Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

CASEO: Context-Aware Smart Email Organizer with Urgency and Deadline Detection

A. Pallavi Nagalakshmi, E. Agasthya Kumar, M. Harshith Raj
Department of Computer Science and Engineering, Geethanjali College of Engineering and Technology

B. Mamatha
Assistant Professor, Department of Computer Science and Engineering, Geethanjali College of Engineering and Technology

ABSTRACT-- ***The growth of digital communication technologies has led to an increased number of emails, complicating user experience in the process of efficient management, prioritization, and actionable data gathering. The paper discusses CASEO (Context-Aware Smart Email Organizer), a system that helps organize emails and improve overall productivity through context-aware analysis. CASEO proposes the use of a hybrid approach that involves rule-based, NLP, and LLM-based solutions to classify emails as either internships, job-related, academic, event-related, or personal. One of the contributions made by CASEO is a novel method to automatically extract deadlines from emails using a combination of different techniques like pattern recognition, heuristics based on keywords, and LLM-powered inference. The latter helps to handle more difficult and ambiguous texts in order to increase the accuracy rate of the extracted information. Apart from providing accurate extraction of deadlines, CASEO allows for actionable insights, including highlighting emails of significance and integration with other applications like calendars. CASEO uses a state-of-the-art full-stack architecture and is evaluated with real-life emails as input data. It was found that the proposed algorithm provides better efficiency in email management as well as increases the accuracy of email classification and relevant deadline extraction in comparison with regular filtering algorithms. CASEO mitigates limitations in modern email management programs due to the introduction of automation and contextual awareness.***

Index Terms- Email Classification, Context-Aware Computing, Natural Language Processing, Information Extraction, Deadline Detection, Smart Email Systems, Large Language Models.

I. INTRODUCTION

One of the significant changes brought about by digital communication is the fast growth in the number of emails exchanged per day. This increase in traffic has led to information overload and decreased efficiency in terms of identifying and responding to emails efficiently. In today's world, emails have turned into one of the most prominent forms of communications in academia, professionally, and even socially. Moreover, the information transmitted via emails is quite crucial for students, employees, etc., since it contains data related to various deadlines, event information, employment, and other significant news. Despite the usefulness of emails, the need to manually organize them is rather inefficient and can lead to errors.

Current technologies for managing emails include rule-based filters and spam detectors. While such solutions can be helpful for sorting and identifying spam emails, they do not offer advanced capabilities for analyzing complex contexts and extracting relevant information from emails. Since rule-based filters are based on pre-set criteria and keyword analysis, they cannot accurately understand the context of email messages. As a result, crucial emails could be missed and filtered as non-important emails, especially those that contain important information in an unstructured format.

Modern NLP and ML-based solutions have the potential to analyze email content and classify the emails based on the results. However, such tools can rely only on one technique, which reduces the effectiveness of the solution significantly, especially if the emails are formatted differently or include information that cannot be identified directly.

This paper proposes CASEO (Context-Aware Smart Email Organizer), an innovative framework for improving the process of organizing and extracting useful data from emails. The system combines rule-based filters, various NLP algorithms, and LLM support to provide reliable and efficient email management. One of the critical features of the proposed framework is the ability to analyze the context of emails and extract deadlines, which require a certain amount of information.

The primary contributions of this work include the design of a hybrid framework for context-aware email classification, the development of a multi-layered deadline extraction mechanism, and the integration of actionable insights such as prioritization and calendar-ready outputs. By combining multiple intelligent techniques, the system improves the accuracy and efficiency of email organization compared to traditional approaches.

II. LITERATURE REVIEW

The issues of email management and classification have been broadly researched in Natural Language Processing (NLP) and information retrieval literature. Rule-based systems were traditionally used in email applications, such as Outlook. Rule-based systems operate using pre-set rules and keyword matching to filter emails and place them into folders or classify them as spam. Although efficient in detecting simple and repeated patterns, rule-based approaches lack adaptability and have trouble identifying complex patterns.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

Machine learning approaches have substantially improved the performance of email categorization systems. Approaches like Naïve Bayes, Support Vector Machine (SVM), or Decision Trees have been used for email classification based on learning patterns from labelled datasets. Such approaches typically outperform rule-based classifiers but require vast training samples and face difficulties with generalizing for different forms of emails and languages.

With recent developments in Deep Learning, more advanced approaches have been suggested, including Recurrent Neural Network and transformer-based networks, which help recognize contextual relationships between elements in the text. They significantly improve the performance of both classification and sentiment analyses and work well with long and complex emails, especially transformer models. Nevertheless, most classification systems still do not focus on extracting meaningful information from emails, such as deadlines or scheduling.

Information extraction from non-structured text is an area of research with great relevance to our problem. Methods like Named Entity Recognition (NER) and Pattern Based Extraction have been successfully used to detect entities, such as locations, dates, and names. The main drawback of these methods is their inability to recognize implicit data and the variations of the expressions for specific entities.

With the rise of Large Language Models (LLMs), new opportunities have emerged in recognizing patterns and implicitly understanding the text. For instance, LLMs are capable of interpreting complex language relations and can be efficiently used for tasks, such as summarization or intent recognition. However, purely LLM-based approaches might result in higher costs and variable outputs.

Given the aforementioned limitations of existing solutions, there is a need for the system that integrates advantages of several approaches. The proposed system will integrate rule-based and ML approaches with LLM to provide highly accurate email classification along with reliable deadline extraction.

III. PROPOSED SOLUTION

CASEO, which stands for Context-Aware Smart Email Organizer, is described as an advanced system consisting of various elements designed to automate the process of email organization and provide information. The proposed approach is based on the creation of different modules responsible for separate tasks during email processing. Such an approach provides great advantages in managing unstructured email data.

A. User Authentication Module

The User Authentication Module controls the process of accessing the system through user registration and login using valid credentials. It provides authentication techniques to ensure the security of users' data and to establish secure connections. The User Authentication Module acts as the gateway for the system, ensuring that only authenticated users can access the e-mail application features.

B. Email Acquisition Module

The Email Retrieval Module ensures that the messages are retrieved from the user's mailbox through the interface provided by the email provider, such as Gmail API. The module ensures that the core parts of each email are extracted. These core parts include the email subject, body, sender, and timestamp. The collected emails are then saved using a structured database schema. Data fetching and data synchronization techniques are employed by the module to achieve continuous integration of new emails to the system.

C. Email Preprocessing Module

The Email Preprocessing Module prepares raw email data for analysis by removing noise and standardizing the content. This includes eliminating HTML tags, special characters, and unnecessary formatting elements. The text is normalized through operations such as lowercasing and whitespace removal. This preprocessing step enhances the quality of input data and improves the performance of downstream modules such as classification and information extraction.

D. Email Classification Module

The Email Classification Module classifies messages according to various categories like internships, jobs, academics, events, and personal communication. In the design of the email classifier, a mixed methodology approach has been used that involves rule based and NLP based filtering systems. Firstly, rule-based classification takes place for the easy classification of simpler emails by using rules and heuristics. If the email is too complicated for this process, then it is analysed with the help of NLP techniques.

E. Deadline Detection Module

Deadline Detection Module forms an integral part of the system that works towards extracting time-related data from emails. A hierarchical approach has been adopted to achieve reliable extraction of data. Firstly, date-time patterns are extracted using regular expressions. Secondly, heuristics are employed to recognize context-based phrases such as "last date," "apply before," or "deadline." In situations where the context is not explicit, an alternate method is employed whereby the large language model is relied upon to make sense of the deadline in the email message

F. Urgency Detection Module

Urgency Detection Module will rate the significance of an email by considering information extracted from the email as well as clues derived from the context. Emails that have deadlines or urgency-related keywords will be rated as highly significant. Scoring system is employed to categorize an email as either a high-priority, medium-priority, or low-priority email.

G. Seach Module

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

This is a module that can efficiently retrieve emails from the system using the user's query. The Search Module can search for emails using keywords, categories, and even time elements such as date. This module utilizes indexed data structures in order to perform searches effectively

H. Action and Integration Module

Information extracted will now be converted into an action plan through the Action & Integration Module. One of the key functions of the Action & Integration Module is the integration with external applications like Google Calendar. Deadlines and events extracted from the input source will be transformed into scheduled tasks that will be placed directly onto the calendar of the user.

I. Email Dashboard Module

Email Dashboard module acts as an interface for visualization of information that has been processed. In this module, users can see emails that have been classified into different categories, deadlines, and priority levels. Users are able to act on insights gathered from this module and hence improve the usability of the system.

IV. SYSTEM ARCHITECTURE

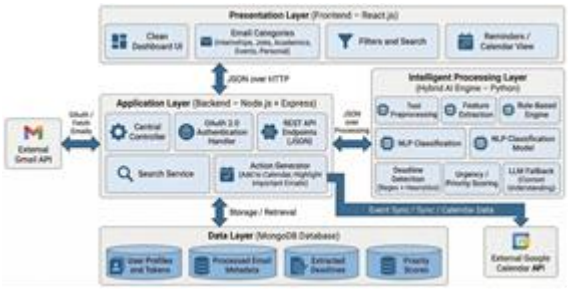

Fig: 1 System Architecture of CASEO

The CASEO architecture is designed as a decoupled, multi-tier microservice system that separates the user interface, business logic, and intelligent data processing. This ensures scalability, security via isolated API handling, and high-performance AI execution.

A. Presentation Layer (Frontend – React.js)

Presentation Layer acts as the user interface of the system and will be developed by employing React.js with Vite to maintain a robust development environment. The layer offers a user-friendly dashboard for effective email management.

Dynamic tagging is employed to display email categories, including academic emails, job emails, internship emails, and event emails. In addition, deadlines are tagged with priority codes, such as critical, high, medium, and low.

Models are provided based on the user's needs. For example, the user may need to view the email content or take an action, such as adding an event to the calendar or retriggering intelligence. This can be done asynchronously using REST API calls over JSON HTTP requests.

B. Application Layer (Backend – Node.js and Express)

The Application Layer plays the role of the central manager of the system and serves as the interface among external services, intelligence layer, and the data layer. It is built with Node.js and Express.

The central controller takes care of the basic business logic like email synchronization, searching, and user preferences. It makes use of an OAuth 2.0 protocol for secure authentication, providing access to the Gmail API and saving encrypted tokens.

There is an action generator module that transforms extracted deadlines into actions and connects those actions with third-party calendar services. There is a Python bridge that helps communicate with the intelligent process. The bridge collects emails and sends them in batches to the processing service via HTTP communications.

C. Intelligent Processing Layer (Hybrid AI Engine - Python)

The Intelligent Processing Layer can be regarded as the main component of analytics within the project. The layer consists of a Python-based server with Flask as its implementation environment that serves to conduct sophisticated NLP analysis.

The pre-processing and feature extraction layer incorporates methods like named entity recognition and context understanding in order to find important information in email texts. Rule-based approach is used to identify specific structure and remove unnecessary elements like numeric entities, which do not relate to deadlines.

Rule-based large language model-based multi-stage pipeline is used in order to enhance accuracy of deadline recognition by combining the two approaches.

D. Data Layer (MongoDB Database)

For the Data Layer, MongoDB will be used, which is a document-oriented NoSQL database and flexible when working with semi-structured emails. It will store user profile data and tokens to provide safe session management. There will be a local cache of parsed emails, which allows retrieving necessary information without making an external call through the API. Besides, intelligent elements like deadlines and priorities will be saved to retrieve results later in case the service for parsing emails is down.

E. Asynchronous Background Processing

In order to increase efficiency and user-friendliness, CASEO uses an asynchronous non-blocking processing method. As soon as an attempt is made to synchronize email accounts, the system will

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

immediately begin processing the newest emails available to it, providing them to the user as quickly as possible.

At the same time, other emails are being processed in the background, using a fire and forget method. Such incremental processing ensures immediate interaction with the system by the user, while the rest of the data set is processed step-by-step.

F. Security and Inter-Service Communication

It features an architectural approach that involves the use of microservices along with well-defined separation among all the components. A gateway approach is adopted whereby only the backend communicates with other external APIs as well as the database to ensure improved security and proper access control. The interaction between the backend and the intelligent processing component takes place through the exchange of JSON messages over HTTP.

V. IMPLEMENTATION

The CASEO application consists of a full-stack software built on contemporary web-based technologies and an artificial intelligence processing pipeline. The development process is modular in nature and is designed to ensure scalability and efficient data transfer among the various modules.

A. Frontend Implementation

The front end has been created with React.js and Vite. This makes the front end highly efficient and very responsive.The dashboards will show emails that are categorized together with urgency levels and deadlines. This will be done through dynamic rendering to make changes in email tags and urgency levels. Context-sensitive modal windows have been created that will enable you to see more details about the emails and also do things like scheduling events on the calendar or reprocessing email intelligence.

The state management system is optimized for efficiency and usability, while the asynchronous API calls have been made using HTTP requests for communication between the frontend and the backend.

B. Backend Implementation

The back end uses Node.js and Express, providing the system's main control unit through RESTful APIs that facilitate communication among the front-end application, external services, and the intelligent processing components.

Authentication uses OAuth 2.0 to provide access to the Gmail API in a secure manner. Access tokens and refresh tokens are safely stored to allow for email synchronization without constant user authentication.

The back end performs the main activities of the system, which include email retrieval, requests for categorization, searching operations, and connections to external systems. In addition, there is a controller responsible for managing all the requests made by the clients.

C. Intelligent Processing Implementation

Intelligent processing is carried out in a separate service that uses the Flask framework for Python. Such a service is intended to run natural language processing and hybrid intelligence processes.

Firstly, text preprocessing is done for preparing email text for further analysis. Feature extraction methods are used, such as named entity recognition, for identifying relevant entities, like dates and keywords. The rule-based system helps in detecting patterns and filtering unnecessary information from text.

Regular expressions, heuristic rules, and Large Language Model-based inference are involved into the hybrid process of deadline detection. As a result, it is guaranteed that all types of deadlines will be identified correctly. Intelligent processing works with the backend through HTTP requests written in JSON format.

D. Database Implementation

MongoDB serves as the main database for application data storage purposes. The database model is configured to manage semi-structured emails effectively.

Collections will be created for user profiles, authentication tokens, email details, and intelligence like deadlines and priorities. Indexing methods are deployed to optimize query processing, especially search queries and filters. Caching is also utilized to minimize repeated API requests.

E. External API Integration

This system connects with external applications to enhance its capabilities. Gmail API is used to fetch emails by accessing the user inbox through secured login credentials. Google Calendar API is used to add the deadline to the calendar to keep track of tasks. All API communications are handled securely using OAuth 2.0 protocols, ensuring data privacy and controlled access.

F. Performance Optimization

Asynchronous processing methods are applied to increase system efficiency. The email synchronization and intelligence processing take place simultaneously, which allows the system to manage large numbers of emails while still remaining responsive to the user.

Batch processing is applied for managing a large number of emails, while the background process allows the system to remain responsive during busy times. All these approaches collectively make the whole experience much better for the user.

G. User Interface Implementation

The CASEO system's user interface is designed to be easy to use and intuitive so that the management of emails can be carried out effectively. The focus of the interface is on decreasing cognitive load by showing only necessary information such as the prioritization of emails, extraction of deadlines, and

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

categorization. The UI is built using modern frontend technology and is based on a modular architecture.

Login Interface

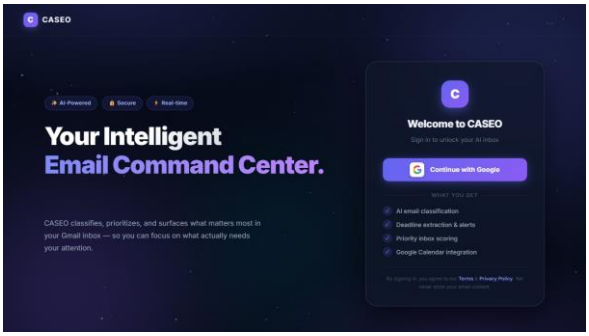

Fig: 2 Login Page

Authentication is done in a secure manner with the help of the Google OAuth 2.0 method. As shown in Fig. 2, authentication can be done using the account registered on Google.

It offers the following benefits:

- Access to user emails in a secure manner
- Removal of unnecessary credentials management
- Ease in user onboarding

The solution adopted here follows a minimalist approach to quick access to the system.

Dashboard Interface

The main dashboard, shown in Figure 3, serves as the main control center of the application, as it displays emails from the user's inbox in an organized manner.

Main characteristics of the dashboard include:

- Classification of emails into categories (Jobs, Education, etc.)
- Indication of priority with High, Medium, and Low labels
- User-friendly design that makes navigating through the dashboard easy

As emails are added to the system, the dashboard updates itself instantly and allows viewing of current information.

Moreover, the dashboard provides decision-making support by stressing timely information.

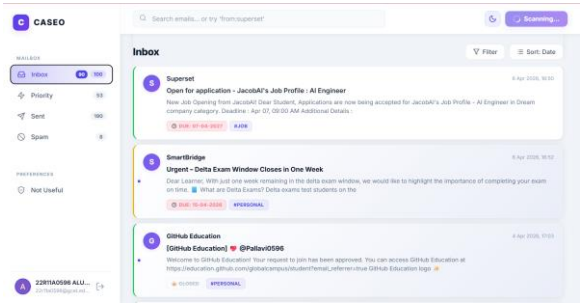

Fig: 3 Dashboard

Prioritized Email View

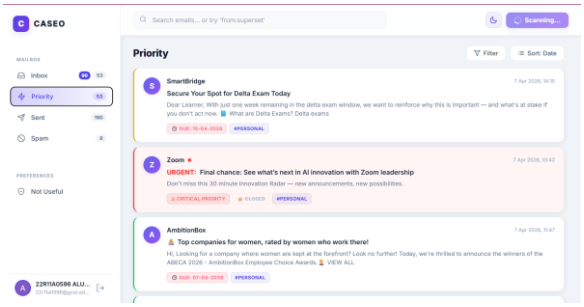

Fig: 4 Prioritized Emails

The prioritized email window, illustrated in Figure 5.3, provides a categorization of emails based on urgency categories generated by the intelligent processing component.

These advantages include:

- Instant detection of urgent emails
- Reduced time spent filtering out unimportant emails
- Increased efficiency and effectiveness

Color coding is among the visual cues that help improve readability.

Email Detail and Action Panel

This tool provides an email detail viewer that is interactive and rendered using modal windows, as shown in Fig. 5. It shows detailed information about the emails selected by users and suggests actions.

Fig: 5 Email Detail View

Some features include:

- Viewing the complete content of emails
- Creating the discovered deadlines on the calendar
- Opening selected emails in the Gmail interface
- Processing email intelligence again if needed

Furthermore, the interface can be extended by including summaries and explanations generated through AI techniques.

Calendar Integration Interface

Fig: 6 Calendar Integration

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

The calendar synchronization tool depicted in Figure 6 enables the user to integrate all extracted deadlines directly into Google Calendar.

This tool offers:

- Automatic extraction of the detected deadlines as events
- One-click scheduling capability
- Enhanced task and time management

With the use of an external calendar service, this system ensures that deadlines will not be forgotten.

Filtering and Sorting Interface

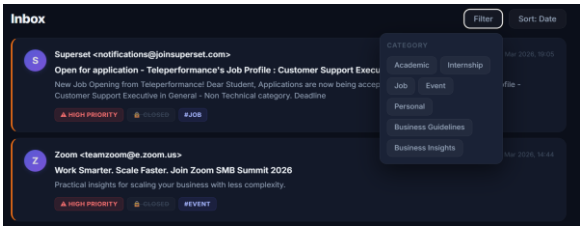

Fig: 7 Filtering

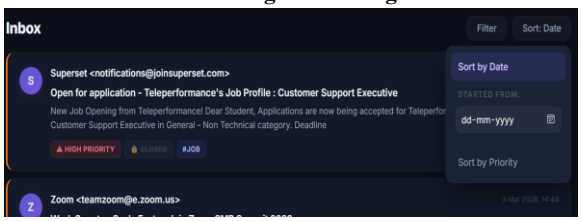

Fig: 8 Sorting Interface

The filtering sad sorting screens shown in Fig. 7 and Fig. 8 are examples of how users can manipulate email display based on different attributes.

Features supported are:

- Email filtering based on categories (for e.g. jobs, academics)
- Email sorting based on priority or date of receipt
- Fast navigation within large email folders

Such a user-friendly system becomes very useful for managing large numbers of emails efficiently.

A flexible design of the UI module ensures future scalability and facilitates inclusion of sophisticated systems such as AI-enabled email summarization and feedback facilities.

VI. RESULTS AND EVALUATION

Evaluation of the CASEO system was conducted by using the selected email data which can simulate real email contexts due to the lack of an available standard benchmark for context aware email intelligence. Three main dimensions are used in the evaluation process: email classification, extracting deadlines, and performance measurement. An email sample set consisting of 100 email samples from different email classes, such as important email, promotional email, and deadline email, was created.

Metric	**Value (%)**
Accuracy	86
Precision	84
Recall	82
F1 - Score	83

Table: 1 Email Classification Performance

The performance of the email classification algorithm was evaluated using conventional evaluation techniques such as accuracy, precision, recall, and F1 score. The results obtained from this experiment are presented in Table 1.

From the result obtained, it is clear that the hybrid method is quite efficient in balancing precision and recall for classification purposes.

A confusion matrix is presented in Table 2 to help understand classification behavior.

	Predicted Important	**Predicted Not Important**
Actual Important	42(TP)	8(FN)
Actual Not Important	6(FP)	44(TN)

Table: 2 Confusion Matrix for Email Classification

As seen from the confusion matrix, the vast majority of emails have been appropriately categorized, and only a few cases have been incorrectly recognized as positive and negative ones. The error usually appears when an email is partially relevant to a certain topic.

Deadline extraction module performance was measured according to its capacity to recognize and parse temporal expressions in emails' content. The results are shown in Table 3 below.

Metric	**Value (%)**
Extraction Accuracy	85
Exact Match Accuracy	80
Partial Match Accuracy	90

Table: 3 Deadline Extraction Performance

The system's performance is high when it comes to extracting explicitly stated deadlines using well-structured expressions. Yet, the performance drops a bit when it comes to identifying vague expressions, such as 'soon', 'by the end of the week'. This suggests some room left for optimization regarding contextual awareness.

System efficiency was assessed by analyzing the time required for sending a request and receiving a response. The results of the test are presented in Table 4 below.

Metric	**Value**

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

Average Response Time	1.3 seconds
Maximum Response Time	2.1 seconds
Processing Time per Email	~1.1 sec

Table: 4 System Performance Metrics

Judging from the observed performance, the system is capable of operating in nearly real time, which makes it applicable for real-life applications involving interaction with end users. Comparative performance with conventional email filters is provided in Table 5.

Feature	**Existing Email Systems**	**CASEO**
Basic Classification	Yes	Yes
Context Awareness	Limited	High
Deadline Extraction	Not Supported	Supported
Smart Prioritization	Limited	Advanced
Calendar Integration	Not Integrated	Integrated

Table: 5 Functional Comparison with Existing Email Systems

According to these results, while traditional filters are good at providing simple filters, they do not provide any advanced contextual analysis or action-related intelligence. On the other hand, CASEO offers advanced features like deadline awareness, email prioritization, and calendar-based actions.

In conclusion, it could be said that CASEO demonstrates consistent and relatively equal performance in all major tasks. These tests confirm the applicability of our AI-based hybrid approach in real-life situations. Some additional work may be needed to improve the algorithm of interpreting ambiguous temporal expressions, as well as testing using a larger sample of email messages.

VII. CONCLUSION

As the number of emails grows, it becomes increasingly important to implement effective methods to manage these emails in order to save time. This project proposes the creation of a system called CASEO, which can automatically analyze emails using a hybrid intelligence and provide recommendations on organizing them based on context.

This system combines rule-based algorithms, Natural Language Processing, and support from Large Language Models in order to turn unstructured data into structured, actionable information. The combination of different intelligent approaches allows the system to work with various forms of email contents and handle ambiguous and implicit information.

An important feature of this system is a multi-level deadline detector, which is able to detect both explicit deadlines and deadlines inferred based on context. With the help of pattern recognition, heuristic rules, and context-based inference, deadlines are detected in emails reliably. The ability to prioritize emails based on urgency makes the system even more useful.

In addition to email classification and deadline extraction, CASEO performs some transformation of processed emails in order to enable the user to take action with ease. Such features as calendar integration and email tagging make it easy for users to deal with information.

The system is built on a modular and layered architecture. This allows the system not only to perform better but also be easily modified, scaled up or down, and communicate effectively with all its components. Implementing asynchronous background processing is one of the solutions that made the system perform better than expected. Emails will be analyzed progressively without disrupting user experience.

In summary, this research presents a powerful way to efficiently organize emails and prevent overload. By classifying emails, extracting necessary information and making suggestions about actions, the system reduces the cognitive load and increases productivity.

VIII. FUTURE WORK

While the proposed CASEO system demonstrates effective email classification and intelligent information extraction, there are several opportunities for further enhancement to improve its functionality, scalability, and user experience.

A. Mobile Application Development

One major area of potential research includes creating a mobile app. Using a mobile app allows users to check emails, deadlines, and notifications all at once. This would be easier to use compared to the current website. Cross-platform tools can be used to allow the mobile app to work with other devices and synchronize smoothly with the current system.

B. Intelligent Notification System

The usability of the application would be greatly improved with the use of a complete and configurable notification system. This proposed notification system would have the ability to work at several levels, taking into account user preference and context.

Users could receive daily notifications on their pending deadlines, thus helping ensure that the most important tasks are brought to light early in the morning. Real-time notifications can also be used for sending alerts for urgent emails related to deadlines for submissions or even event registrations.

Emails can also be set up for certain events to remind users about them later. A context-aware notification system would help to

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

give priority to certain types of alerts to avoid unnecessary notifications.

C. AI-Based Email Summarization

Possible future improvements could include the implementation of more sophisticated methods of automatically summarizing emails through artificial intelligence algorithms. The result would be an automatic summary that can convey the most important parts of long email messages and save valuable time by avoiding reading them completely. This approach would particularly benefit the work environment and educational settings.

D. Personalized Smart Reminders

Another potential avenue would be creating an intelligent reminder system based on the behavior and interactions of the users. Analyzing the pattern of activity of the user in terms of opening emails, responding, or completing tasks, the system could create reminders for the user.

If the user responds to certain types of emails at a later time than others, then it would be beneficial to create reminders that help the user respond quickly and efficiently.

E. Integration with Productivity and Collaboration Tools

There is ample opportunity for extension by incorporating the functionality of other productivity applications, such as task managers and collaboration applications. This will allow for transforming emails into tasks with deadlines as well as collaborating with team members via the same application.

Such features will turn the tool from a simple mail organization platform into a complete productivity assistant.

F. Multi-Platform Email Support

Currently, the application can only integrate with Gmail-related services. Future development must involve expanding the scope of services to accommodate multiple email providers like Outlook and Zoho Mail. This will increase the versatility of the solution and extend its applicability across a larger range of customers using different email services.
Incorporating a single interface for working with multiple mailboxes might be considered as an additional feature for future development.

G. Advanced Context-Aware Intelligence

The algorithm's ability to understand the context of emails can be further improved by integrating more sophisticated machine learning algorithms. The use of deep learning models or even the fine-tuning of already existing language models will contribute to improving the effectiveness of information retrieval from emails.

Adding multilingual support can also be considered as an option in the course of future work on improving the algorithm's performance.

IX. ACKNOWLEDGEMENT

The authors would like to express their sincere gratitude to Mrs. B. Mamatha, Assistant Professor, Department of Computer Science and Engineering for her continuous support, valuable guidance, and constructive suggestions throughout the development of this work. Her encouragement and insights played a significant role in shaping the direction and quality of this research.

The authors also extend their appreciation to the faculty members of the Department of Computer Science and Engineering at Geethanjali College of Engineering and Technology for their support and motivation during the course of this project.
Finally, the authors would like to thank Geethanjali College of Engineering and Technology for providing the necessary resources and academic environment to successfully carry out this work.

X. RFERENCES

[1] Gryka P., Kowalski M., Nowak A., "Detection of AI-Generated Emails – A Case Study", Journal of Artificial Intelligence Research, 2024.
[2] Alsuwit K., Alotaibi F., Alharbi S., "Advancing Email Spam Classification Using Machine Learning and Deep Learning Techniques", International Journal of Computer Science and Information Security, 2024.
[3] Wu Z., Chen L., Zhang Y., "Actively Discovering New Slots for Task-Oriented Conversation", Proceedings of the Association for Computational Linguistics, 2024.
[4] Wang H., Liu B., Chen X., "Context-Aware Intent Identification in Email Conversations", Expert Systems with Applications, 2019, 126, 14–23.
[5] Nasreen G., Khan M.M., Younus M., Zafar B., Hanif M.K., "Email Spam Detection by Deep Learning Models", Journal of Information Security and Applications, 2024.
[6] Devlin J., Chang M.W., Lee K., Toutanova K., "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", arXiv preprint, 2018. https://arxiv.org/abs/1810.04805
[7] Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A.N., Kaiser L., Polosukhin I., "Attention Is All You Need", Advances in Neural Information Processing Systems, 2017, 30, 5998–6008.
[8] Brown T., Mann B., Ryder N., Subbiah M., Kaplan J., Dhariwal P., Neelakantan A., Shyam P., Sastry G., Askell A., et al., "Language Models are Few-Shot Learners", Advances in Neural Information Processing Systems, 2020, 33, 1877–1901.
[9] Zhao W., Zhou K., Li J., Tang T., Wang X., Hou Y., Min Y., Zhang B., Zhang J., Dong Z., et al., "A Survey of Large Language Models", arXiv preprint, 2023. https://arxiv.org/abs/2303.18223
[10] Kowsari K., Jafari Meimandi K., Heidarysafa M., Mendu S., Barnes L., Brown D., "Text Classification Algorithms: A Survey", Information, 2019, 10 (4), 150.
[11] Young T., Hazarika D., Poria S., Cambria E., "Recent Trends in Deep Learning Based Natural Language Processing", IEEE Computational Intelligence Magazine, 2018, 13 (3), 55–75.
[12] Google Developers, "Gmail API Documentation". https://developers.google.com/gmail/api
[13] Google Developers, "Google Calendar API Documentation". https://developers.google.com/calendar
[14] spaCy, "Industrial-Strength Natural Language Processing Library". https://spacy.io