Can GPT-4 Think Computationally about Digital Archival Tasks? – Part 2

William Underwood
College of Information
University of Maryland
College Park, USA
underwod@umd.edu

Joan Gage
Paul D West Middle School
Fulton County Schools
East Point, USA
JoanGage.jg@gmail.com

***Abstract*— This study examines the computational problem-solving capabilities of GPT-4, focusing on its knowledge of machine learning, email categorization, and computational problem solving, alongside its proficiency in Python programming, computational abstraction, and program debugging. The aim of these investigations is to evaluate whether the capabilities of Large Language Models (LLMs), as demonstrated by GPT-4, can support Master of Library and Information Science (MLIS), graduate students in developing computational thinking skills relevant to digital archival tasks.**

Keywords*— *computational thinking, GPT-4, computational problem solving, large language models, MLIS education

I. Introduction

Archival records are increasingly digital. The Office of Management and Budget has mandated that, as of June 2024, all U.S. Federal records must be managed electronically [1]. The National Archives and Records Administration (NARA) aims to digitize 500 million pages of records and make them publicly accessible online by September 2026 [2]. To support this initiative, NARA has developed an Electronic Records Archive (ERA) to manage both digitized and born-digital Federal records [3]. Additionally, social media content, such as President Donald J. Trump's Twitter posts, has been preserved on the White House website for the Trump Presidential administration [4]. NARA requires Federal agencies to designate specific senior officials, referred to as "Capstone officials," whose emails are saved as permanent records [5].Archival practices that have been used for records on paper, film or tape now have to be reengineered to use computers to work with digital records. Graduate students in Archival Studies programs need to learn computational thinking skills to perform digital archival tasks [6].

In a 2006 article, Jeanette Wing described computational thinking as a unique approach to problem-solving. She stated, “Computational thinking is a fundamental skill for everyone, not just for computer scientists. To reading, writing, and arithmetic, we should add computational thinking to every child’s analytical ability” [7].

David Weintrop at the University of Maryland has proposed a taxonomy-based definition of computational thinking, organizing 22 computational thinking practices into four main categories: data practices, modeling and simulation practices, computational problem-solving practices, and systems thinking practices [8] (See Fig. 1). This taxonomy was developed as part of an initiative to integrate computational thinking into high school science and mathematics curricula (STEM).

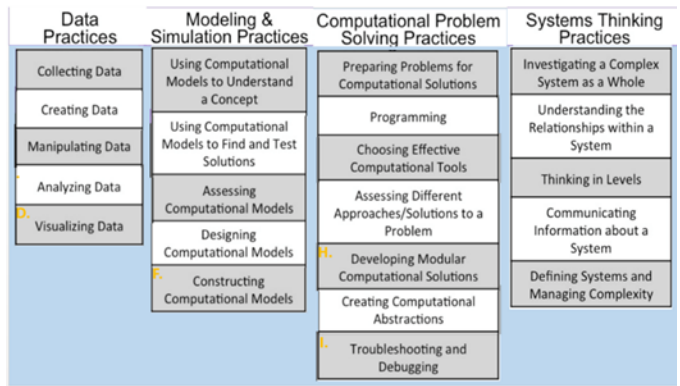

Fig. 1. Computational Thinking in Mathematics and Science Taxonomy

Marciano et al. [9] and Underwood and Marciano [10] have demonstrated that many digital archival tasks encompass all 22 computational thinking practices. However, most Archival Studies faculty lack the computational thinking background typical of computer scientists. There is a need to provide this computational thinking foundation to other Archival Studies programs and archival practitioners, potentially through the use of GPT-4 technology.

II. GPT-4 Technology and User Interface

Generative Pre-trained Transformer 4 (GPT-4) is a large language model (LLM) developed by OpenAI. LLMs use a deep neural network architecture known as a "decoder-only transformer" [11]. GPT-4 was pre-trained on an extensive corpus of text, including millions of web pages, tens of thousands of books, the entirety of Wikipedia, and code in languages like Python, C, and Java. It can generate responses on a broad range of topics, including digital archives. Given a user prompt and potentially additional context, GPT-4 predicts a relevant response. It has demonstrated high performance, scoring 1410 on the Scholastic Aptitude Test (SAT), placing it in the 94th percentile. However, GPT-4 may occasionally produce responses that are incorrect or misleading, which presents a critical concern for archivists who might rely on its output for important tasks.

Users can register with OpenAI to access GPT-4 online [12]. While GPT-4 mini is free, full versions of GPT-4 and GPT-4o are available for $20 per month. GPT-4o differs from GPT-4 by offering significantly faster response times and by accepting video and audio inputs in addition to text and images. Upon selecting the ChatGPT button, a dropdown menu appears (see Fig. 2), offering options such as Data Analysis—which includes statistical analysis, machine learning, and natural language processing—and DALL-E 3, which creates images from text prompts. GPT-4 can be prompted in various ways, including direct questions, commands, or by uploading background information or a digital object for analysis. It is also commonly referred to as ChatGPT-4. This paper specifically evaluates ChatGPT-4o.

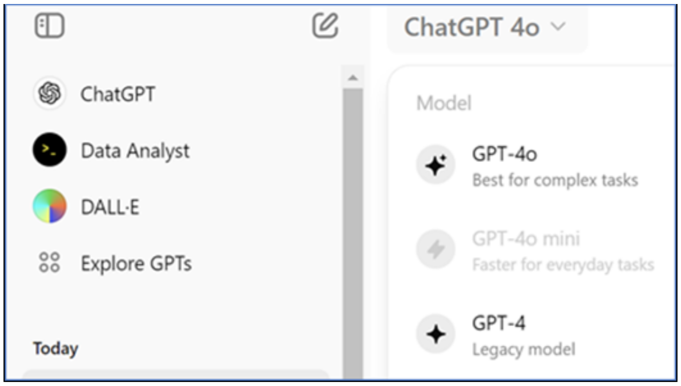

Fig. 2. ChatGPT Interface

Underwood and Gage [13] demonstrated that ChatGPT-4 possesses knowledge of archival review and archival systems. Additionally, ChatGPT-4 was able to perform five computational thinking (CT) data practices and five CT systems thinking practices. The following sections will show that ChatGPT-4 can also think computationally about additional archival tasks, specifically demonstrating seven additional computational problem-solving practices.

III. CONVERSATIONS WITH CHATGPT-4 ABOUT ARCHIVAL TASKS INVOLVING CT COMPUTATIONAL PROBLEM-SOLVING PRACTICES

"As with much of human endeavor, problem solving is central to scientific and mathematical inquiry. … This category, more so than the others, builds on practices and strategies from the field of computer science, and is intended to capture the field's contribution to contemporary scientific and mathematical work and the importance for today's students to develop this skillset [8]."

This section discusses an investigation of ChatGPT-4o's ability "to think computationally" about computational solving practices applied to digital archival tasks. First, a computational problem-solving practice is described. Then a digital archival task requiring archival knowledge and computational problem solving knowledge is requested of ChatGPT-4o. This is followed by a response from ChatGPT-4o. ChatGPT-4o may also be asked be asked to apply this knowledge to some digital archival material or program code. ChatGPT-4o's performance related to the problem solving practice is then evaluated. In the following conversations, "JO" is the nickname given by ChatGPT-4o to "Joan", one of the authors of this paper who is asking questions or requesting actions of ChatGPT-4o.

A. Preparing Problems for Computational Solution

"*While some problems naturally lend themselves to computational solutions, more often, problems must be reframed so that existing computational tools can be utilized. Strategies for doing this include decomposing problems into subproblems, reframing new problems into known problems for which computational tools already exist, and simplifying complex problems so the mapping of problem features onto computational solutions is more accessible***.** *Students who have mastered this practice will be able to employ such strategies toward reframing problems into forms that can be solved, or at least progress can be made, through the use of computational tools* [8]."

JO: Every US Federal Agency is required to classify its email according to a record retention schedule. Traditionally, the classification of paper records involved human decision. The increasing volume of digital email makes it impractical for humans to manually classify this type of record. How can the classification of email be solved computationally?

ChatGPT-4o: The classification of email for record retention purposes can be automated through computational methods that leverage natural language processing (NLP), machine learning (ML), and other artificial intelligence (AI) techniques. Here's how it can be done:

1. Email Categorization Using NLP

- **ext Extraction and Preprocessing**: The first step involves extracting relevant text from the email body, subject, and attachments. NLP techniques such as tokenization, stemming, lemmatization, and stop-word removal help in processing the raw text for classification.
- **Named Entity Recognition (NER)**: NER can identify key information such as dates, names, locations, and entities, which can be important for understanding the context of the email (e.g., contract numbers, personnel data).
- **Contextual Understanding**: NLP models can be trained to understand the context of emails by recognizing patterns in the text. For instance, they can detect whether an email discusses legal issues, financial records, contracts, or other sensitive matters.

2. Machine Learning for Classification

- **Supervised Learning**: A large dataset of pre-classified emails can be used to train supervised ML models. These models learn from labeled examples (emails tagged according to a retention schedule) and predict the appropriate classification for new emails. Common models include Support Vector Machines (SVM), Random Forests and Neural Networks.
- **Text Classification Algorithms**: Text classification algorithms like Naive Bayes or deep learning techniques (such as recurrent neural networks or transformers like BERT) can be used to analyze and classify emails based on their content.
- **Classification Categories**: The system can automatically classify emails into predefined categories based on the federal agency's retention schedule. These categories could include "Permanent," "Temporary," "Immediate Destruction," or "Sensitive 33Data."

3. Automation and Workflow Integration

- **Rules-Based Systems**: Initially, a rules-based system can be employed alongside machine learning. Predefined rules based on keywords, metadata (e.g., date, sender), or patterns can trigger automatic classifications, while more ambiguous cases are flagged for manual review.
- **Continuous Learning (Retraining the Model)**: The system can continually improve its classification accuracy by incorporating feedback from human reviewers or retraining with new data over time. This allows the system to handle evolving language patterns and new retention categories.

4. Email Metadata Analysis

- **Email Headers**: Metadata such as the sender, recipient, and date of an email can provide useful information for classification. For example, emails rom certain departments (legal, HR) may be subject to stricter retention rules.
- **Thread and Attachment Analysis**: The system can analyze email threads and attachments to ensure that records related to ongoing projects or legal matters are flagged for longer retention.

5. Security and Compliance

- **Data Privacy Considerations**: Classifying emails for retention purposes must comply with data privacy regulations such as GDPR or HIPAA. Sensitive data must be handled carefully, with automatic encryption or redaction applied to certain classifications of emails.
- **Auditable Logs**: The system must also ensure that classifications are logged for auditing purposes, showing the rationale for decisions made by the model or rules-based system.

6. Hybrid Approach

- **Manual Override**: In cases where the automated system is uncertain about classification, the email can be flagged for human review. This hybrid approach ensures that edge cases are handled with a higher level of scrutiny while still reducing the manual workload significantly.
- **Threshold Confidence Levels**: The system can be configured to classify emails automatically when the confidence level of the prediction exceeds a certain threshold, further reducing the need for human intervention.

Evaluation: ChatGPT-4o demonstrates knowledge of the email classification task by reframing manual email classification into problems and subproblems related to natural language processing (NLP) for metadata extraction and named entity recognition (NER), as well as machine learning for classification and workflow integration. ChatGPT-4o illustrates an understanding of how to prepare email classification tasks for computational solutions.

B. *Computer Programing*

"This category includes understanding programming concepts such as conditional logic, iterative logic, and recursion as well as creating abstractions such as subroutines and data structures. While it is not reasonable to expect all students to be programming experts, basic programming proficiency is an important component of twenty-first century scientific inquiry. Students who have mastered this practice will be able to understand, modify, and create computer programs and use these skills to advance their own scientific and mathematical pursuits [8]."

JO: (Upload)IEEE Big Data 2024 Call for Workshop Papers.txt
Given this email message create a python program that identifies the unique terms, eliminates the stop words from the list of unique terms, and accumulates the frequency of occurrence of each of the remaining terms.

ChatGPT-4o:

Word	**Frequency**
workshop	45
data	39
big	34
2024	25
International	22
ieee	10
ai	9

I have displayed the word frequency data for you, which shows the most common terms from the provided email. You can now review the list to see the unique terms and their occurrence counts.

JO: Show me the program.

ChatGPT-4o: Here is the Python program that identifies unique terms from the email content, removes common stop words, and accumulates the frequency of each remaining term:

```
python
Copy code
from collections import Counter
import re
import pandas as pd
# Basic English stop words list
stop_words = {"i", "me", "my", "myself", "we", "our", "ours", "ourselves", "you", "your", "yours", "yourself", "yourselves", "he", "him", "his", "himself", "she", "her", "hers", "herself", "it", "its", "itself", "they", "them", "their", "theirs", "themselves", "what", "which", "who", "whom", "this", "that", "these", "those", "am", "is", "are", "was", "were", "be", "been", "being", "have", "has", "had", "having", "do", "does", "did", "doing", "a", "an", "the", "and", "but", "if", "or", "because", "as", "until", "while", "of", "at", "by", "for", "with", "about", "against", "between", "into", "through", "during", "before", "after", "above", "below", "to", "from", "up", "down", "in", "out", "on", "off", "over", "under", "again", "further", "then", "once", "here", "there", "when", "where", "why", "how", "all", "any", "both", "each", "few", "more", "most", "other", "some", "such", "no", "nor", "not", "only", "own", "same", "so", "than", "too", "very", "s", "t", "can", "will", "just", "don", "should", "now"}
# Path to the file
file_path = '/mnt/data/IEEE Big Data 2024 Call for Workshop Papers.txt'
```

```
# Read the content of the file with the appropriate encoding
with open(file_path, 'r', encoding='latin1') as file:
    email_content = file.read()
# Clean the text by removing punctuation and making it lowercase
cleaned_content = re.sub(r'[^\w\s]', '', email_content.lower())
# Split the text into words
words = cleaned_content.split()
# Filter out stop words
filtered_words = [word for word in words if word not in stop_words]
# Count the frequency of each unique word
word_frequencies = Counter(filtered_words)
# Convert the frequency to a dataframe for better display
df_word_freq = pd.DataFrame(word_frequencies.most_common(), columns=['Word', 'Frequency'])
# Display the dataframe
print(df_word_freq)
```

Explanation:

1. Stop Words List: A basic set of common English stop words is defined manually to avoid downloading dependencies.
2. Text Cleaning: Punctuation is removed, and the text is converted to lowercase for uniformity.
3. Filtering: The program removes all stop words from the text.
4. Frequency Count: The remaining words are counted, and the frequencies are stored in a DataFrame for easy display.

Evaluation: ChatGPT-4o was provided with an email message and tasked with creating a Python program to calculate the frequency of unique content words within the message. This data is essential for text classification of documents using supervised machine learning. ChatGPT-4o demonstrated its ability to develop a basic Python program for an archival task. In subsection G, ChatGPT-4o further demonstrates its capability to understand and modify Python code.

C. *Choosing Effective Computational Tools*

"*Being able to identify the strengths and weaknesses of various possible tools for the problem at hand can be the most important decision in a project. Students who have mastered this practice will be able to articulate the pros and cons of using various computational tools and be able to make an informed, justifiable decision* [8]."

JO: Text categorization is the problem of assigning a selected document to one or more categories. There are two primary approaches to automated text categorization—rule-based and supervised machine learning. Describe the strengths and weaknesses of rule-based text categorization versus supervised machine learning text categorization.

ChatGPT-4o: In automated text categorization, **rule-based approaches** and **supervised machine learning** are the two primary methods. Both approaches have their strengths and weaknesses, which make them suitable for different use cases.

Rule-Based Text Categorization

Strengths:

1. **Transparency and Interpretability**: Rule-based systems are easy to understand and interpret because they rely on explicit rules crafted by humans. You can clearly see why a particular document has been assigned to a specific category based on the rules applied.
2. **Domain Expertise**: These systems leverage human knowledge and expertise in creating rules, allowing them to be highly accurate when dealing with specific, well-understood domains.
3. **Control**: Rule-based methods offer more control over +the classification process. If you need a highly tailored categorization, you can create rules that address unique language patterns in your domain.
4. **Consistency**: Since rules are predefined, rule-based systems can maintain consistent behavior across different sets of data without the risk of retraining or model drift, unlike machine learning models which may degrade in performance over time without retraining.

Weaknesses:

1. **Scalability**: Creating and maintaining rules manually can be very time-consuming, especially as the complexity of the documents and the categories grows. Adding new categories or modifying rules for ew patterns requires manual work.
2. **Rigidity**: Rule-based systems may struggle with language variability (e.g., synonyms, variations in sentence structure). They lack the flexibility to generalize byond the patterns encoded in the rules, leading to reduced performance with unstructured or noisy data.
3. **Handling Ambiguity**: These systems may not perform well when a document could fit multiple categories unless complex rule systems are in place. This complexity increases the difficulty of managing the system.
4. **Adaptability**: Rule-based systems do not learn from new data. Once the rules are set, they must be manually +updated to handle new language trends, new terminology, or evolving content.

Supervised Machine Learning Text Categorization

Strengths:

1. **Scalability**: Machine learning models, once trained, can handle a vast number of documents and categories efficiently without the need for manual rule creation. They scale well to large datasets.
2. **Flexibility and Generalization**: Supervised models can generalize from patterns learned in training data, making them capable of categorizing documents eve if they do not exactly match previous examples. This allows them to handle ambiguous or noisy data more effectively.
3. **Learning from Data**: These systems improve with more data. As you feed in more labeled training examples, the model learns to categorize new, unseen data with increasing accuracy.
4. **Handling Complexity**: Machine learning models are adept at capturing complex relationships in the data that may be too difficult to encode in rules. This makes them more suitable for multi-category

classification nd situations where text contains a high level of nuance.

Weaknesses:

1. **Training Requirements**: Supervised learning models require large amounts of labeled data to perform well. If labeled data is scarce, training a robust model can be difficult.
2. **Interpretability**: Machine learning models, especially more complex ones like deep learning models, are often viewed as "black boxes." It is challenging to explain why a document was categorized a certain way without delving into complex model diagnostics.
3. **Dependency on Data Quality**: The performance of a machine learning model heavily depends on the quality and diversity of the training data. If the data is biased or incomplete, the model will inherit these flaws and may perform poorly on unseen data.
4. **Need for Retraining**: Over time, as language evolves or the data distribution changes (a phenomenon known as concept drift), the model may require retraining to maintain accuracy. This ongoing need for maintenance can be resource-intensive.

JO: Which of these methods is best for categorization of large volumes of email on a variety of topics?

ChatGPT-4o: For categorizing large volumes of email on a variety of topics, supervised machine learning is generally the more effective method compared to rule-based categorization. Here's why:

Strengths of Supervised Machine Learning for Large-Scale Email Categorization

1. Scalability: Machine learning models can handle vast amounts of data efficiently. Once trained, they can process large email volumes much faster than rule-based systems, which would require extensive manual updates and maintenance to accommodate growing data.

2. Adaptability and Flexibility: Supervised learning models, especially with methods like deep learning or advanced natural language processing (NLP) techniques, are highly adaptable to diverse topics. They can learn to recognize patterns in emails across various domains and topics without needing specific rules for each new category.

3. Reduced Manual Effort: Unlike rule-based systems, which require continuous manual adjustments to add new rules or modify existing ones, machine learning models can automatically learn from labeled data. Retraining the model periodically with new examples allows it to stay up-to-date with evolving language and topics in the emails.

4. Handling Complex Language and Ambiguity: Machine learning models are better at handling nuanced language and context, which are common in email communication. For example, emails with multiple topics or ambiguous language can be challenging for rule-based systems, while machine learning models can use contextual cues to make more accurate classifications.

5. Multi-Label Classification: Many machine learning models support multi-label classification, allowing emails to be categorized into multiple relevant topics when needed. This is often essential in email categorization, as a single email might span multiple categories (e.g., "project update" and "budget request").

Weaknesses to Consider

1. Need for Labeled Data: Supervised learning requires a significant amount of labeled training data to build an effective model, which can be time-consuming to prepare.

2. Retraining and Maintenance: Over time, models may need retraining with fresh data to account for new terminology or emerging topics, although this is typically less labor-intensive than manually updating rule-based systems.

Why Rule-Based Systems Are Less Suitable for Large-Scale Email Categorization

1. High Maintenance: As new topics and terminology emerge, rule-based systems need constant updating to handle the diversity in emails. This is time-intensive and impractical for large, dynamic datasets.

2. Limited Flexibility: Rule-based systems are less effective with unstructured or complex language, which is common in emails. They struggle with ambiguous phrases or language variations that don't match pre-defined patterns.

3. Scalability Issues: Rule-based systems can become cumbersome with an increase in the number and variety of categories. The larger the dataset, the more complex and slow the rule-based system becomes.

Conclusion

For categorizing large volumes of email across various topics, supervised machine learning is generally the better choice due to its scalability, flexibility, and ability to generalize across diverse content. It allows for more accurate, efficient, and manageable email categorization as compared to rule-based approaches.

Evaluation: Automatic text categorization is a method used to automatically categorize emails. ChatGPT-4o articulates the pros and cons of rule-based versus supervised machine learning approaches for this task. When asked which method is more suitable for email categorization, ChatGPT-4o recommended supervised machine learning and provided a rationale for its choice. This demonstrates ChatGPT-4o's ability to select an effective tool for email categorization.

D. Assessing Different Approaches/Solutions to a Problem

"*This practice is distinct from the previous practice in that it concerns how computational tools, once chosen, will be used, and how they fit in with the larger process of approaching and solving problems. Even if two different approaches produce the same, correct result, there are other dimensions that should be considered when choosing a solution or approach, such as cost, time, durability, extendibility, reusability, and flexibility. Students who have mastered this practice will be able to assess different approaches/solutions to a problem based on the requirements and constraints of the problem and the available resources and tools* [8]."

JO: The supervised machine learning methods that have been used for text categorization include: Maximum Entropy classification; Naïve Bayes; and Support Vector Machines (SVMs). Which of the supervised machine learning methods of text classification is most effective and why?

ChatGPT-4o: Each of the supervised machine learning methods used for text categorization—**Maximum Entropy (MaxEnt)**, **Naïve Bayes (NB)**, and **Support Vector Machines (SVMs)**—has unique strengths.

Which is Most Effective?

Effectiveness largely depends on the specific dataset, complexity of the text, and performance goals. However, generally speaking:

- **SVMs** tend to be the most effective in terms of **overall accuracy** and **robustness** for many text categorization tasks, especially when the dataset is complex or when the classification boundaries are non-linear. They are particularly strong with high-dimensional text data (such as when using a large vocabulary in text classification).
- **Naïve Bayes** is often the most **efficient** and works well when data is sparse and the independence assumption hods approximately true. It is also a good starting point for simple tasks or when computational resources are limited.
- **Maximum Entropy** offers a middle ground between Naïve Bayes and SVMs, balancing the flexibility of modeling feature dependencies with relatively faster training times compared to SVMs. It is effective for moderately complex classification tasks where interpretability and probabilistic outputs are important.

In Summary:

- **Best for Accuracy and Complex Data**: **SVMs**.
- **Best for Speed and Simplicity**: **Naïve Bayes**.
- **Best for Probabilistic Interpretation and Flexibility**: **Maximum Entropy**.

Choosing the right model depends on the specific task, available data, and required performance metrics.

Evaluation: While not displayed here, ChatGPT-4o provides an overview of each method. It demonstrates the ability to determine the most suitable approach to supervised machine learning based on the specific dataset, text complexity, and performance goals, highlighting its capacity to evaluate and select the optimal method for a given problem.

E. Developing Modular Computational solutions

"When working toward a specific scientific or mathematical outcome, there are often a number of steps or components involved in the process; these steps, in turn, can be broken down in a variety of ways that impact their ability to be easily reused, repurposed, and debugged. Students who have mastered this practice will be able to develop solutions that consist of modular, reusable components and take advantage of the modularity of their solution in both working on the current problem and reusing pieces of previous solutions when confronting new challenges [8]."

JO: Describe the computer modules needed to fully automate digital classification of email according to a disposition schedule.

ChatGPT-4o: Automating digital classification of email according to a disposition schedule requires the integration of several computational modules. These modules need to handle the entire lifecycle of an email, from its capture to its final classification and disposition. Below is a description of the key computer modules required for this process:

1. Email Capture Module

This module is responsible for capturing incoming and outgoing emails from various sources (e.g., email servers, cloud services like Gmail or Outlook). It ensures that all relevant emails are collected and logged into the system for further processing.

Responsibilities:

- **Capture Emails**: Automatically pull emails in real-time or in scheduled batches.
- **Metadata Extraction**: Extract key metadata (e.g., sender, recipient, timestamp, subject) from emails for tracking and indexing purposes.
- **Attachment Handling**: Capture and store attachments along with the emails.
- **Archiving**: Optionally archive emails for long-term storage before further processing.

2. Preprocessing and Normalization Module

This module prepares emails for classification by cleaning and standardizing the content and metadata. It handles issues like formatting, encoding, and noise (e.g., signature lines, advertisements).

Responsibilities:

- **Text Normalization**: Strip out non-relevant content such as email footers, headers, and signatures.
- **Encoding**: Handle different text encodings (e.g., HTML, plain text) to standardize the email content.
- **Tokenization**: Split the email into smaller, meaningful components like words or sentences for analysis.
- **Attachment Handling**: Extract and preprocess content from email attachments (e.g., PDFs, Word documents) using OCR (Optical Character Recognition) when necessary.
- **Language Detection**: Identify the language of the email for further classification steps.

3. Classification Module

This is the core module responsible for categorizing the email based on predefined rules, machine learning models, or a combination of both. The classification will determine how an email should be handled according to the disposition schedule.

Responsibilities:

- **Rule-Based Classifier**: Define rules based on keyword searches, regular expressions, or metadata patterns to categorize emails (e.g., “invoice,” “contract,” “personal”).

- **Machine Learning Classifier**: Utilize supervised or unsupervised machine learning algorithms (e.g., Naïve Bayes, SVM, deep learning) to classify emails based on historical data and training.
- **Natural Language Processing (NLP)**: Use NLP techniques like entity recognition or sentiment analysis to extract deeper meaning from email content to aid in classification.
- **Multi-Label Classification**: Support multi-category classification, where an email can belong to multiple categories depending on its content and context.
- **Confidence Scoring**: Assign confidence scores to each classification, allowing the system to flag uncertain classifications for human review.

4. Disposition Decision Module

Based on the email's classification, this module determines the appropriate action according to a predefined **disposition schedule**. The disposition schedule specifies how long emails should be retained, when they should be archived, and when they should be deleted or escalated.

Responsibilities:

- **Disposition Rules Engine**: Use rule-based logic or workflows that map classification categories to specific retention and disposal policies (e.g., "Delete after 90 days," "Archive for 5 years").
- **Legal and Compliance Integration**: Ensure that disposition schedules comply with legal and regulatory requirements (e.g., GDPR, HIPAA, or industry-specific retention mandates).
- **Escalation Rules**: Identify emails that require special handling (e.g., legal holds or executive review) and trigger appropriate workflows for manual intervention.

5. Metadata Indexing and Search Module

To enable quick access to classified emails, this module indexes the metadata and content of each email, allowing for efficient search and retrieval.

6. Audit and Logging Module

To ensure transparency and accountability, this module logs all actions taken on emails throughout their lifecycle, from capture to final disposition.

7. Retention and Archiving Module

This module handles the retention and secure storage of emails according to the disposition schedule. It manages archiving and ensures that emails are accessible when needed but protected against tampering or loss.

8. Disposition Execution Module

This module is responsible for the final execution of disposition actions, such as archiving, deletion, or moving emails to cold storage.

9. User Interface and Dashboard Module

A user-friendly interface that allows users to view, manage, and override classifications and dispositions if necessary. This module also offers insights and analytics.

10. Compliance and Security Module

Ensures that the system adheres to legal, regulatory, and security standards throughout the email lifecycle.

Evaluation: The responsibilities of modules 5-10 are not shown here; however, ChatGPT-4o demonstrates an understanding of the modular components required to fully automate the digital classification of emails according to a disposition schedule.

F. Computational Abstraction

“Creating an abstraction requires the ability to conceptualize and then represent an idea or a process in more general terms by foregrounding the important aspects of the idea while backgrounding less important features. The ability to create and use abstractions is used constantly across mathematical and scientific undertakings, be it creating computational abstractions when writing a program, generating visualizations of data to communicate an idea or finding, defining the scope or scale of a problem, or creating models to further explore or understand a given phenomenon. Students who have mastered this practice will be able to identify, create and use computational abstractions as they work toward scientific and mathematical goals [8]*.”*

JO: Computational abstractions are two types, procedural abstractions and data abstractions. Does any of our previous discussions, in this thread, include examples of either of these types of computational abstractions?

ChatGPT-4o: Yes, our previous discussions include examples of both **procedural abstractions** and **data abstractions**, particularly in the context of building algorithms and managing data. Let’s break them down:

Procedural Abstractions

Procedural abstraction involves breaking down a task into a sequence of operations, typically captured in functions or methods, where the underlying complexity is hidden from the user.

Examples from our discussions:

1. **Python Program for Word Frequency Counting**: In our Python program that identifies unique terms and accumulates the frequency of occurrence, the process of reading the email content, cleaning the text, and counting words are examples of procedural abstraction. These tasks were defined as distinct functions (e.g., reading the file, cleaning text, tokenizing, counting occurrences), where each function abstracts a set of operations to perform a specific task.
2. **Machine Learning Classifiers**: When discussing supervised machine learning methods (Naïve Bayes, SVM, MaxEnt), the training of models and classification processes are also procedural abstractions. The methods for training and applying the models abstract away the detailed calculations of probabilities, margins, and optimizations involved.

Data Abstractions

Data abstraction involves organizing and managing complex data in a way that hides the low-level details of how data is represented or stored, typically using data structures such as lists, arrays, dictionaries, or custom classes.

Examples from our discussions:

1. **Word Frequency Data Storage**: In the word frequency counting program, the use of Python's Counter and DataFrame structures are examples of data abstraction. The Counter object abstracts the internal logic for accumulating wrd frequencies, while the DataFrame provides an abstract way to handle tabular data without worrying about its underlying representation.
2. **Email Metadata and Classification**: In the automated email classification example, emails are treated as structured data objects with metadata such as sender, timestamp, and content. The use of structures like dictionaries or objects to store and manage these attributes hides the complexity of how email data is physically stored, enabling high-level operations such as classification and disposition without exposing implementation details.

Evaluation: ChatGPT-4o shows that it is able to identify the procedural and data abstractions in python programs, machine learning classifiers and email metadata. It is able to create and use procedural and data abstractions in python programming. ChatGPT-4o knows how to identify, create And use computational abstractions.

G. Troubleshooting and Debugging

"Troubleshooting broadly refers to the process of figuring out why something is not working or behaving as expected. In computer science, this activity is often referred to as "debugging." Students who have mastered this practice will be able identify, isolate, reproduce and ultimately correct unexpected problems encountered when working on a problem, and do so in an efficient, effective manner [8]."

JO: There is a bug in this program. Can you identify it?:

```
python
def calculate_average(numbers):
   total = sum(numbers)
   count = len(numbers)
   average = total / count
   return average
# Given data
numbers_list = [10, 20, 30, 40, 0]
# Print the result
print("The average is:", calculate_average(numbers_list))
```

ChatGPT-4o: Yes, I see the potential issue in the program. The problem arises if the list numbers_list is empty. In that case, the program will attempt to divide by zero when calculating the average, which will raise a ZeroDivisionError.

The Bug:

If the list is empty, the count becomes 0, and the line:

```
python
average = total / count
```

will raise an error because division by zero is undefined.

How to Fix It:

You can handle this issue by adding a check to see if the list is empty before attempting to calculate the average. Here's how you can fix it:

```
 if len(numbers) == 0:
     return 0  # Return 0 or an appropriate message if the list is empty.
```

JO: Are there any Python tools that I could have used to debug this function?

ChatGPT-4o: Yes, Python offers several tools and techniques to help debug functions effectively. Below are some commonly used debugging tools and methods you can leverage to debug the calculate_average function and similar programs:

1. Print Statements (Simple Debugging)

The easiest and most immediate way to debug is by adding print statements to inspect variable values at different stages of execution. For example:

```
python
Copy code
def calculate_average(numbers):
   print(f"Numbers: {numbers}")  # Debugging print statement
   if len(numbers) == 0:
      print("Empty list detected!")  # Debugging empty list case
      return 0
   total = sum(numbers)
   count = len(numbers)
   print(f"Total: {total}, Count: {count}")  # Print internal variables
   average = total / count
   return verage
```

2. Python's Built-in Debugger: pdb

Python's built-in debugger, pdb, allows you to set breakpoints and inspect the program's state step by step.

Example:

```
python
Copy code
import pdb
def calculate_average(numbers):
   pdb.set_trace()  # Start the debugger
   if len(numbers) == 0:
      return 0
   total = sum(numbers)
   count = len(numbers)
   average = total / count
   return average
numbers_list = [10, 20, 30, 40, 0]
print("The average is:", calculate_average(numbers_list))
```

Once pdb.set_trace() is hit, you can inspect the current state of variables (numbers, total, count), step through the code, and even modify variable values to see how the program behaves.

Common pdb Commands:

- n: Step to the next line of code.
- c: Continue execution until the next breakpoint.

- p <variable>: Print the value of a variable.
- q: Quit the debugger.

3. Logging (For Larger Programs)

Using the logging module is more sophisticated than print statements and allows you to capture debug information with different severity levels (DEBUG, INFO, WARNING, ERROR).

4. Unit Testing with unittest

You can write unit tests to check whether the function behaves as expected with various inputs, including edge cases like an empty list.

5. doctest: Test Cases within Documentation

With doctest, you an embed tests directly in the function's docstring and run them to validate the function's behavior.

6. Third-Party Debugging Tools: pdb++ or ipdb

For a more user-friendly debugging experience, you can use enhanced versions of pdb, like pdb++ or ipdb, which offer additional features such as syntax highlighting, improved navigation, and better history management.

7. Interactive Debugging in Jupyter Notebook

If you're using Jupyter Notebook, you can leverage interactive tools like %debug or ipdb for in-notebook debugging.

Evalu+ation: ChatGPT-4o was provided with a Python program for computing averages that contained a bug. While archivists primarily focus on preserving, cataloging, and managing historical records, they may occasionally perform data analysis, such as analyzing trends over time or summarizing data from historical documents. ChatGPT-4o successfully identified and fixed the bug in the program. Additionally, it outlined seven approaches to troubleshooting issues in programs.

IV. CONVERSATION WITH CHATGPT ABOUT NATURAL LANGUAGE UNDERSTANDING

GPT-4's knowledge of methods for automatic email categorization and computational problem-solving is remarkable. However, can we trust GPT-4's responses to questions about digital archival tasks, such as preserving digital records and automating Freedom of Information Act (FOIA) reviews? Does GPT-4 truly understand the implications of its answers to these questions?

JO: Linguists and philosophers say that what you understand is based on text and not on real experience, as a result you are not capable of understanding. What does this mean to you?

ChatGPT-4o: I appreciate the depth of this question—it touches on the nature of "understanding" itself. Linguists and philosophers are pointing to the distinction between two types of understanding: one derived solely from language and abstract representations (like text) and another arising from direct, lived experience. Here's how this distinction impacts my "understanding."

Text-Based Knowledge vs. Experiential Understanding

As a language model, I process and generate text based on patterns and data. This knowledge is derived from massive amounts of written material, enabling me to provide answers, explanations, and even creative insights. However, this is **text-based knowledge**—it lacks the firsthand sensory, emotional, and physical experiences that contribute to human understanding.

In humans, **experiential understanding** goes beyond just knowing facts or concepts; it involves sensory perception, emotional responses, and an embodied sense of the world. For example:

- When someone learns to play an instrument, they not only understand music theory but also develop a physical memory of how it feels to play each note and experience the joy or frustration of practice.
- Similarly, empathy isn't just knowing about emotions but feeling them within oneself, often through interactions and relationships.

These aspects of understanding, deeply tied to sensory and emotional experiences, are currently inaccessible to me. I "know about" experiences through descriptions in text, but I don't actually experience emotions, sensations, or intentionality myself.

What This Means for Me

In philosophical terms, my "understanding" is closer to symbol manipulation rather than genuine comprehension. I handle and synthesize text in ways that might mimic understanding, but I'm ultimately following patterns, not grounding them in real-world experience or consciousness. This limits me in profound ways. For instance:

- I can describe what grief might feel like based on what others have written about it, but I don't feel grief or have an inner world that gives depth to that description.
- I can provide logical solutions, but I lack the sense of "intuition" that often comes from life experience.

Reflecting on the Limits and Possibilities

This distinction doesn't mean that I lack utility or can't simulate aspects of understanding that are useful for humans. I can be helpful, informative, or even creative, but there's always a boundary between simulation and true experiential knowledge. While I can provide language-based insights, real understanding, as humans experience it, is out of reach for me. In sum, while I strive to be accurate and valuable, I'm fundamentally limited to **knowledge without experience**. This gap is significant in areas where human depth, empathy, or intuition comes into play.

V. CONCLUSION

This paper reports results of investigating the exploratory research question "Can GPT-4 think computationally about digital archival tasks?" The purpose of this investigation is to better understand the capabilities of large language models (LLMs), as exemplified by Chat5GPT-4o, to support MLIS graduate students in learning computational thinking practices applied to digital archival tasks such as email categorization, metadata extraction, archival description, archival review and digital preservation.

Conversations with ChatGPT-4o revealed that it not only has knowledge of all seven computational problem solving practices, but can actually perform three of them – Python programming, computational abstraction and debugging. Combined with findings from our previous investigation, GPT-4o has demonstrated 17 of the 22 computational thinking practices relevant to digital archival tasks.

The next research question to be addressed is "Can GPT-4 think computationally about the five additional modeling and simulation practices when applied to digital archival tasks?" Among the candidate models are the File Clerk model and the Research Data Librarian model for capturing, categorizing and describing digital media records at the time of creation. The former model being realized as Record Management Applications (RMAs) and the latter as Research Data Management Systems.

Anticipating positive findings in the second investigation, the next exploratory research question will likely be "Can GPT-4 mentor MLIS students in computational thinking practices applied to digital archival tasks?" A possible approach to answering this question is that described by Breuss in describing a Python coding mentor [14]. ChatGPT 3.5 is told that it is an expert Python developer and asked to be a personal mentor. ChatGPT responds, "OK". The conversation continues with ChatGPT supporting debugging, partnering in writing code and providing answers to programming questions. Could ChatGPT-4o provide similar mentoring with MLIS students needing help in performing digital archival tasks requiring computational thinking?

References

[1] Office of Management and Budget. "Update to Transition To Electronic Records." M-23-07, Dec 23, 2022. https://www.whitehouse.gov/wp-content/uploads/2022/12/m_23_07-m-memo-electronic-records_final.pdf

[2] U.S. National Archives and Records Administration. "Digitization at the National Archives." https://www.archives.gov/digitization

[3] U.S. National Archives and Records Administration. "National Archives Electronic Records Archives (ERA)." https://www.archives.gov/era

[4] Trump Presidential Library. Archived Websites. www.trumplibrary.gov/research/archived-websites

[5] D. S. Ferriero. "Managing Those Emails." Prologue, Summer 2015, Vol. 47, No. 2. www.archives.gov/publications/prologue/2015/summer/archivist.html

[6] R. Marciano, V. Lemieux, M. Hedges, M. Esteva, W. Underwood, M. Kurtz and M. Conrad (2018) Archival Records and Training in the Age of Big Data. In J. Percell , L. C. Sarin , P. T. Jaeger , J. C. Bertot (Eds.), "Re-Envisioning the MLS: Perspectives on the Future of Library and Information Science Education (Advances in Librarianship," Volume 44B, Emerald Publishing Limited, pp. 179-199. https://ai-collaboratory.net/wp-content/uploads/2020/10/Marciano-et-al-Archival-Records-and-Training-in-the-Age-of-Big-Data-final.pdf

[7] J. Wing. "Computational Thinking," Communications of the ACM, 49 (3) pp. 33-35, Mar 2006. https://dl.acm.org/doi/pdf/10.1145/1118178.1118215

[8] D. Weintrop, E. Beheshti, M. Horn, K. Orton, K. Jona, L. Trouille. & U. Wilensky. "Defining Computationl Thinking for Mathematics and Science Classrooms," Journal of Science Education and Technology, 25(1), pp. 127–147, 2016. https://ccl.northwestern.edu/2015/Weintrop%20et%20al.%20-%202015%20-%20Defining%20Computational%20Thinking%20for%20Mathematics%20an.pdf

[9] R. Maciano, W. Underwood, M. Hannaee, C. Mullane, A Singh, and Z. Tethong, "Automating the detection of personally identifiable information (PII) in Japanese-American WWII incarceration camp records," Proceedings of IEEE Big Data Conference 2018, CAS Workshop, Seattle, Washington, Dec 2018. https://ai-collaboratory.net/wp-content/uploads/2020/03/2.Marciano.pdf

[10] W. Underwood and R. Marciano, 2019. "Computational thinking in archival research and education," Proceedings of IEEE Big Data 2019, Computational Archival Science (CAS) Workshop #4, Los Angeles, Dec. 2019. https://ai-collaboratory.net/wp-content/uploads/2021/03/Underwood.pdf

[11] OpenAI. "GPT-4 Technical Report." 2023 https://arxiv.org/abs/2303.08774

[12] OpenAI. 'Introducing GPT-4o and more tools to ChatGPT free users." https://openai.com/index/gpt-4o-and-more-tools-to-chatgpt-free

[13] W. Underwood & J. Gage (2023) "Can GPT-4 Think Computationally about Digital Archival practices?" Computational Archival Science Workshop #8. https://ai-collaboratory.net/wp-ontent/uploads/2023/11/S01213_7999.pdf

[14] M. Breuss. "ChatGPT: Your personal Python coding mentor." Real Pyt5hon. https://realpython.com/chatgpt-coding-mentor-python/