A Cognitive Multi-Stage Autonomous Email Agent for Context-Aware Task Extraction and Priority-Oriented Action Structuring using Large Language Models

Akriti Bhanot

Researcher, New Delhi, India

M.S in Business Analytics and Information Systems, University of South Florida

Email: akritib2111@gmail.com

Abstract

In today's digital environment, emails remain the primary means for professionals and corporations to communicate with one another. Nonetheless, emails contain several embedded actions which are stated in an unstructured manner using natural language, thereby necessitating manual decoding, sorting, and execution. Such actions lead to inefficiency and cognitive overload. In this article, we propose a cognitively-based multi-staged autonomous email agent which employs the power of Large Language Models (LLMs) to transform unstructured emails into actionable intelligence through structuring them. In the proposed method, there are two stages involved in the transformation process. The first stage involves a task extraction algorithm which extracts actionable items from emails while the latter stage involves a task refinement algorithm where the extracted tasks are prioritized, timed and next steps are identified.

Keywords: Cognitive Agents, Email Automation, Large Language Models, Task Extraction, NLP, Workflow Automation, Generative AI

1. Introduction

Despite being one of the most common ways of communication, emails serve a prominent role in information exchange, task delegation, and overall coordination of tasks and activities in both personal and professional settings. At the same time, one of the key issues that comes along with emails is the unstructured nature of their content, with many actionable tasks embedded into the body of the email that are in plain language. Manually parsing this information and performing corresponding actions proves to be a cumbersome process, potentially taking a lot of time and increasing cognitive load on users and time commitment.

However, due to the appearance of large language models such as GPT or LLaMA, designing an intelligent system that is able to capture nuances of language and context, extract semantic information, and provide structured output has become quite viable. In addition to that, such models employ large transformers trained in a way that makes them capable of performing various reasoning processes.

The idea behind the project presented in this report is to create a cognitive, multi-stage autonomous agent that transforms unstructured email information into structured, actionable tasks. The process would be accomplished in the following steps:

- Extracting tasks from email text
- Contextual enhancement of identified tasks
- Generating structured output

2. Literature Review

Initially, rule-based systems were widely deployed to automate the processing of emails based on certain rules and patterns. Such techniques were efficient in simple tasks but were inflexible in handling other types of emails. Subsequently, techniques like keyword extraction and supervised machine learning were used to classify emails and retrieve basic information from them.

The advent of transformer-based models, such as BERT, marked a significant milestone in the development of Natural Language Processing models. Such models allowed machines to analyse relationships between different parts of a sentence effectively. This advancement facilitated the improvement of sentiment analysis, named entity recognition, and text classification processes. Nevertheless, the functionality of such models was confined to comprehending and classifying texts. The process of producing text and making decisions remained beyond their capabilities.

Large Language Models have made significant advances in natural language processing, and that has enabled

• Text generation based on context;

• Instruction following;

• Multistep decision making and reasoning.

Despite recent achievements, there is still an obvious gap in creating automated systems capable of accomplishing comprehensive tasks, such as text extraction, filtering, refining, and structuring data into meaningful units. The study addresses this limitation by introducing a multi-stage cognitive pipeline that integrates these functionalities into a cohesive system

3. Methodology

The proposed model is built as a cognitive multi-stage autonomous agent for processing emails and structuring them into actionable items by utilizing the potential of Large Language Models (LLMs). The proposed methodology adheres to a pipeline approach, where the responsibilities of each component include specific cognitive functions like perception, reasoning, and action planning.

Differing from other methodologies where the system works either on predefined rules or machine learning, the proposed methodology uses prompt-based interaction with LLMs. This methodology has been developed using the Python programming language and transformer-based models are incorporated via API.

3.1 System Architecture Overview

The design of the proposed architecture involves a two-phase sequential pipeline that is shown below:

- Task Extraction Phase
- Task Refinement & Structuring Phase

The execution of each phase utilises a Large Language Model and is directed using precisely designed prompts to perform human-like cognitive operations.

The flow can be described as follows:

- Unstructured Email -> Task Extraction -> Task List -> Task Refinement -> Structured Output (JSON)

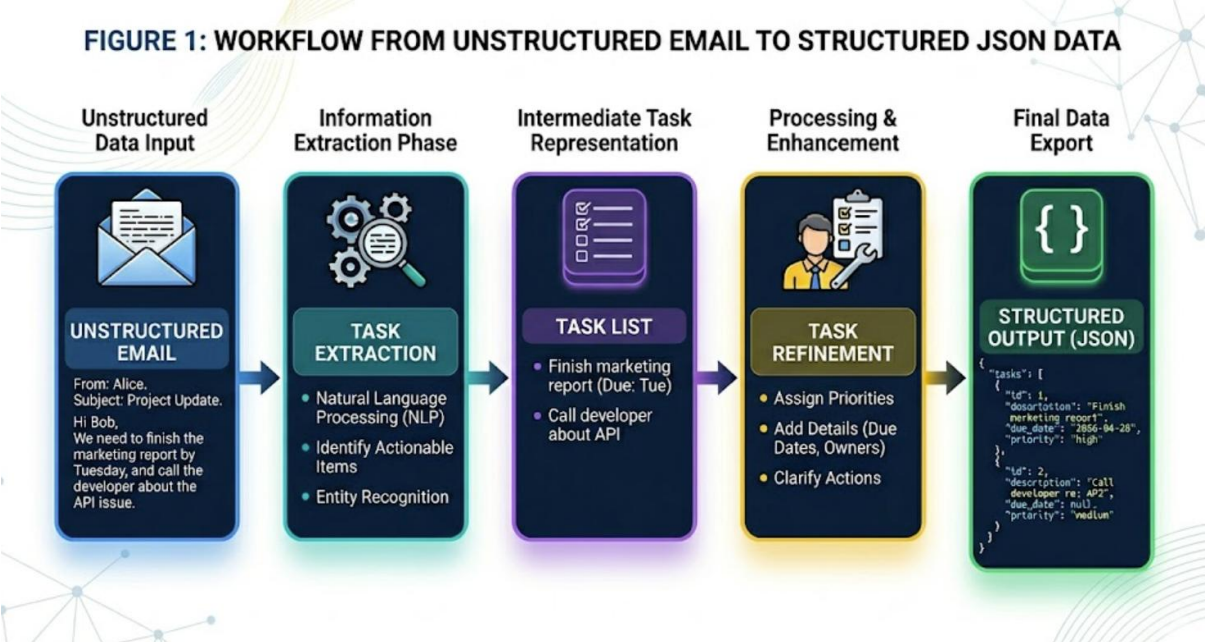

Figure 1

The multi-layer approach allows for improved interpretability and accuracy over single-shot models.

3.2 Technology Stack and Implementation Environment

The implementation of the system was done using the Python programming language because of its wide range of NLP capabilities and API integrations. The technologies used are listed below:

- **Python**: Employed to manage the pipeline and run the modular operations.
- **Hugging Face Inference Client**: Functions as the medium to communicate with the Large Language Model through API requests.
- **Meta Llama 3 8B Instruct**: Transformer-based model trained to perform natural language understanding, reasoning, and generation tasks.

The LLM functions as the reasoning engine, obviating the need for feature engineering or training the model explicitly.

3.3 Input Handling and Preprocessing

The system accepts raw email text as input. Unlike traditional NLP pipelines, minimal preprocessing is performed, as LLMs are capable of handling raw natural language effectively.

The input is directly embedded into a structured prompt, which includes:

- Context definition (role of the AI assistant)
- Instruction for task extraction
- The email content

This approach leverages the contextual understanding capabilities of transformer models, reducing the need for explicit tokenization or parsing.

3.4 Task Extraction Stage

The first stage of the system focuses on identifying actionable tasks embedded within the email text. This stage is implemented through the extract_tasks() function.

3.4.1 Prompt Design

A structured prompt is constructed to guide the LLM.The model is assigned the role of an AI assistant. The input email is provided as context.The output format is constrained to a numbered list. This prompt design ensures clarity and consistency in task extraction.

3.4.2 Processing Mechanism

The LLM processes the input by:

- Identifying action-oriented phrases
- Detecting verbs indicating tasks (e.g., schedule, send, review)
- Separating multiple instructions within a single sentence

The output is generated as a list of tasks, preserving semantic meaning while simplifying structure.

3.4.3 Cognitive Interpretation

This step involves the perception process for cognitive agents where the system will interpret the information received in the form of emails to extract relevant information related to the tasks. The Large Language Model will interpret the emails based on semantic analysis and identify the explicit and implicit actions without any prior knowledge. The model can identify action-oriented expressions, identify various tasks in one go, and even recognize the context like time sensitivity. The output at this stage is an extraction of the task list, which resembles human cognitive behaviour.

3.4.4 Example Output

Let's look at an example for an input email:

Schedule a meeting tomorrow at 5 PM and send the report urgently.

The extracted tasks from the above are:

1. Schedule a meeting tomorrow at 5 PM
2. Send the report urgently

3.5 Task Refinement and Structuring Stage

The second stage enhances the extracted tasks by adding contextual and operational details. This stage is implemented using the refine_tasks() function.

3.5.1 Prompt Design

The refinement prompt instructs the LLM to do the following:

- Add time information (if available or inferred)
- Assign priority levels (High, Medium, Low)
- Suggest next steps
- Return output strictly in JSON format

This ensures that the output is both structured and machine-readable.

3.5.2 Reasoning and Enrichment

The LLM performs multi-step reasoning to assign the following:

- Interpret urgency indicators (e.g., “urgent”)
- Infer missing details based on context
- Assign priority levels using semantic cues
- Generate actionable next steps

For instance, the word “urgently” leads to a high priority classification, while the phrase “tomorrow at 5 PM” is interpreted as a time constraint.

3.5.3 Structured Output Generation

The final output is generated in JSON format, which facilitates integration with workflow automation systems.

Example:

```
[
  {
    "task": "Schedule a meeting",
    "time": "Tomorrow at 5 PM",
    "priority": "High",
    "next_step": "Send calendar invite to participants"
  }
]
```

3.6 Prompt Engineering Strategy

Prompt engineering plays a critical role in the proposed system. Instead of training a custom model, the system leverages the pre-trained capabilities of LLMs through carefully designed prompts. PAPER

3.7 Multi-Stage Cognitive Pipeline

The proposed system follows a multi-stage cognitive pipeline, which mimics human problem-solving behaviour:

- Perception Stage → Understanding email content
- Interpretation Stage → Extracting tasks

- Reasoning Stage → Assigning priority and context
- Action Planning Stage → Generating next steps

This layered processing enables the system to handle complex and unstructured inputs effectively.

3.8 Implementation Details

To validate the proposed cognitive multi-stage autonomous email agent, the system was implemented using Python and integrated with a Large Language Model (LLM) through an API-based interface. The implementation follows a two-stage pipeline consisting of task extraction and task refinement.

The system leverages the Hugging Face InferenceClient to interact with the Meta Llama 3 8B Instruct, enabling semantic understanding and reasoning over unstructured email text.

3.8.1 System Implementation

The complete implementation of the proposed system is presented below:

email_agent.py

email_agent.py > extract_tasks

```python
from huggingface_hub import InferenceClient

# 🔑 Replace with your Hugging Face token
client = InferenceClient(token="**************************")


# ------------------------------
# STEP 1: Extract Tasks
# ------------------------------
def extract_tasks(email):

    prompt = f"""
    You are an AI assistant.

    Extract all tasks from the email below.

    EMAIL:
    {email}

    Return tasks as a simple numbered list.
    """

    response = client.chat_completion(
        model="meta-llama/Meta-Llama-3-8B-Instruct",
        messages=[
            {"role": "user", "content": prompt}
        ]
    )

    return response.choices[0].message["content"]
```

The above **email agent** is an intelligent system that automatically analyses email content to extract tasks and convert them into structured, actionable items. It reduces manual effort by understanding context and assisting in task management and decision-making.

The extract_tasks() function takes the email text as input and uses a Large Language Model to identify and extract actionable tasks from it. It returns these tasks as a simple numbered list for further processing.

```
# ------------------------------
# STEP 2: Refine Tasks (Agent Step)
# ------------------------------
def refine_tasks(tasks):

    prompt = f"""
    You are an intelligent AI agent.

    Improve the following tasks.

    For each task:
    - Add time (if possible)
    - Add priority (High/Medium/Low)
    - Suggest next step

    Return output in JSON format.
    Return only valid JSON. Do not add explanation.

    Tasks:
    {tasks}
    """

    response = client.chat_completion(
        model="meta-llama/Meta-Llama-3-8B-Instruct",
        messages=[
            {"role": "user", "content": prompt}
        ]
    )

    return response.choices[0].message["content"]


# ------------------------------
# MAIN PROGRAM
# ------------------------------
if __name__ == "__main__":

    # 🔹 Input email
    email = "Schedule a meeting tomorrow at 5 PM and send the report urgently"

```

The refine_tasks() function takes the extracted tasks and uses a Large Language Model to enhance them by adding priority, time estimates, and next steps. It converts these tasks into a structured JSON format for easy execution and integration with workflow systems.

```
64
65  # ------------------------------
66  # MAIN PROGRAM
67  # ------------------------------
68  if __name__ == "__main__":
69
70      # 🔹 Input email
71      email = "Schedule a meeting tomorrow at 5 PM and send the report urgently"
72
73      print("📧 Input Email:")
74      print(email)
75
76      # 🔹 Step 1: Extract tasks
77      tasks = extract_tasks(email)
78
79      print("\n Step 1 - Extracted Tasks:")
80      print(tasks)
81
82      # 🔹 Step 2: Refine tasks (Agent behavior)
83      final_output = refine_tasks(tasks)
84
85      print("\n Step 2 - Final Agent Output:")
86      print(final_output)
```

The **main program** acts as the controller of the system, where the input email is provided and both functions are executed in sequence. It first calls extract_tasks() to get the tasks, then passes them to refine_tasks() to generate the final structured output, and prints the results.

3.8.2 Explanation of Implementation

The implementation is structured into two primary modules corresponding to the proposed multi-stage pipeline.

Task Extraction Module

The extract_tasks() function is responsible for identifying actionable items from unstructured email text. The function constructs a prompt that instructs the LLM to extract tasks and return them as a numbered list.

This stage leverages the semantic understanding capabilities of the LLM to detect both explicit and implicit tasks without relying on predefined rules.

Task Refinement Module

The refine_tasks() function enhances the extracted tasks by introducing additional attributes such as priority, time estimation, and next steps. This stage simulates intelligent agent behavior by performing contextual reasoning.

The output is generated in JSON format, ensuring that the results are structured and suitable for integration with downstream systems such as task managers or workflow automation tools.

LLM Interaction

Both modules utilize the chat_completion() method provided by the Hugging Face inference client to interact with the LLM. The model processes the prompt and returns a response containing the generated output.

The use of prompt engineering allows the system to control the behavior of the LLM without requiring model training or fine-tuning.

3.8.3 Execution Workflow

The system operates in a sequential manner:

- The input email is provided to the system
- The task extraction module identifies actionable tasks
- The extracted tasks are passed to the refinement module
- The final structured output is generated in JSON format

This workflow demonstrates a clear separation between task identification and task structuring, improving both accuracy and interpretability.

---

4. Sample Output and Case Study

To demonstrate the effectiveness of the proposed system, a sample email is processed through the pipeline.

4.1 Input Email

Schedule a meeting tomorrow at 5 PM and send the report urgently.

4.2 Extracted Tasks

1. Schedule a meeting tomorrow at 5 PM
2. Send the report urgently

4.3 Final Structured Output

```
📩 Input Email:
Schedule a meeting tomorrow at 5 PM and send the report urgently

🧩 Step 1 - Extracted Tasks:
1. Schedule a meeting for tomorrow
2. Send the report urgently
3. Schedule the meeting at 5 PM

🤖 Step 2 - Final Agent Output:
{
  "task1": {
    "task": "Schedule a meeting for tomorrow",
    "time": null,
    "priority": "Medium",
    "next_step": "Check team members' availability and send a meeting invite by end of the day"
  },
  "task2": {
    "task": "Send the report urgently",
    "time": "immediately",
    "priority": "High",
    "next_step": "Prepare the report and send it to the concerned department right away"
  },
  "task3": {
    "task": "Schedule the meeting at 5 PM",
    "time": "tomorrow at 5 PM",
    "priority": "Low",
    "next_step": "Confirm the meeting invite with all team members and send a reminder if necessary"
  }
}
```

5. Discussion

The results demonstrate that the system successfully transforms unstructured email content into structured and actionable information. The multi-stage architecture enables effective separation of concerns, where the first stage focuses on extraction and the second stage focuses on reasoning and structuring.

The generated output is consistent, interpretable, and suitable for real-world applications such as workflow automation and productivity enhancement.

6. Conclusion

In this paper, an email agent is proposed which uses cognitive processing to extract tasks from emails automatically. As emails may contain several tasks, it would be quite difficult for users to understand and process them manually. By utilizing the Large Language Model approach, we can automate such tasks and achieve higher productivity.

Our approach involves two stages – task extraction and task refinement. Task extraction is the process of recognizing tasks mentioned in the email text. In contrast, task refinement enhances the extracted tasks by giving priorities, determining time requirements, and providing guidance about what action should be taken next. Such a split in stages allows for a more effective output.

During implementation, we found that the task extraction and refinement processes using LLMs were highly efficient, as no rules had to be defined for the program to understand and process tasks. Additionally, we were able to provide structured data in machine-readable format as an output of our approach.

In conclusion, our system is highly productive and efficient and helps users reduce cognitive loads by automating certain tasks.

---

7. Future Work

The proposed system can be further improved in several ways to make it more practical and powerful for real-world use. One important enhancement would be integrating the system with live email platforms such as Gmail or Outlook. This would allow the agent to automatically read incoming emails and generate tasks in real time, reducing the need for manual input.

Another potential improvement is the ability to analyse entire email threads instead of single emails. In real scenarios, conversations often span multiple replies, and understanding the full context across these messages would help in extracting more accurate and meaningful tasks.

The system can also be enhanced by incorporating user-specific preferences. For example, it could learn how a particular user prioritises tasks, their working style, or frequently assigned responsibilities. This would allow the system to generate more personalized and relevant outputs.

In addition, improving prompt design and fine-tuning the underlying model can increase the accuracy and consistency of results. This would help reduce errors and ensure more reliable task extraction and structuring.

Another valuable extension would be support for multilingual emails. Since communication often happens in different languages, enabling the system to understand and process multiple languages would significantly increase its usability.

Finally, the system can be integrated with task management and productivity tools such as calendars, to-do lists, or project management platforms. This would enable automatic creation of tasks, reminders, and schedules, leading to a fully automated workflow and improved efficiency