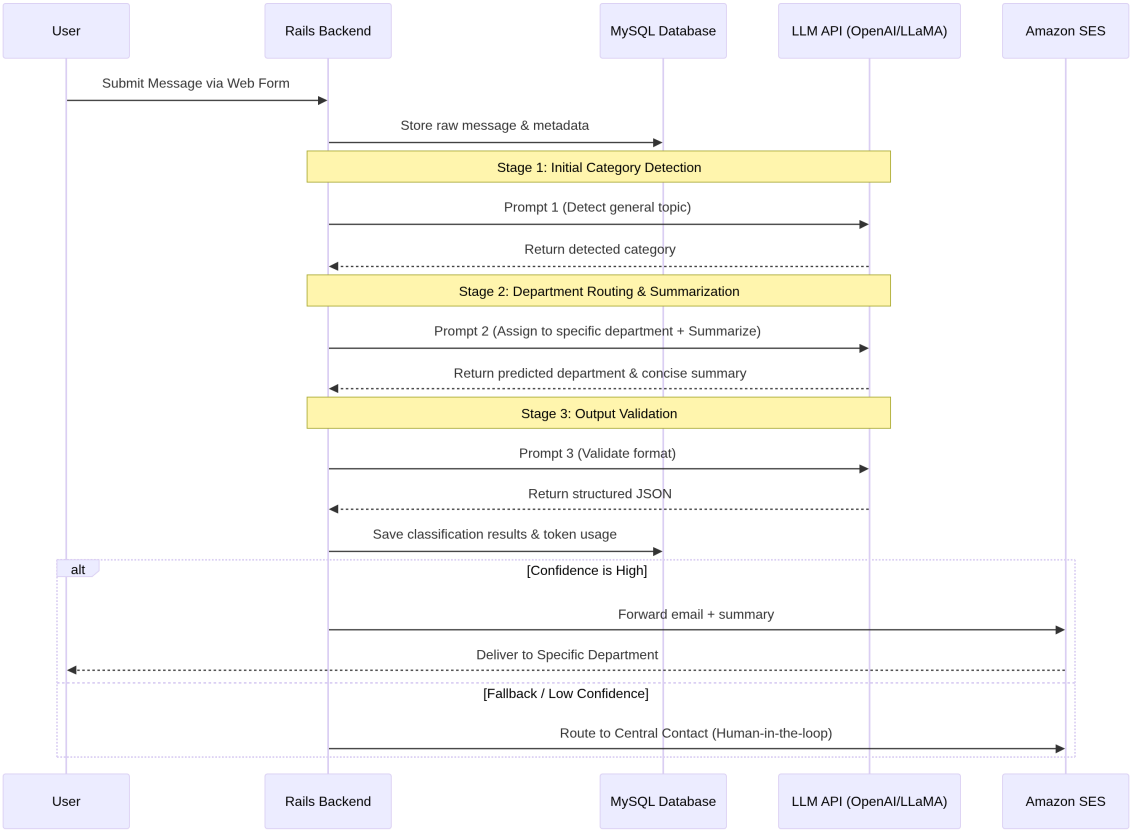

Figure 1.1: The 3-Stage LLM Classification Sequence Diagram

1.4.4 Dataset and Evaluation Strategy

The experimental evaluation is conducted using a dataset of real-world emails collected from municipal and organizational communication contexts. The dataset contains several thousand labeled messages distributed across multiple departments.

System performance is assessed using quantitative evaluation metrics, including classification accuracy, precision, recall, F1-score, misclassification rate, and inference time. In addition, runtime characteristics such as latency and throughput are analyzed to assess

deployment feasibility.

Comparative analysis is performed across different large language models to evaluate trade-offs between classification performance, computational efficiency, and system scalability in both self-hosted and API-based setups.

1.5 Contribution

This thesis makes several important contributions to the field of automated email classification and routing. First, it introduces a new three-phase framework that combines keyword extraction, context-aware classification with large language models, and automated routing with summarization. This design goes beyond existing single-phase or rule-based systems by integrating multiple techniques into one coherent solution.

Second, the thesis benchmarks this framework against traditional approaches such as rule-based systems and standalone machine learning classifiers. The comparison is based on real organizational emails and uses metrics like accuracy, precision, recall, F1-score, misclassification rate, and inference time. In this way, the work demonstrates not only the potential of LLM-driven approaches but also provides a fair reference to established methods.

Another contribution is the cost-performance analysis, which examines the trade-offs between using commercial cloud APIs and self-hosted open-source models. By analyzing both processing efficiency and operational expenses, the thesis offers organizations practical guidance for deployment.

Furthermore, the framework shows scalability and multilingual support. It demonstrates that advanced LLMs can successfully handle multi-label classification tasks, process ambiguous messages, and operate across multiple languages—areas where traditional approaches usually fall short.

Finally, the thesis delivers a practical blueprint for organizations that wish to improve their communication workflows. By reducing manual workload, increasing accuracy, and ensuring reliable message delivery, the proposed system provides a scalable and adaptable solution that can be applied in real-world contexts.

1.6 Report Outline

This thesis is organized into five chapters that progressively present the problem, the theoretical background, the proposed system, and the evaluation results.

- **Chapter 1: Introduction and Motivation**

  This chapter introduces the motivation behind automated email routing systems and explains the problem addressed in this research. It presents the research question, hypothesis, expected outcomes, and the overall methodology used to develop and evaluate the proposed solution.

- **Chapter 2: Background and Related Work**

  This chapter reviews the theoretical foundations required to understand the proposed system. It introduces large language models, transformer architectures, prompt engineering, text classification techniques, and multilingual challenges. In addition, the chapter analyzes existing research on automated email classification and summarizes related work in the field.

- **Chapter 3: System Model**

  This chapter describes the architecture and design of the proposed email routing system. It explains the system components, including the frontend interface, backend processing layer, semantic keyword extraction module, LLM-based classification pipeline, email delivery mechanism, and database structure. The chapter also discusses deployment using containerization technologies.

- **Chapter 4: Data Analysis and Notification System**

  This chapter presents the experimental design and evaluation of the proposed system. Multiple large language models are tested using a dataset of real-world emails from several organizational departments.The chapter analyzes classification accuracy, department-level performance, runtime latency, and resource utilization, providing a comparative evaluation of the models.

- **Chapter 5: Conclusion**

  This chapter summarizes the main findings of the thesis and discusses the implications of using large language models for automated email routing. It highlights the

contributions of the proposed system, addresses its limitations, and outlines possible directions for future research and system improvements.

Chapter 2

Background and Related Work

Contents

2.0.1 Large Language Models

Large Language Models (LLMs) are neural network models trained on large-scale text corpora to understand and generate natural language. Modern LLMs are primarily based on the Transformer architecture introduced by Vaswani et al. [1]. These models learn contextual representations of language through large-scale pretraining and can perform a wide variety of natural language processing tasks, including translation, summarization, question answering, and text classification [2–4].

Unlike earlier machine learning approaches that required task-specific training, LLMs can generalize to new tasks through prompting and instruction-following mechanisms. This capability enables models to perform tasks such as classification or reasoning with little or no additional training data.

2.0.2 Transformer Architecture

Most modern LLMs are built on the Transformer architecture, which relies on self-attention mechanisms to model relationships between words in a sequence [1]. In contrast to earlier recurrent neural networks, Transformers process tokens in parallel and capture long-range dependencies efficiently.

A typical Transformer consists of stacked layers of self-attention mechanisms and feed-forward neural networks. Each token in the input sequence is represented as an embedding vector combined with positional encoding to preserve word order. During processing, attention mechanisms allow tokens to dynamically interact with each other, enabling the model to identify semantic relationships within the text.

This architecture enables LLMs to scale to billions of parameters and process extremely large datasets, which has significantly improved performance across NLP tasks.

2.0.3 Families of Large Language Models

Over the past few years, several major LLM families have been developed. Among the most influential are the GPT series developed by OpenAI, the LLaMA family developed by Meta AI, and open-source models such as Mixtral and Qwen [5–7].

GPT models use a decoder-only transformer architecture optimized for autoregressive text generation. These models demonstrated that large-scale pretraining can produce strong zero-shot and few-shot capabilities [6]. Later models such as GPT-4 further improved reasoning and contextual understanding.

The LLaMA family focuses on efficiency and open research access. These models demonstrate that carefully curated training datasets and architectural optimizations can produce competitive results even with smaller parameter sizes [7].

Mixture-of-experts architectures, such as Mixtral, introduce sparse activation, where only part of the model is used during inference. This approach improves computational efficiency while maintaining high model capacity [8].

While proprietary models often achieve slightly higher performance, open models have rapidly improved and are widely used for research and local deployment.

2.0.4 Open-Source vs Closed-Source LLMs

Large language models can be broadly categorized as open-source or closed-source systems. Open-source models provide access to model weights and training methods, enabling researchers and organizations to inspect, modify, and deploy the models locally [7]. This transparency allows independent verification of model behavior and reduces dependency on external providers.

In contrast, closed-source models are typically offered through commercial APIs. These systems provide strong performance and integrated infrastructure but limit transparency regarding training data and model architecture.

The choice between these approaches often depends on trade-offs between performance, cost, transparency, and deployment requirements.

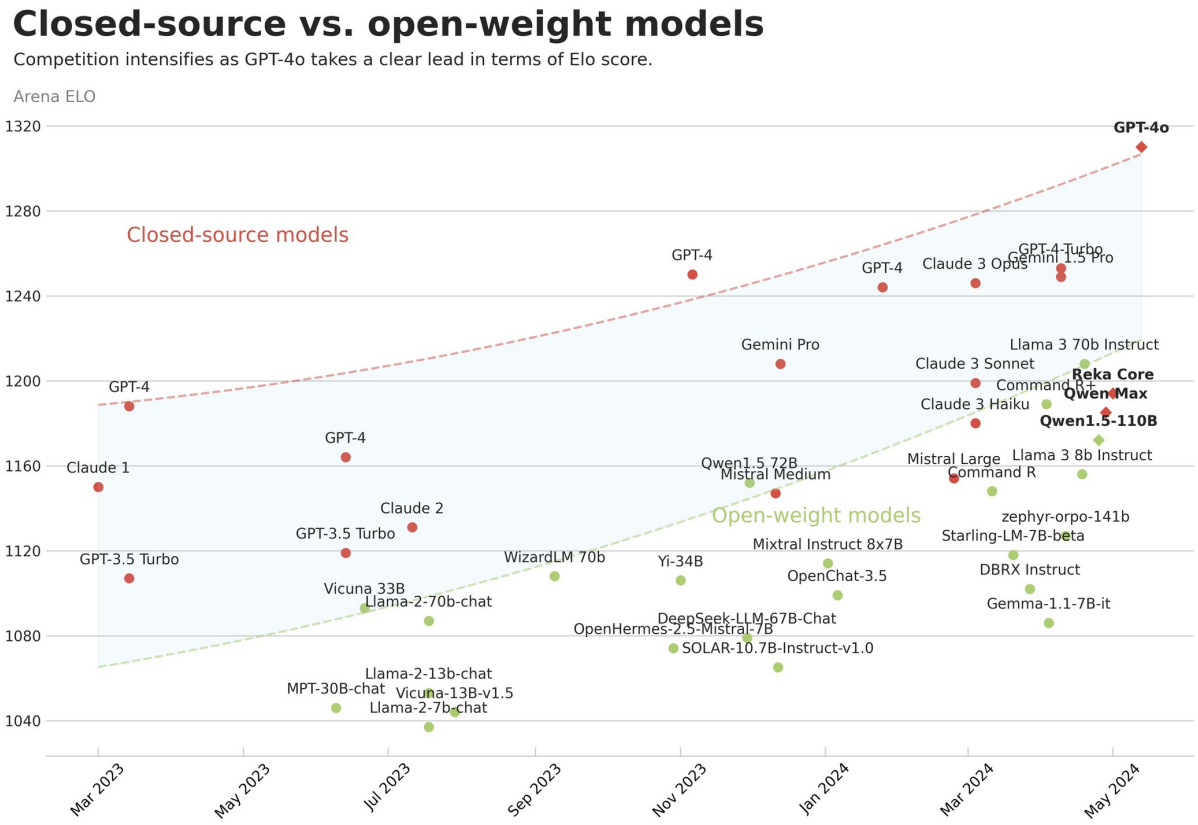

Figure 2.1: Comparison of closed-source and open-weight large language models based on Arena Elo scores.

2.0.5 Text Classification Methods

Text classification is a core task in natural language processing that involves assigning predefined categories to text documents. Early approaches relied on rule-based systems and statistical models such as Naïve Bayes, Support Vector Machines, and logistic regression.

With the introduction of deep learning, transformer-based models such as BERT and RoBERTa have significantly improved classification performance [4]. These models learn contextual embeddings that capture semantic relationships between words.

More recently, large language models have enabled classification without task-specific training. Through prompting techniques, LLMs can classify documents using natural language instructions, which reduces the need for labeled datasets.

2.0.6 Evolution of Email Classification

Email classification systems have evolved significantly over time. Early systems relied on manually defined rules based on keywords or sender information. While easy to implement, these approaches struggled with language variation and required constant maintenance.

Later systems adopted machine learning models trained on labeled datasets. Techniques such as Naïve Bayes and SVM improved accuracy but still relied on static feature representations such as TF-IDF [9].

Recent research has explored deep learning and transformer-based approaches that capture contextual meaning rather than simple keyword matching. These models are better suited for complex scenarios such as multilingual messages, ambiguous queries, and multi-label classification tasks.

2.0.7 Prompt Engineering and Instruction-Based Learning

One of the defining characteristics of modern large language models is their ability to perform tasks through natural language instructions rather than explicit retraining. This paradigm, commonly referred to as prompt engineering, involves designing textual prompts that guide the model toward producing the desired output. Instead of modifying model parameters through supervised learning, prompt engineering leverages the knowledge acquired during large-scale pretraining [4, 6].

Prompt engineering typically involves structuring the input to include instructions, contextual information, examples, and constraints. The prompt defines the task in natural language, and the model generates an output consistent with the provided instructions. For example, in classification tasks, a prompt may instruct the model to assign an email to a predefined category based on its content.

Two commonly used prompting strategies are zero-shot and few-shot prompting. In zero-shot prompting, the model receives only a description of the task without example outputs. The model must rely entirely on its pretrained knowledge to generate an appropriate response. In contrast, few-shot prompting provides several example input–output pairs that demonstrate how the task should be performed. These examples help the model infer

the expected behavior and often improve task performance [6].

Instruction-tuned models further enhance this capability. During instruction tuning, models are trained on large collections of prompts and responses designed to teach the model how to follow natural language instructions effectively. This training process significantly improves task adaptability and enables LLMs to perform a wide variety of tasks without additional fine-tuning [4].

Prompt engineering has therefore become an important technique in modern NLP systems, particularly in situations where labeled datasets are limited. By carefully designing prompts, developers can guide model behavior, enforce structured output formats, and improve classification accuracy without retraining the model.

2.0.8 Multi-Label Text Classification

In many real-world applications, a document may belong to multiple categories simultaneously. This problem is known as multi-label classification and differs from traditional single-label classification, where each document is assigned to only one category.

Email communication frequently represents a multi-label problem. For example, a customer message may simultaneously address billing issues and technical support concerns. In such cases, restricting the classification system to a single label may lead to incorrect routing or delayed responses.

Traditional machine learning approaches often addressed multi-label classification by transforming the task into multiple binary classification problems. Methods such as binary relevance train separate classifiers for each label. Although simple to implement, these approaches ignore relationships between labels and may therefore limit performance [4].

More advanced approaches attempt to capture dependencies between labels. Neural network models can learn shared representations that model relationships between categories, which can improve classification accuracy. In deep learning systems, multi-label classification is typically implemented using sigmoid activation functions and binary cross-entropy loss functions, allowing the model to predict multiple labels simultaneously.

Large language models offer additional flexibility for multi-label classification tasks. In-

stead of predicting labels through fixed output layers, LLMs can generate multiple categories through structured prompts. This approach enables classification systems to dynamically adapt to new categories without retraining the model.

The ability to support multi-label classification is particularly important in automated email routing systems, where messages may contain multiple topics that require attention from different departments [9].

2.0.9 Multilingual Challenges in Email Classification

Another important challenge in automated email classification systems is multilingual communication. Organizations that operate internationally often receive emails written in multiple languages, including English, German, French, and other regional languages.

Traditional machine learning systems typically require separate models for each language. These systems rely heavily on language-specific preprocessing techniques such as tokenization, stemming, and stopword removal. Maintaining separate processing pipelines for multiple languages increases system complexity and reduces scalability.

Transformer-based architectures have significantly improved multilingual language processing. Many modern language models are trained on large multilingual datasets that include text from multiple languages. As a result, the models learn shared semantic representations across languages, allowing them to process multilingual input effectively [4].

Multilingual large language models can therefore analyze messages written in different languages without requiring separate classification pipelines. This capability is particularly useful for automated support systems and communication platforms that interact with users from diverse linguistic backgrounds.

Despite these improvements, multilingual classification still presents challenges. Some languages may be underrepresented in training data, which can reduce model accuracy. Additionally, cultural differences and domain-specific terminology may affect classification performance. Addressing these challenges often requires careful evaluation and domain-specific model adaptation.

2.0.10 Evaluation Metrics for Text Classification

Evaluating the performance of classification systems requires appropriate quantitative metrics. In text classification tasks, several standard evaluation measures are commonly used to assess model performance.

Accuracy is one of the most widely used metrics and represents the proportion of correctly classified instances relative to the total number of instances. However, accuracy alone may not provide a complete assessment of model performance, particularly when datasets are imbalanced and some classes appear more frequently than others [4].

Precision measures the proportion of correctly predicted positive instances among all predicted positive instances. Recall measures the proportion of correctly predicted positive instances among all actual positive instances. These two metrics provide insight into the balance between false positives and false negatives.

The F1-score combines precision and recall into a single metric by calculating their harmonic mean. This measure is particularly useful when both precision and recall are important for evaluating system performance.

In multi-label classification scenarios, additional evaluation methods such as macro-averaged and micro-averaged F1-scores are often used. These metrics provide a more comprehensive assessment of performance across multiple categories.

In the context of email classification systems, evaluation metrics are essential for comparing different approaches, including rule-based systems, traditional machine learning models, and modern transformer-based methods.

2.0.11 Human-in-the-Loop Systems

Although automated classification systems can significantly improve efficiency, fully automated decision-making may introduce risks in sensitive applications. As a result, many real-world systems adopt a human-in-the-loop approach, where automated predictions are supplemented by human review [4].

In human-in-the-loop systems, machine learning models perform the initial classification or routing task. Messages that are classified with high confidence are processed automatically, while uncertain cases are forwarded to human operators for verification. This

approach balances automation with quality assurance and helps prevent incorrect decisions in ambiguous situations.

Confidence scores are commonly used to determine when human intervention is necessary. If the classification model produces a low confidence score, the message can be flagged for manual review. This mechanism ensures that complex or unusual cases receive appropriate attention.

Human-in-the-loop systems are particularly valuable in email routing environments, where incorrect classification could delay responses or route messages to the wrong department. By combining automated analysis with human oversight, organizations can maintain high accuracy while benefiting from the efficiency of AI-based systems [9].

2.0.12 Tokenization and Text Preprocessing

Before natural language text can be processed by machine learning models, it must first be transformed into a structured format that computers can interpret. This transformation is typically performed through a set of preprocessing steps collectively known as *text preprocessing*. One of the most fundamental steps in this pipeline is *tokenization*. Tokenization refers to the process of splitting raw text into smaller units called *tokens*. These tokens may represent words, subwords, or characters depending on the tokenization strategy used.

Tokenization is essential because neural networks cannot directly process raw text. Instead, language models operate on numerical representations of textual elements. Therefore, each token must eventually be converted into a numeric representation that can be interpreted by the model. This step forms the bridge between human-readable language and machine-interpretable numerical data.

Traditional natural language processing systems commonly used whitespace-based tokenization, where sentences are divided into individual words using spaces and punctuation. Although this approach works well for many simple tasks, it introduces several limitations. Natural language contains morphological variations, compound words, and rare vocabulary that may not appear frequently in training datasets. As a result, word-based tokenization can lead to extremely large vocabularies and difficulties in handling previously unseen words.

To address these challenges, modern transformer-based models employ *subword tokenization*. Instead of representing each word as a single token, subword tokenization breaks words into smaller reusable units. Popular algorithms include Byte Pair Encoding (BPE), WordPiece, and SentencePiece. These methods construct a vocabulary of frequently occurring subword units that can be combined to represent a large number of words. For example, a word such as *classification* may be decomposed into smaller components such as *class*, *##ifica*, and *##tion*. Even if the complete word is not present in the training dataset, the model can still infer meaning from its subword components [1, 10].

Another important aspect of text preprocessing involves normalization operations. These operations aim to reduce noise and improve the consistency of textual input. Typical normalization techniques may include converting text to lowercase, removing punctuation, filtering stop words, or applying stemming and lemmatization. While traditional machine learning pipelines relied heavily on such preprocessing operations, modern large language models often require minimal preprocessing because they are trained on massive and diverse text corpora that already contain natural language variations.

Once tokenization is completed, each token is mapped to a numeric identifier within the model vocabulary. These identifiers are then transformed into dense vector representations known as embeddings. Embeddings capture semantic and contextual relationships between tokens and enable neural networks to learn patterns in language. This transformation from raw text to vector representations forms the foundation of most modern natural language processing pipelines.

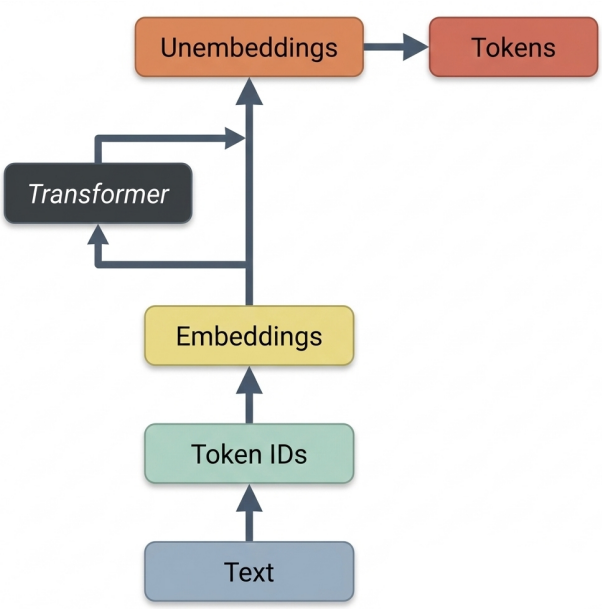

Figure 2.2: Processing pipeline of transformer-based language models from text to output tokens.

Through this preprocessing pipeline, raw textual input is transformed into a structured numerical format that allows machine learning models to analyze and understand natural language effectively. Efficient tokenization methods therefore play a crucial role in enabling modern transformer architectures and large language models to process large volumes of textual data.

2.0.13 Semantic Text Representation

An important development in natural language processing is the shift from traditional frequency-based representations toward semantic vector representations. Early approaches such as Bag-of-Words (BoW) and Term Frequency–Inverse Document Frequency (TF–IDF) represent documents using word occurrence statistics. Although these techniques are computationally efficient, they fail to capture semantic relationships between words and ignore contextual meaning.

To address these limitations, distributed word representations known as word embeddings were introduced. Models such as Word2Vec and GloVe map words into dense vector spaces where semantically similar words are located close to each other. These embeddings allow machine learning models to capture semantic similarity between terms.

Transformer-based models further improved semantic representations by introducing con-

textual embeddings. In these models, the representation of a word depends on its surrounding context within the sentence. As a result, the same word may receive different vector representations depending on its meaning in the specific context [4].

Sentence embedding models extend this idea to larger text units such as sentences or entire documents. Models such as Sentence-BERT and MiniLM generate vector representations that can be used to compute semantic similarity between texts. This capability enables systems to compare messages based on meaning rather than exact word matching.

Semantic representations are particularly useful in email classification tasks. Emails often express similar intents using different vocabulary or phrasing. Embedding-based approaches allow classification systems to detect conceptual similarity even when exact keywords do not match, improving classification accuracy in real-world communication scenarios.

2.0.14 Semantic Similarity and Sentence Embeddings

Semantic similarity refers to the task of determining how closely two pieces of text are related in meaning. In many natural language processing applications, it is necessary to compare texts based on their semantic content rather than their exact wording.

Sentence embedding models transform sentences into dense numerical vectors that represent their meaning. Once the sentences are converted into vectors, similarity between them can be calculated using distance metrics such as cosine similarity. If two sentences have similar meanings, their embedding vectors will appear close to each other in the vector space.

Transformer-based architectures have significantly improved the quality of sentence embeddings. These models capture contextual relationships between words through self-attention mechanisms, allowing them to represent complex semantic structures within text [11].

In classification systems, semantic similarity techniques can be used to compare user input with predefined category descriptions. By calculating similarity between embeddings, the system can determine which category best matches the content of the message.

This approach is particularly useful in email classification systems where messages may

vary significantly in wording but still represent the same underlying request. Semantic similarity methods allow classification systems to generalize beyond exact keyword matching and improve robustness in real-world applications.

2.0.15 Retrieval-Augmented Generation (RAG)

Retrieval-Augmented Generation (RAG) is an architecture that combines traditional information retrieval techniques with large language models (LLMs) to improve factual accuracy and provide responses grounded in external knowledge sources. Instead of relying solely on the information stored in the model's parameters, RAG systems retrieve relevant documents from a knowledge base and incorporate them into the model's context during generation [12]. This approach allows LLM-based systems to access up-to-date or domain-specific information that may not have been present during the model's training phase.

The concept of RAG was introduced to address one of the fundamental limitations of large language models: their tendency to generate plausible but incorrect information, often referred to as hallucination. By augmenting the generation process with retrieved evidence, the system ensures that responses are grounded in real documents rather than relying purely on probabilistic text generation. The retrieved documents are typically embedded using vector representations and stored in a retrieval index that allows efficient similarity search.

A typical RAG pipeline consists of several stages. First, incoming user input is converted into an embedding representation using a sentence encoder or embedding model. These embedding models capture semantic meaning and enable efficient comparison between pieces of text [13]. The resulting vector representation is then used to query a vector database or search index that contains embeddings of documents from a knowledge base. The retrieval component returns the most relevant documents based on similarity measures such as cosine similarity or dot-product similarity.

After the retrieval step, the selected documents are inserted into the prompt context that is sent to the large language model. The model then generates a response conditioned not only on the original query but also on the retrieved evidence. This hybrid approach combines the strengths of retrieval systems—efficient access to large collections

of documents—with the generative capabilities of transformer-based language models.

RAG architectures can generally be categorized into two types: RAG-Sequence and RAG-Token. In RAG-Sequence models, the same retrieved documents are used for generating the entire output sequence. In contrast, RAG-Token models allow the system to retrieve different documents for each generated token, enabling more dynamic knowledge integration during generation [12]. While the token-level approach offers greater flexibility, it also introduces higher computational complexity.

In practical applications, RAG has been widely adopted in knowledge-intensive natural language processing tasks such as question answering, enterprise knowledge assistants, document summarization, and automated customer support systems. Organizations frequently integrate internal documentation, databases, or communication archives into RAG pipelines in order to provide context-aware and domain-specific responses.

Another significant advantage of RAG systems is their ability to update knowledge without retraining the underlying language model. Instead of retraining the model when new information becomes available, developers can update the external knowledge base used by the retrieval system. This significantly reduces computational costs and allows systems to remain up-to-date in dynamic environments.

However, RAG systems also introduce certain challenges. The quality of generated responses strongly depends on the performance of the retrieval component. If relevant documents are not retrieved, the language model may still generate incorrect or incomplete responses. Additionally, maintaining large retrieval indexes and ensuring efficient search at scale requires specialized infrastructure such as vector databases.

2.0.16 Vector Databases and Embedding-Based Retrieval

Vector databases have become an essential component in modern artificial intelligence systems that rely on semantic search and retrieval-based architectures. Unlike traditional relational databases that store structured data in rows and columns, vector databases store high-dimensional numerical representations known as embeddings. These embeddings capture the semantic meaning of data and allow systems to perform similarity-based retrieval.

Embeddings are generated using machine learning models that transform input data into

dense vector representations. In natural language processing, transformer-based models are commonly used to generate embeddings representing words, sentences, or entire documents. Texts with similar meanings produce embeddings that are located close to each other in vector space [13]. This property enables semantic search by comparing distances between vectors rather than relying only on keyword matching.

To support efficient similarity search in large datasets, vector databases use specialized indexing structures known as Approximate Nearest Neighbor (ANN) algorithms. These algorithms allow systems to quickly identify vectors that are closest to a given query vector even when the database contains millions of embeddings. Popular ANN techniques include Hierarchical Navigable Small World graphs (HNSW) and other optimized similarity search structures [14].

Another widely used framework for large-scale vector similarity search is FAISS (Facebook AI Similarity Search), which enables efficient retrieval of embeddings using optimized indexing methods and GPU acceleration [15]. Systems such as FAISS allow developers to perform large-scale semantic search across massive collections of embeddings with high performance.

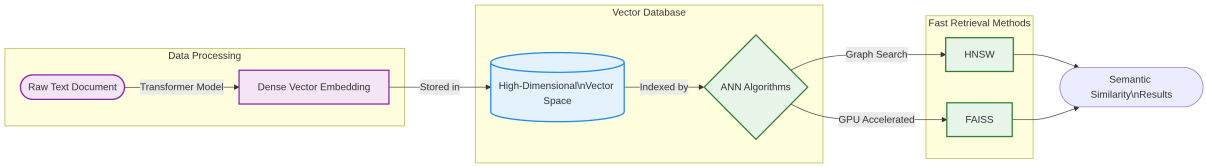

Figure 2.3: Architecture of an Embedding-Based Retrieval Pipeline using ANN and FAISS

Vector databases are particularly useful in applications that require semantic understanding of text. For example, in automated email classification or routing systems, embeddings can represent the semantic meaning of incoming messages. When a new email arrives, its embedding can be compared with embeddings of previously categorized messages or predefined departmental descriptions. Based on the similarity score, the system can determine the most appropriate destination for the email.

In addition to classification tasks, vector databases are widely used in recommendation systems, document retrieval platforms, question answering systems, and conversational AI applications. These systems often combine vector databases with large language models to build advanced retrieval-augmented pipelines that integrate semantic search with natural language generation.

Despite their advantages, vector databases also introduce several technical considerations. Embedding dimensionality affects storage requirements and query performance, while maintaining vector indexes requires efficient update strategies when new data is added. As datasets grow larger, distributed vector database architectures become necessary to maintain scalability and low latency.

Nevertheless, vector databases have become a fundamental building block for modern AI systems that rely on semantic representations and large-scale information retrieval. Their integration with embedding models and large language models enables powerful new capabilities in natural language processing and intelligent automation.

2.0.17 Cost and Computational Considerations of LLM Systems

Although large language models provide powerful capabilities, their deployment introduces important computational and economic considerations. Modern LLMs contain billions of parameters and require substantial computational resources during both training and inference.

Training large models typically requires high-performance computing infrastructure such as GPU clusters or specialized hardware accelerators. However, in most practical applications organizations rely on pretrained models rather than training models from scratch. Pretrained models can be adapted to specific tasks through prompting or fine-tuning techniques [4, 6].

During inference, the computational cost depends on the size of the model and the number of tokens processed. Large models such as GPT-based architectures can process complex language tasks but may introduce higher latency and operational costs.

Two main deployment strategies are commonly used. The first approach involves accessing proprietary models through cloud-based APIs provided by commercial vendors. These services offer highly optimized models and infrastructure but typically charge based on token usage.

The second approach involves deploying open-source models locally. Open-source models allow organizations to maintain full control over their data and infrastructure, although they require dedicated hardware resources and maintenance efforts [6].

In large-scale systems such as automated email routing platforms, balancing model accuracy with computational cost becomes an important design consideration. Organizations must evaluate trade-offs between performance, operational cost, and system scalability.

2.0.18 Email Routing Systems in Organizations

Email routing systems are widely used in organizations to manage large volumes of incoming communication. These systems analyze incoming messages and automatically direct them to the appropriate department or service unit.

Traditional routing systems relied primarily on rule-based mechanisms. Administrators manually defined rules based on sender addresses, subject line keywords, or predefined patterns. While these systems were simple to implement, they were often brittle and required continuous maintenance as communication patterns changed.

Machine learning techniques later improved routing accuracy by learning patterns from labeled datasets. Statistical classifiers such as Naïve Bayes and Support Vector Machines have been used to categorize messages and support requests based on their textual content [9].

More recently, deep learning and transformer-based models have significantly improved the ability of systems to interpret natural language. Large language models can analyze complex queries, extract intent, and summarize messages before routing them to the correct department.

As digital communication continues to grow, intelligent routing systems are becoming increasingly important for organizations seeking to maintain efficient customer support and communication workflows.

2.0.19 Challenges in Automated Email Classification

Despite significant advances in natural language processing, automated email classification remains a challenging problem. Several factors contribute to the complexity of accurately interpreting and categorizing email messages.

One major challenge is linguistic ambiguity. Emails are often written in informal language and may include abbreviations, spelling errors, or incomplete sentences. These

characteristics make it difficult for classification systems to accurately interpret the user's intent.

Another challenge arises from multi-topic messages. A single email may address multiple issues simultaneously, such as billing inquiries combined with technical support questions. This scenario requires classification systems to support multi-label predictions rather than assigning a single category.

Data imbalance also presents difficulties in many classification tasks. Some categories may appear far more frequently than others, causing models to become biased toward majority classes. Addressing this issue often requires specialized training strategies or dataset balancing techniques.

Finally, domain variability presents additional challenges. Email classification systems trained on data from one organization may not generalize well to another organization due to differences in terminology and communication styles [4, 9].

Addressing these challenges requires classification frameworks that combine semantic understanding, contextual reasoning, and robust evaluation methodologies.

2.1 Related Work

Automatic email processing has been widely studied in recent years, encompassing tasks like spam/phishing detection, content summarization, and automated assistance. A number of approaches leverage modern natural language models to perform these tasks with high accuracy. For instance, Rojas-Galeano [16] demonstrated that a large pre-trained language model can classify spam emails in a zero-shot manner: by simply prompting the model with email content, the LLM successfully distinguished spam from legitimate messages without any task-specific fine-tuning. This work illustrates the intrinsic capability of large language models to understand email content patterns.

A key focus in the literature is on detecting malicious emails. ChatSpamDetector by Koide et al. [17] uses an LLM (e.g. GPT-4) to detect phishing emails. They feed raw email text (including attachments) into a model prompt, and achieve 99.7

Beyond security, LLMs have been applied to email summarization and automation. Sudara et al. (2025) created an email assistant for internships/job offers that uses an LLM to

both summarize email threads and extract structured information (e.g., deadlines, company names) [18]. Their experiments showed the LLM greatly outperformed template-based summarizers, achieving over 60

Attention to model robustness and interpretability appears as well. Sajad (2025) focused on making a small Transformer (DistilBERT) robust and explainable for phishing detection [19]. He applied adversarial training and then used XAI techniques (LIME, SHAP, Integrated Gradients) to extract feature importances. A follow-up LLM (Flan-T5) generated human-readable explanations from these attributions. This pipeline yielded an end-to-end framework that was both accurate and user-friendly. Similarly, Lin et al. (2025) fine-tuned smaller LLMs on explanation-enhanced datasets to improve their performance [20]. They showed that training a model to produce explanations (as labels) can greatly boost its classification accuracy. These studies indicate that pairing LLMs with explanation mechanisms yields more trustworthy email processing systems.

Another line of work incorporates personalization and context via Retrieval-Augmented Generation (RAG). Al Barwani et al. (2026) integrated user-specific email context and real-time threat intelligence into the detection process [21]. For each incoming email, their RAG system retrieved relevant historical emails and external data to condition the LLM's decision. This approach dramatically reduced false positives (e.g. lowering the FPR from 12

Large-scale annotated datasets have also been developed. T'oth et al. (2025) constructed a comprehensive email dataset with phishing, spam, and legitimate messages, labeled with categories and even emotional/motivational attributes [22]. They used this to benchmark LLM performance on email security tasks. Their methodology of curating real-world emails and systematically labeling them provides a reference for building task-specific corpora. For routing, a similar dataset of internal emails labeled by department or request type would be needed. Existing public sets (like Enron) may be adapted, but tailored annotation is likely required.

Evaluation metrics in these works typically include accuracy, precision/recall, and F1-score. Koide et al. report overall accuracy on phishing detection, while Al Barwani et al. focus on F1 and false-positive rates under their RAG model [21]. Sudara et al. measure time savings in user tasks. For routing, relevant metrics would be classification accuracy

per category and the reduction in manual reassignments or handling time. In general, all works emphasize the practical effectiveness of LLM approaches in measurable terms.

In summary, the cited literature demonstrates that LLMs are highly effective for email classification and assistance tasks. They have been used to detect spam/phishing with very high accuracy, often with added explanation features [17, 23]. They have also been used to summarize and extract email content far better than static templates [18]. Techniques such as adversarial training and RAG provide robustness and personalization [19, 21]. All these systems rely on a combination of large-scale pretraining and clever prompting or fine-tuning.

Our Contributions: Our work builds on these foundations but targets a different problem: automated routing of general support emails using LLMs. Unlike previous studies that focused on binary spam/phishing detection or domain-specific email summarization, we address multi-class routing within an organization. We integrate an LLM to both classify an email's intent (mapping it to one of many possible recipient categories) and to generate a brief explanation of that choice. We train and prompt the model on a custom dataset of real internal emails labeled by department. We also leverage user-context retrieval (in a RAG-like manner) to provide the model with relevant historical information when needed. In contrast to prior work, our system connects email understanding with organizational workflow: it is designed to plug directly into a helpdesk or ticketing pipeline. This end-to-end approach — combining intent classification, contextual retrieval, and explanation — is, to our knowledge, unique in the literature. It offers both high routing accuracy and transparency, fulfilling operational needs that were not the focus of earlier LLM-based email systems.

Chapter 3

System Model

Contents

3.1 Proposed Architecture

The main goal of the system is to automatically receive, understand, classify, and forward user-submitted messages to the correct department within an organization. This is especially useful in public institutions or companies that receive many requests by email. The architecture is built using Ruby on Rails, Large Language Models (LLMs) such as GPT-4 and LLaMA 3, and a cloud-based email service (Amazon SES) to send messages reliably.

The system is modular, meaning each part is responsible for a specific task, and it is fully containerized using Docker, which makes it easier to install, deploy, and maintain. The overall design follows the Model-View-Controller (MVC) pattern, which is a standard structure in web development.

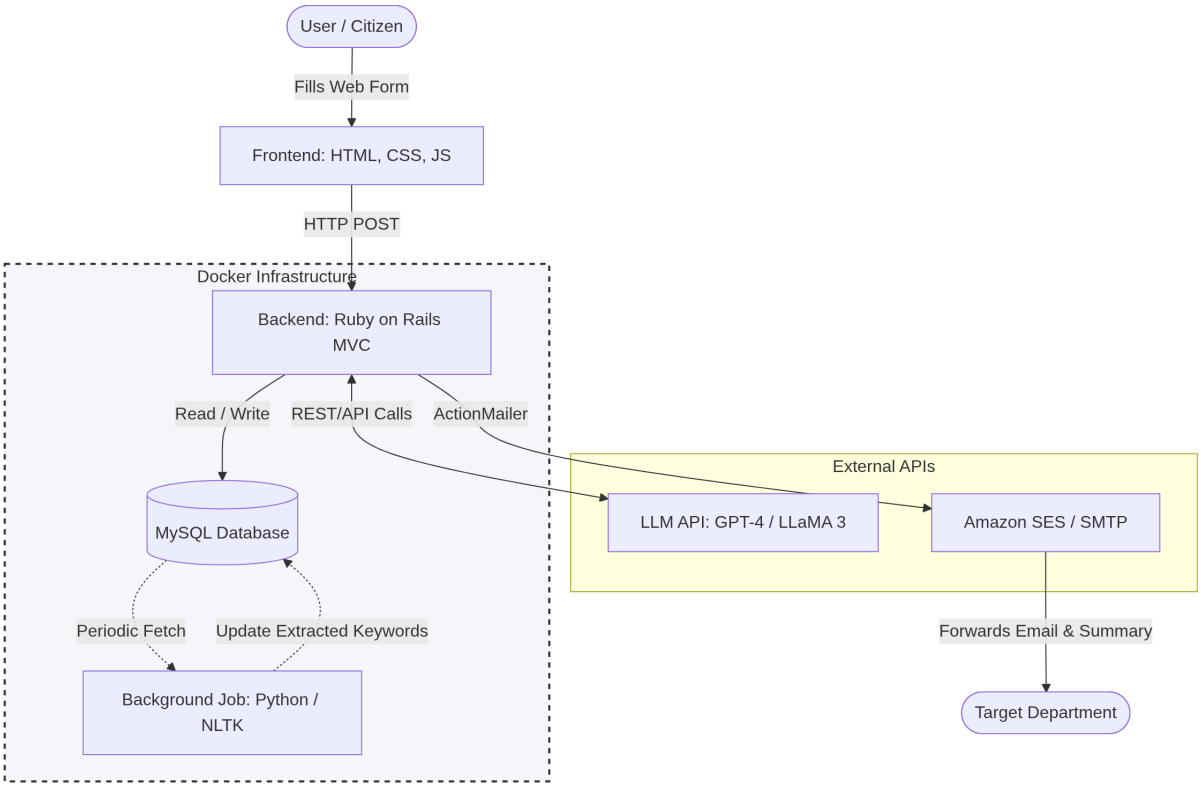

Figure 3.1: Architecture

Technology Stack Overview

3.1.1 System Layers and Components

Layer	**Technology**	**Purpose**
Frontend	Ruby on Rails Views, HTML, CSS, JavaScript	User interaction through a web form.
Backend	Ruby on Rails (MVC)	Handles application logic, LLM integration, and email sending.
Keyword Extraction	Python, NLTK, Sentence-Transformers (all-MiniLM-L6-v2)	Extracts semantic keywords from stored messages and enriches department metadata.
NLP Classification	OpenAI GPT-4 / LLaMA 3	Classifies message category and department and generates message summaries.
Email Delivery	Amazon SES (SMTP)	Sends classified messages to the appropriate departments via email.
Database	MySQL	Stores user input, classification logs, departments, and categories.
Containerization	Docker, Docker Compose	Manages services and ensures reproducible deployment environments.

Table 3.1: Overview of the technology stack used in the proposed architecture

The architecture consists of several logical layers that work together to receive user messages, process them using large language models, and forward them to the appropriate department. Each layer is described below.

3.1.2 User Interface Layer (Frontend)

The frontend layer is responsible for collecting user input and providing a simple interface for submitting messages. It is implemented using Ruby on Rails views together with standard web technologies such as HTML, CSS, and JavaScript.

Users interact with the system through a web form where they can provide the subject and message content, as well as contact information such as name and email address. Additional optional fields include phone number, address, and file attachments. Once the form is submitted, the information is transmitted to the backend for processing and classification.

3.1.3 Application Logic Layer (Backend)

The backend layer manages the core logic of the application and coordinates all system operations. It is implemented using the Ruby on Rails Model–View–Controller (MVC) framework, which organizes the system into controllers, models, and views.

The primary controller responsible for message processing is the `UsersController`. This component receives the form data, stores the submitted message in the database, and initiates the classification pipeline. The backend also handles communication with external services such as the OpenAI API and the email delivery infrastructure.

More specifically, the backend performs the following tasks:

- receiving and validating user input,
- storing messages and metadata in the database,
- invoking the LLM-based classification pipeline,
- identifying the destination department,
- forwarding the message via the email delivery service, and
- recording operational statistics for monitoring and evaluation.

3.1.4 Semantic Keyword Extraction for Departments

In addition to the main LLM-based classification pipeline, the system includes a background job responsible for extracting semantic keywords from previously submitted messages. The purpose of this component is to enrich departmental metadata with keywords derived from real user communication. These keywords improve the semantic context available to the classification system and help identify recurring topics associated with specific departments.

The keyword extraction process is implemented as a Python background job that runs periodically and analyzes messages stored in the system database. The job connects directly to the application database and retrieves the textual content of user messages from the `Users` table. The extracted messages are then processed using natural language processing techniques and semantic similarity models.

The processing pipeline consists of several stages.

First, the system performs text preprocessing using the Natural Language Toolkit (NLTK). Messages are tokenized into individual words, and common German stopwords are removed. This step eliminates frequently occurring but semantically weak words such as articles, conjunctions, and auxiliary verbs. Only alphabetic tokens are retained, and all words are normalized to lowercase to ensure consistent processing.

Next, semantic representations of words are generated using the `SentenceTransformer` model `all-MiniLM-L6-v2`. This model produces dense vector embeddings that capture semantic relationships between words. Unlike simple keyword matching, embeddings allow the system to detect conceptual similarity between terms even if they are not identical.

Each extracted word embedding is compared against embeddings representing known departmental keywords and acronyms. These reference keywords are defined for each department and represent common terms related to that department's responsibilities. For example, the Kids Care department contains terms related to childcare services, while the HR department includes keywords associated with job applications and recruitment processes.

Semantic similarity between words and departmental keywords is calculated using cosine similarity. For each word, the system identifies the department whose keyword embed-

dings produce the highest similarity score. If the similarity exceeds a predefined threshold, the word is assigned to that department as a potential keyword. Otherwise, it is categorized as unrelated.

Over time, this process builds a collection of frequently occurring keywords that appear in user messages. These extracted keywords are combined with the predefined department keywords and stored back in the database. The updated keyword lists are saved in the `Departments` table, ensuring that the system continuously learns new vocabulary used by users.

The overall workflow of the keyword extraction process can be summarized as follows:

- Retrieve messages from the system database.
- Tokenize and filter words using German stopword removal.
- Generate semantic embeddings for each word using SentenceTransformers.
- Compare word embeddings with department keyword embeddings using cosine similarity.
- Assign words to the most similar department if the similarity exceeds a defined threshold.
- Store newly discovered keywords in the database for future classification.

This mechanism allows the system to automatically expand its vocabulary based on real-world user communication. As a result, the classification process becomes more robust to variations in wording, synonyms, and informal expressions used by users. The extracted keywords also provide useful insights into the most common topics addressed to each department, which can support further analysis and system improvement.

3.1.5 NLP Classification Layer

The natural language processing layer performs the semantic analysis of submitted messages. This component integrates large language models such as **OpenAI GPT-4** and **LLaMA 3** to interpret the content of user messages and determine the appropriate organizational department.

After a message is received, the system sends structured prompts to the language model through the OpenAI API. The processing pipeline consists of three main stages. First, the model analyzes the message and determines the most relevant category based on predefined category descriptions. Second, the system generates a short summary of the message to assist human reviewers and to simplify downstream processing. Finally, the model identifies the department responsible for handling the request by comparing the message content with department descriptions and acronyms.

All prompts are dynamically generated and the results returned by the model are stored in the database for auditing and evaluation purposes.

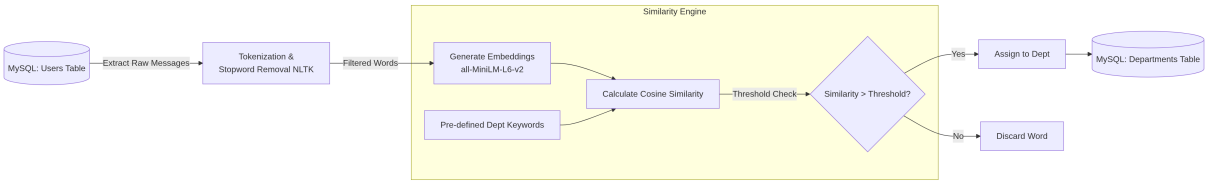

Figure 3.2: Semantic Keyword Extraction Data Flow

3.1.6 Email Delivery Layer

Once the responsible department has been identified, the system forwards the message using **Amazon Simple Email Service (SES)** through the SMTP protocol. Amazon SES provides a scalable and reliable infrastructure for sending transactional emails and ensures high delivery rates.

If a valid departmental email address is found, the system automatically sends the message to that address. In cases where the classification step fails to determine a department, a fallback mechanism is applied. The message is then routed to a predefined central contact address, allowing human operators to review and redirect the request manually.

Email transmission is implemented using the Rails `ActionMailer` framework, which handles email formatting, SMTP communication, and delivery retries.

3.1.7 Database Layer

All persistent data used by the system is stored in a **MySQL** relational database. This layer maintains structured records of user submissions, classification results, and system usage statistics.

Key database tables include:

- `Users` – stores submitted messages and sender information,
- `Departments` – contains department metadata such as names, acronyms, and email addresses,
- `Categories` – defines the classification categories used by the system,
- `OpenaiUsage` – records API usage metrics and token consumption, and
- `ClassificationTest` – stores evaluation results of classification attempts.

Using a relational database provides reliable transactions, structured data organization, and efficient querying for later analysis. The database is integrated with Rails through the ActiveRecord object–relational mapping (ORM) layer.

3.1.8 Containerization and Deployment

To simplify deployment and ensure consistent runtime environments, the system is containerized using **Docker**. Containerization allows the application and all its dependencies to run in isolated environments, avoiding configuration conflicts across development and production systems.

The deployment architecture includes two main containers. The first container runs the Ruby on Rails application, which handles the user interface, classification pipeline, and email routing logic. The second container hosts the MySQL database used for persistent storage.

Both services are orchestrated using a shared `docker-compose.yml` configuration file, which defines networking, environment variables, and service dependencies. This approach enables reproducible deployments, simplified maintenance, and easier scalability of the system.

Chapter 4

Data Analysis and Notification System

Contents

4.1 Data Analysis

4.2 Experimental Design

This section describes the experimental setup used to evaluate the performance of multiple Large Language Models (LLMs) for automated email classification across different municipal departments.

4.2.1 Departments and Dataset Structure

The evaluation was conducted across six distinct departments:

- Betreuung (Kids Care)
- Bewerbung (Jobs – HR)
- Meine Kinder (My Children – Digital App)
- Gruppenhäuser (Group Houses / Youth Camps)
- Kommunikation (Communication / Media Relations)
- Vermietung Sporthallen (Sports Hall Rental – Sport Activities)

Each department contains real-world email samples representing typical citizen inquiries. For every department, the same dataset was used consistently across all evaluated models to ensure fairness and comparability.

All models were tested on identical email inputs within each department. This guarantees that performance differences arise from model capabilities rather than dataset variations.

4.2.2 Evaluated Models

In order to assess the effectiveness of large language models for automated email classification and routing, five open-source LLMs were selected and evaluated. The selected models represent different architectural designs, parameter scales, and optimization strategies. This diversity enables a comprehensive comparison of model capabilities in terms of classification accuracy, computational requirements, and runtime behavior.

All models were executed locally within the same inference environment. Running the models under identical hardware and software conditions ensures that the observed differences in performance originate from the models themselves rather than external factors such as network latency or cloud API variability. By standardizing the execution environment, the evaluation focuses purely on model architecture and inference efficiency.

Table 4.1 summarizes the evaluated models and their key characteristics.

Model	**Parameters**	**Type**	**Notes**
gpt-oss:20b	20B	Open-source	Local inference
llama3.3:70b	70B	Open-source	Local inference
qwen2.5:32b	32B	Open-source	Local inference
mixtral:8x7b	8x7B (MoE)	Open-source	Local inference
apertus:8b	8B	Open-source	Local inference

Table 4.1: Evaluated Large Language Models

The evaluated models differ significantly in parameter size and architectural structure. For instance, LLaMA 3.3 (70B parameters) represents a very large dense transformer model designed for high-level reasoning and language understanding tasks. In contrast, smaller models such as Apertus (8B) prioritize computational efficiency and faster inference.

The Mixtral model follows a mixture-of-experts (MoE) architecture, which activates only a subset of model parameters during inference. This design aims to provide the reasoning capabilities of large models while maintaining lower computational cost. Meanwhile, Qwen and GPT-OSS represent intermediate model sizes that attempt to balance performance and efficiency.

By evaluating models that span a wide range of parameter sizes and architectural approaches, this study enables a systematic analysis of how model complexity affects classification performance and runtime behavior.

4.2.3 Prompt and Processing Pipeline

To ensure a fair and controlled comparison between models, all evaluations were conducted using an identical prompt structure and classification workflow. Each model received the same instructions, input format, and output constraints. No model-specific prompt

engineering or fine-tuning was performed during the experiments. This approach ensures that the results reflect the intrinsic capabilities of each model rather than optimizations tailored to a particular architecture.

Every email classification request followed a standardized processing pipeline consisting of three sequential LLM calls. The first step performs an initial category detection in order to determine the general topic of the incoming email. The second step applies a more specific classification task in which the model assigns the email to one of the predefined organizational departments. Finally, a validation step ensures that the output is returned in a structured and machine-readable format. This three-stage pipeline increases the reliability of the classification process while maintaining consistent behavior across all evaluated models.

4.2.4 Evaluation Metrics

Model performance was primarily evaluated using classification accuracy. For each processed email, the predicted department was compared with the ground truth label stored in the dataset. The result of this comparison was recorded in the database using a boolean status field, where a value of `true` indicates a correct classification and a value of `false` indicates an incorrect prediction.

This binary evaluation mechanism allows for a straightforward calculation of overall accuracy by dividing the number of correctly classified emails by the total number of processed samples.

4.2.5 Overall Model Accuracy Comparison

Table 4.2 presents the aggregated classification results for all evaluated models across the entire dataset. Each model processed approximately the same number of email samples, ensuring a balanced comparison.

Model	Total Samples	Correct	Accuracy (%)
qwen2.5:32b	230	176	76.52
gpt-oss:20b	230	175	76.09
llama3.3:70b	228	172	75.44
apertus:8b	230	134	58.26
mixtral:8x7b	230	91	39.57

Table 4.2: Overall classification accuracy of evaluated models

The results indicate that `qwen2.5:32b` achieved the highest overall accuracy, correctly classifying 176 out of 230 emails, which corresponds to an accuracy of 76.52%. The performance of `gpt-oss:20b` and `llama3.3:70b` is very close, both achieving accuracies above 75%. This relatively small difference suggests that intermediate-sized models can achieve competitive performance even when compared with significantly larger architectures.

In contrast, the smaller `apertus:8b` model shows noticeably lower accuracy, achieving 58.26%. The mixture-of-experts model `mixtral:8x7b` performs substantially worse than the other evaluated models, with an accuracy of only 39.57%. This indicates that architectural design alone does not guarantee strong performance in the email classification task.

Overall, the results demonstrate that medium-sized models such as Qwen and GPT-OSS can provide strong classification performance while avoiding the significantly higher computational cost associated with very large models.

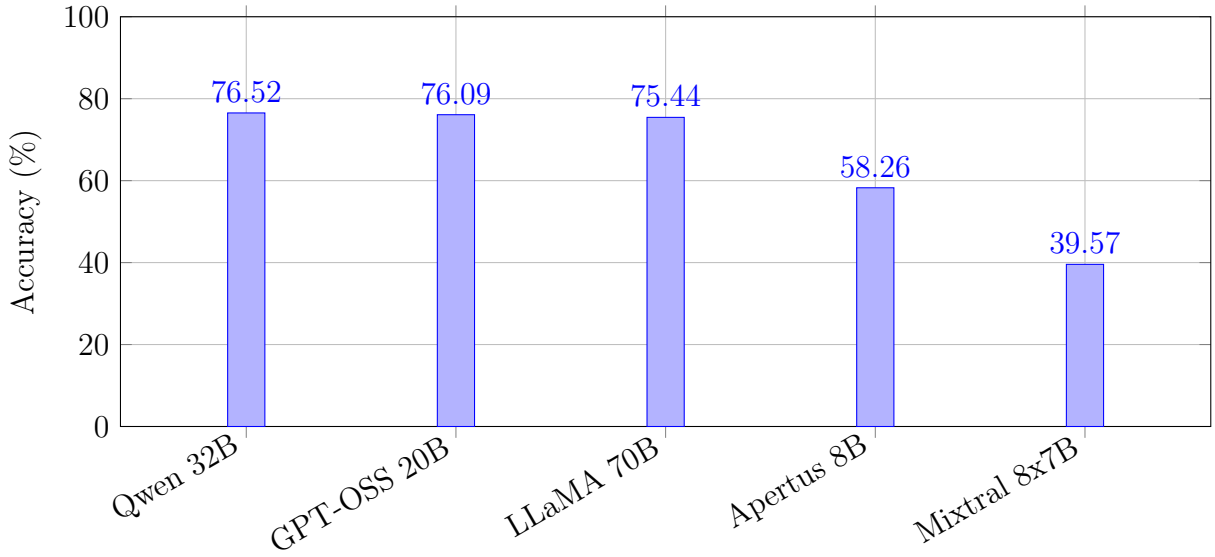

Figure 4.1: Overall accuracy comparison across evaluated models

The results show that qwen2.5:32b achieved the highest overall accuracy (76.52%), closely followed by gpt-oss:20b and llama3.3:70b. The small difference among the top three models indicates comparable classification capabilities.

Notably, parameter size alone does not strictly determine performance, as the 32B Qwen model slightly outperforms the 70B LLaMA model. This suggests that architectural design and optimization are more influential than raw parameter count.

4.2.6 Department-Level Performance Analysis

While the previous section evaluated overall model performance, it is equally important to analyze how classification accuracy varies across different application domains. Email content often differs substantially between departments in terms of language structure, terminology, and contextual complexity. Therefore, model performance may vary depending on the semantic characteristics of the domain.

To assess domain robustness, the average classification accuracy was calculated for each department by aggregating predictions from all evaluated models. Table 4.3 summarizes the number of evaluated samples and the resulting average accuracy values.

Department	**Total Samples**	**Correct**	**Avg. Accuracy (%)**
Jobs – HR	330	260	78.79
Sports Hall Rental	95	71	74.74
Youth Camps	70	47	67.14
Kids Care	404	264	65.35
Communication	129	79	61.24
Digital App	120	27	22.50

Table 4.3: Average classification accuracy per department

The results indicate a substantial variation in classification difficulty between departments. The highest average accuracy is observed for the *Jobs – HR* category, where models achieve nearly 79% accuracy. Emails related to job applications tend to contain structured information such as resumes, application inquiries, and employment-related terminology. These linguistic patterns make them easier for language models to interpret and categorize correctly.

Similarly, the *Sports Hall Rental* domain achieves relatively high accuracy. Requests in this category typically follow predictable phrasing patterns related to booking availability, rental schedules, or facility usage, which simplifies classification.

Moderate accuracy levels are observed in departments such as *Youth Camps*, *Kids Care*, and *Communication*. Emails in these categories often contain broader conversational content and may include overlapping terminology that can appear in multiple departments. This increases semantic ambiguity and reduces classification reliability.

In contrast, the *Digital App* category represents the most challenging classification task, with an average accuracy of only 22.50%. Messages in this domain frequently contain technical issues, short descriptions of problems, or vague references to application features. Such emails often lack clear contextual signals, making it difficult for language models to infer the correct department.

Figure 4.2 provides a model-level comparison of classification accuracy across departments.

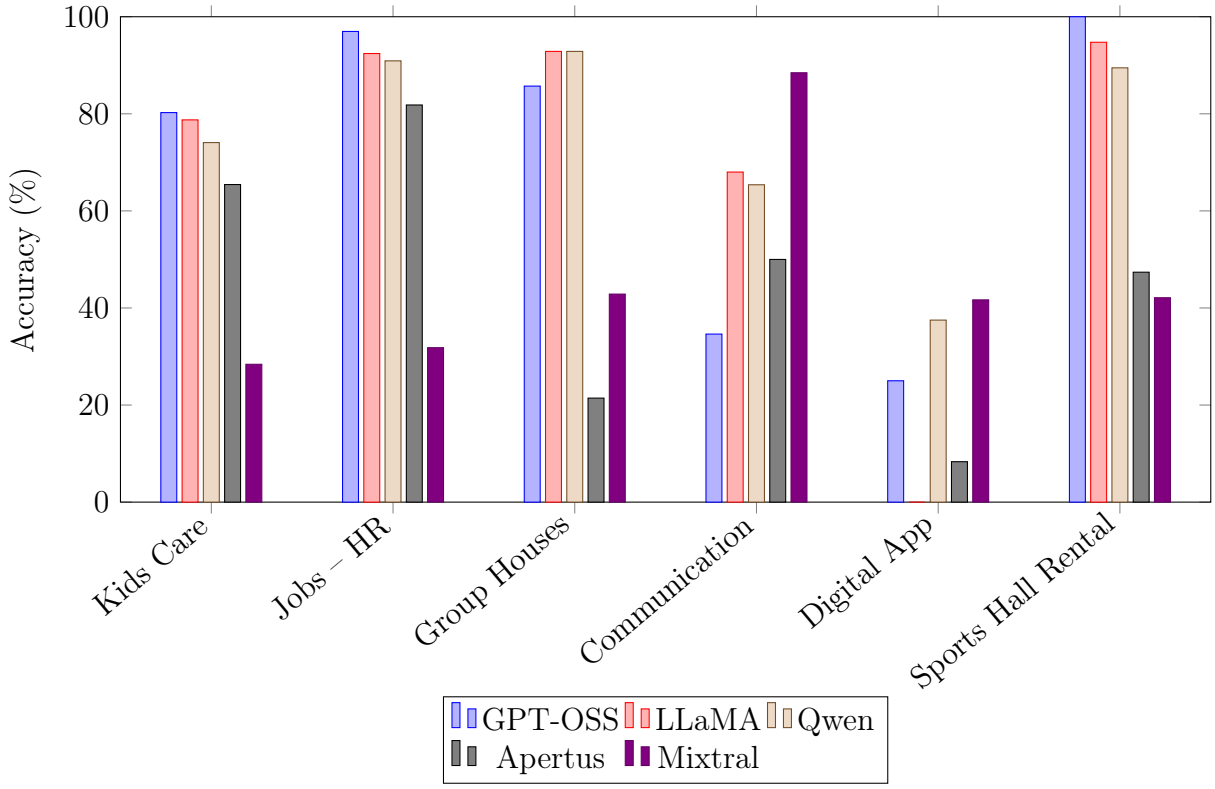

Figure 4.2: Department-level accuracy comparison per model

The figure illustrates that classification performance varies considerably between models depending on the domain. Some models perform consistently across departments, while others show large fluctuations depending on the linguistic characteristics of the input data. This observation highlights the importance of domain-specific evaluation when deploying large language models for real-world routing systems.

4.2.7 Resource Utilization

In addition to runtime performance, resource consumption plays an important role when deploying large language models in production environments. Higher communication overhead can increase infrastructure costs and reduce system scalability.

Table 4.4 presents the average bandwidth usage observed for each evaluated model.

Model	Avg Bandwidth (bytes)
gpt-oss:20b	4057.2
mixtral:8x7b	1492.6
apertus:8b	1419.0
llama3.3:70b	1158.9
qwen2.5:32b	1132.2

Table 4.4: Bandwidth usage comparison

The results indicate that GPT-OSS generates significantly larger response payloads compared to the other evaluated models. Higher bandwidth usage may increase communication overhead, especially in distributed architectures where model inference is performed on remote servers.

Although the differences in bandwidth consumption are smaller than those observed in latency measurements, these values still provide useful insight into the resource requirements associated with each model.

4.2.8 Runtime Performance and Scalability

Beyond classification accuracy, runtime performance is a critical factor for real-world deployment. In practical systems that process large volumes of incoming emails, inference latency directly affects system responsiveness and scalability.

Table 4.5 summarizes the average inference latency and the corresponding standard deviation observed for each evaluated model.

Model	Avg Latency (s)	Std Dev (s)
apertus:8b	1.443	0.705
gpt-oss:20b	5.768	1.600
mixtral:8x7b	15.366	11.992
qwen2.5:32b	20.622	6.155
llama3.3:70b	47.247	9.877

Table 4.5: Latency and stability per model

The results demonstrate a clear relationship between model size and inference latency. Smaller models such as *apertus:8b* produce responses significantly faster than larger architectures. With an average latency of only 1.443 seconds, this model provides the most responsive inference behavior.

In contrast, the largest model in the evaluation, *llama3.3:70b*, requires an average processing time of more than 47 seconds per request. While large models often provide stronger reasoning capabilities, their computational requirements introduce substantial delays during inference.

Figure 4.3 visualizes the relationship between model accuracy and inference latency.

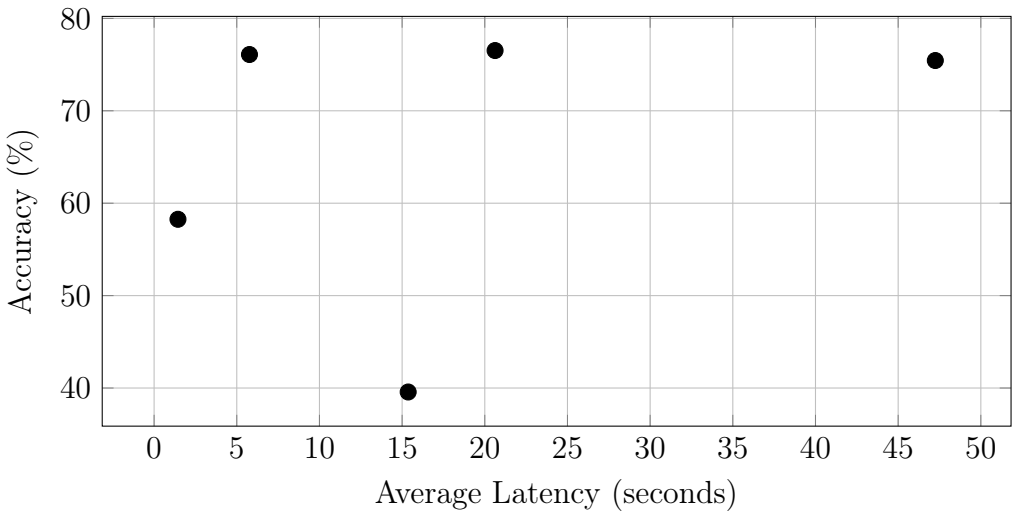

Figure 4.3: Accuracy vs. latency trade-off

The figure highlights the trade-off between classification accuracy and inference speed. While smaller models provide fast responses, their classification performance may be limited. Larger models tend to achieve higher accuracy but introduce significantly higher latency.

4.2.9 Integrated Comparative Ranking

To synthesize the evaluation results, Table 4.6 summarizes all key performance indicators across accuracy, runtime, stability, scalability, and resource utilization.

Model	Accuracy (%)	Avg Latency (s)	Std Dev (s)	Avg Tokens	Bandwidth	Req/s
qwen2.5:32b	76.52	20.622	6.155	753.3	1132.2	0.048
gpt-oss:20b	76.09	5.768	1.600	761.0	4057.2	0.173
llama3.3:70b	75.44	47.247	9.877	739.4	1158.9	0.021
apertus:8b	58.26	1.443	0.705	831.2	1419.0	0.693
mixtral:8x7b	39.57	15.366	11.992	1096.3	1492.6	0.065

Table 4.6: Integrated performance comparison across all evaluation dimensions

4.2.10 Model Selection Considerations

The integrated evaluation demonstrates that no single model dominates across all dimensions.

- **Highest Accuracy:** qwen2.5:32b
- **Lowest Latency and Highest Stability:** apertus:8b
- **Highest Throughput:** apertus:8b
- **Balanced Accuracy–Performance Trade-off:** gpt-oss:20b

While qwen2.5:32b achieves the highest classification accuracy, its latency is significantly higher than mid-sized alternatives. Conversely, apertus:8b provides exceptional runtime efficiency but exhibits lower classification accuracy.

GPT-OSS 20B offers a strong compromise between accuracy (76.09%) and runtime performance (5.768 seconds average latency), making it a suitable candidate for balanced production deployment.

Therefore, model selection should depend on deployment priorities:

- Accuracy-critical applications may favor qwen2.5:32b.
- Real-time or high-volume systems may prioritize apertus:8b.
- Balanced municipal production environments may benefit from gpt-oss:20b.

4.2.11 Multi-Dimensional Performance Visualization

To visualize performance trade-offs across multiple dimensions, Figure 4.4 presents a normalized radar comparison.

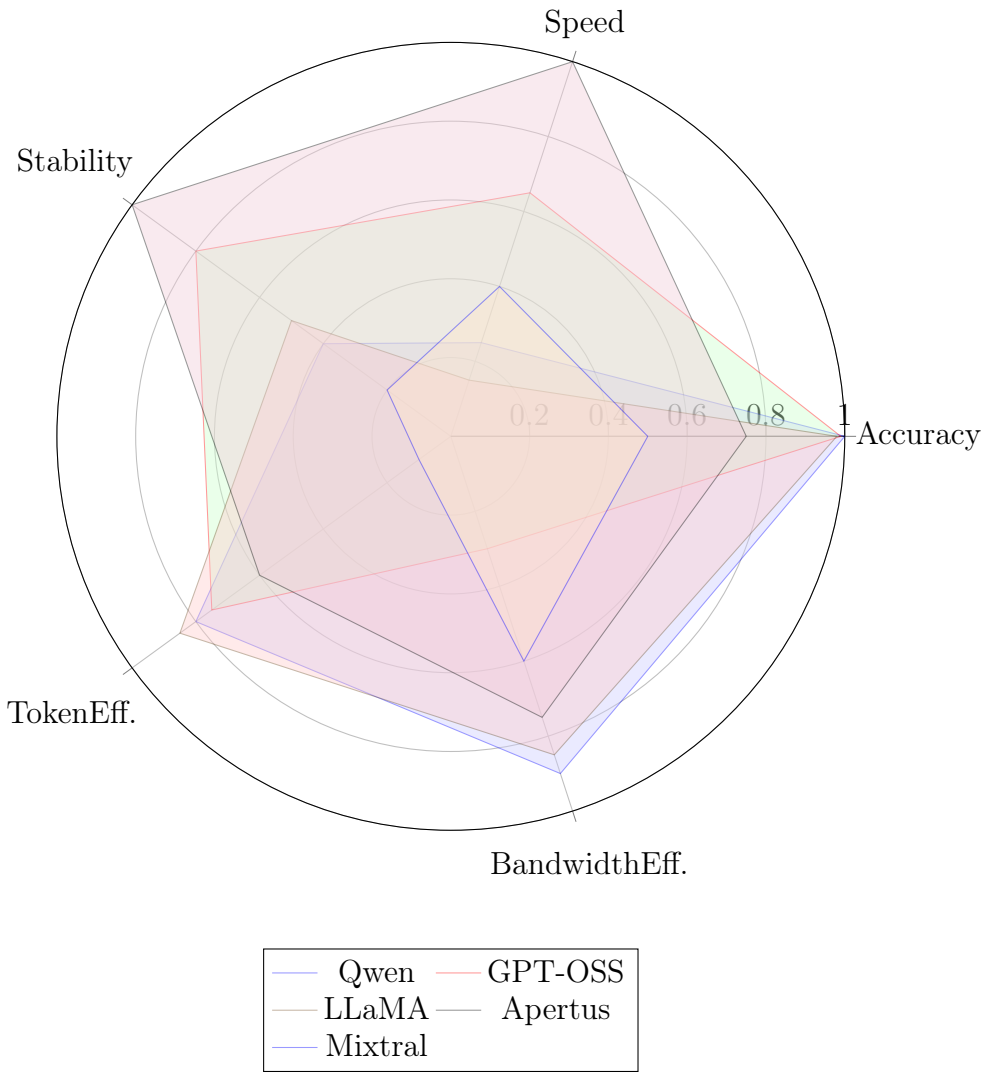

Figure 4.4: Normalized multi-dimensional model comparison

4.2.12 Key Observations

Several important insights emerge from the evaluation:

1. **Larger model size does not guarantee higher accuracy.** The 32B Qwen model slightly outperforms the 70B LLaMA model.

2. **Higher token consumption does not imply better performance.** Mixtral consumes the most tokens yet achieves the lowest accuracy.

3. **The fastest model is not the most accurate.** Apertus achieves superior speed but moderate classification accuracy.

4. **Strong domain variance exists.** The "Meine Kinder" department presents substantially higher difficulty compared to structured domains such as Bewerbung.

These findings highlight the importance of multi-dimensional evaluation when assessing LLM-based classification systems.

4.2.13 Limitations of the Proposed Approach

Despite the promising results obtained in the experimental evaluation, several limitations should be acknowledged. These limitations provide important context for interpreting the results and highlight potential directions for future improvements of the proposed email routing system.

One limitation of this study is the size of the evaluation dataset. Although the experiments were conducted using real email messages distributed across multiple departments, the total number of evaluated samples remains relatively small compared to the scale of real-world organizational email systems. Large institutions typically process thousands of messages per day, and a larger dataset would allow for a more comprehensive evaluation of model generalization and robustness. Consequently, while the reported results demonstrate the feasibility of the proposed approach, they should be interpreted as an initial experimental evaluation rather than a large-scale production benchmark.

Another limitation concerns the absence of domain-specific fine-tuning for the evaluated models. All language models were used in their pre-trained form without additional training on the specific email domain considered in this study. This design decision ensured a fair comparison between models and avoided introducing model-specific bias during evaluation. However, fine-tuning on domain-specific data could potentially improve classification performance, particularly for departments where email content is short, ambiguous, or context-dependent.

Furthermore, the evaluation focused primarily on overall classification accuracy and system-level performance metrics such as latency and resource consumption. While these metrics provide useful insights into system behavior, they do not fully capture all aspects of clas-

sification quality. Additional metrics, such as precision, recall, and confusion matrices, could provide a more detailed understanding of model performance and reveal specific patterns of misclassification between departments.

Another factor that may influence the results is the experimental environment in which the models were executed. All models were run locally under the same inference conditions in order to eliminate network variability and ensure consistent runtime measurements. While this approach allows for fair comparisons between models, it may not fully reflect performance in cloud-based deployments or large-scale distributed inference systems.

Finally, the classification task itself introduces inherent challenges related to the ambiguity of natural language communication. Email messages often contain incomplete information, informal language, or overlapping terminology between departments. As observed in the experimental results, certain domains such as the Digital App category exhibit substantially lower classification accuracy due to the lack of clear contextual indicators. These challenges suggest that future systems may benefit from hybrid approaches that combine large language models with rule-based heuristics or domain-specific knowledge sources.

Overall, while the proposed system demonstrates the feasibility of using large language models for intelligent email routing, addressing these limitations could further improve system reliability, scalability, and real-world applicability.

Chapter 5

Conclusion

This thesis investigated the feasibility of using Large Language Models (LLMs) to automate the classification and routing of organizational emails. The growing volume of electronic communication in modern institutions creates a significant challenge for manual email processing, often resulting in delayed responses, misrouted messages, and increased administrative workload. The primary objective of this research was therefore to design and evaluate an intelligent system capable of automatically analyzing incoming emails and assigning them to the appropriate department.

To address this challenge, a multi-stage email classification framework was proposed. The system combines structured prompt-based reasoning with a controlled inference pipeline consisting of three consecutive LLM calls. The first stage performs an initial semantic analysis of the email content, the second stage determines the most appropriate department based on contextual understanding, and the final stage validates the structured output to ensure consistency and reliability. All evaluated models were executed locally under identical conditions in order to guarantee fair comparison and reproducible results.

Five open-source large language models were evaluated within this framework: GPT-OSS (20B), LLaMA 3.3 (70B), Qwen 2.5 (32B), Mixtral (8x7B), and Apertus (8B). These models differ significantly in parameter size and architectural design, enabling a comprehensive comparison of performance trade-offs between model complexity, classification accuracy, runtime performance, and resource consumption.

The experimental results demonstrate that large language models can effectively perform

email classification tasks with relatively high accuracy. Among the evaluated models, Qwen 2.5 achieved the highest overall accuracy, followed closely by GPT-OSS and LLaMA 3.3. These results indicate that modern open-source language models are capable of capturing contextual signals within email messages and can therefore support intelligent routing systems in organizational environments.

However, the experiments also revealed substantial variability in performance across different departments. Domains with structured and predictable communication patterns, such as job applications or facility booking requests, achieved significantly higher classification accuracy. In contrast, domains involving technical support messages or short problem descriptions proved more difficult for the models to interpret. This observation highlights the importance of domain-specific evaluation when deploying LLM-based classification systems.

In addition to classification performance, the thesis also examined runtime behavior and resource utilization. The results show a clear trade-off between model size and system responsiveness. Smaller models such as Apertus offer significantly lower latency and higher throughput, making them suitable for real-time environments. Larger models, on the other hand, provide stronger reasoning capabilities and slightly higher classification accuracy but require substantially longer inference times. These findings demonstrate that selecting an appropriate model depends strongly on the deployment requirements of the target system.

Overall, the results of this study confirm that large language models represent a viable solution for intelligent email routing. Even without domain-specific fine-tuning, the evaluated models achieved competitive performance when applied to real email messages. The proposed system therefore demonstrates the practical potential of LLM-based automation in administrative workflows and digital communication management.

Despite these promising results, several challenges remain. The experiments were conducted on a relatively limited dataset, and further evaluation on larger and more diverse email corpora would provide a more comprehensive assessment of system robustness. Additionally, future work could explore domain-specific fine-tuning, hybrid rule-based and LLM approaches, and multilingual classification scenarios to further improve system performance.

In conclusion, this research demonstrates that large language models can significantly enhance automated email routing systems by providing contextual understanding of natural language communication. As LLM technologies continue to evolve, their integration into organizational information systems has the potential to reduce manual workload, improve response efficiency, and support more intelligent digital communication infrastructures.

Bibliography

[1] A. Vaswani *et al.*, "Attention is all you need," in *NeurIPS*, 2017.

[2] AWS Blog, "What is a large language model?." `https://aws.amazon.com/what-is/large-language-model/`, 2023. Accessed: 2025-08-13.

[3] Elastic Blog, "Understanding large language models." `https://www.elastic.co/blog/understanding-large-language-models`, 2023. Accessed: 2025-08-13.

[4] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, *et al.*, "A survey of large language models," *arXiv preprint arXiv:2303.18223*, 2023.

[5] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," 2018.

[6] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et al.*, "Language models are few-shot learners," *arXiv preprint arXiv:2005.14165*, 2020.

[7] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, "Llama: Open and efficient foundation language models," tech. rep., Meta AI, 2023.

[8] "Mixtral 8×7b." `https://huggingface.co/mixtral/8x7b`. Accessed: 2026-02-11.

[9] I. Alsmadi and I. Alhami, "Clustering and classification of email contents," *Journal of King Saud University - Computer and Information Sciences*, vol. 27, no. 1, pp. 46–57, 2015.

[10] W. X. Zhao *et al.*, "A survey of large language models," *arXiv preprint arXiv:2303.18223*, 2023.

[11] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[12] P. Lewis, E. Perez, A. Piktus, *et al.*, "Retrieval-augmented generation for knowledge-intensive nlp tasks," *Advances in Neural Information Processing Systems*, 2020.

[13] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, 2019.

[14] Y. A. Malkov and D. A. Yashunin, "Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.

[15] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with gpus," *IEEE Transactions on Big Data*, 2019.

[16] S. Rojas-Galeano, "Zero-shot spam email classification using pre-trained large language models." `https://arxiv.org/abs/2405.15936`, 2024. arXiv preprint. Accessed 2026.

[17] T. Koide, N. Fukushi, H. Nakano, and D. Chiba, "Chatspamdetector: Leveraging large language models for effective phishing email detection." `https://arxiv.org/abs/2402.18093`, 2024. arXiv preprint. Accessed 2026.

[18] T. S. M. Sudara, S. Revathi, V. Muthupriya, and R. Akila, "Ai-powered email summarization and assistant system for internship and job offer management using llms." `https://www.atlantis-press.com/proceedings/icisdiska-25/`, 2025. Advances in Intelligent Systems Research. Accessed 2026.

[19] U. P. Sajad, "A robust and explainable transformer-based framework for phishing email detection." `https://arxiv.org/abs/2511.12085`, 2025. arXiv preprint. Accessed 2026.

[20] Z. Lin, Z. Liu, and H. Fan, "Improving phishing email detection performance of small large language models." https://arxiv.org/abs/2505.00034, 2025. arXiv preprint. Accessed 2026.

[21] A. H. Al Barwani, A. Amara Korba, and R. W. Anwar, "User-centric phishing detection: A rag and llm-based approach." https://arxiv.org/abs/2601.21261, 2026. arXiv preprint. Accessed 2026.

[22] R. Toth, T. Bisztray, and R. A. Dubniczky, "Constructing and benchmarking a labeled email dataset for text-based phishing and spam detection framework." https://arxiv.org/abs/2511.21448, 2025. arXiv preprint. Accessed 2026.

[23] N. Hasan, P. BusiReddyGari, H. Zhao, Y. Ren, J. Xu, and S. Zhang, "Phishing email detection using large language models (llm-pea)." https://arxiv.org/abs/2512.10104, 2025. arXiv preprint. Accessed 2026.