Email Classification Model Prompt Iteration Documentation

1. Initial Prompts

The initial approach involved a simple prompt to categorize an email into one of five categories: complaint, inquiry, feedback, support_request, or other.

First Prompt (Initial Version):

```
prompt = f"""
    Categorize the following email into one of these categories:
    complaint, inquiry, feedback, support_request, other.
    Email:
    Subject: {email["subject"]}
    Body: {email["body"]}
    Respond with only the category.
"""
```

- **Purpose**: This prompt was intended to classify the email into one of the five predefined categories.
- **Result**: While this provided basic categorization, the accuracy was inconsistent as it didn't include examples or specific guidelines.

Second Prompt (Improved Version):

```
prompt = f"""
    You are an AI that classifies emails into one of the following categories:
    - complaint
    - inquiry
    - feedback
    - support_request
    - other

    Here is an email:
    Subject: {email["subject"]}
    Body: {email["body"]}

    Please return ONLY the category that best describes this email.
"""
```

- **Purpose:** The goal was to give the model more structured guidance by listing the categories explicitly.
- **Result:** Improved the classification slightly but still lacked fine-tuned accuracy for complex cases.

2. Iteration: Refining the Approach

To improve accuracy and ensure more reliable results, more specific examples and rules were added to guide the AI.

Third Prompt (Intermediate Version):

```
prompt = f"""
    You are an AI that classifies emails into one of the following categories:
    - complaint
    - inquiry
    - feedback
    - support_request
    - other

    Here is an email:
    Subject: {email["subject"]}
    Body: {email["body"]}

    Please return ONLY the category that best describes this email.
"""

try:
    response = self.client.chat.completions.create(
        model="gpt-4o-2024-11-20",
        messages=[{"role": "system", "content": "You are an AI assistant that classifies emails in complaint, inquiry, feedback, support_request or other."},
                  {"role": "user", "content": prompt}],
        max_tokens=10,
        temperature=0
    )
```

- **Purpose:** This was an intermediate refinement where the model was directed to classify emails into one of the five categories, still without specific examples or rules.
- **Result:** The classification process started becoming a bit more consistent but failed in edge cases where emails could belong to multiple categories or were unclear.

3. Final Prompt (Current Version)

In the final iteration, we made the most significant improvements by including specific examples, rules for handling uncertain or mixed cases, and constraints to reduce randomness.

Final Prompt:

```python
prompt = f"""
    You are an AI that classifies emails into one or more of the following categories:
    - complaint (e.g., "My product arrived broken, I want a refund.")
    - inquiry (e.g., "Does this software work on Mac OS?")
    - feedback (e.g., "Great customer service, I'm very satisfied!")
    - support_request (e.g., "I'm getting error 5123 while installing the app.")
    - other (e.g., "We'd like to explore a partnership opportunity.")

    Rules:
    1. If an email belongs to multiple categories, return other.
    2. If uncertain, return other.

    Here is an email:
    Subject: {email["subject"]}
    Body: {email["body"]}

    Please return ONLY the category that best describes this email.
"""

try:
    response = self.client.chat.completions.create(
        model="gpt-4o-2024-11-20",
        messages=[{"role": "system", "content": "You are an AI assistant that classifies emails in complaint, inquiry, feedback, support_request or other."},
                  {"role": "user", "content": prompt}],
        max_tokens=10,
        temperature=0
    )

```

- **Purpose:** The final prompt included specific examples for each category and introduced rules for handling emails that may belong to multiple categories or those that are unclear.
- **Result:** This prompt significantly improved classification accuracy, making the results much more consistent and accurate in identifying the correct category.

4. Results:

After implementing the final prompt with the added examples and rules, the model achieved 100% classification accuracy in a test run:

Before Fine-Tuning (Initial Accuracy of 80%):

The model classified emails and showed the following output with a misclassification in email 005

```
Processing completed:
  email_id  success   classification  response_sent
0      001     True       complaint          True
1      002     True         inquiry          True
2      003     True        feedback          True
3      004     True  support_request          True
4      005     True         inquiry          True

Classification Accuracy: 0.80
Misclassified email 005: predicted 'inquiry', actual 'other'
```

- **Accuracy**: 80%

- **Issue**: Misclassification occurred in **email 005** where the prediction was 'inquiry' instead of 'other'.

After Fine-Tuning (Achieved 100% Accuracy):

After applying the final prompt with detailed rules and examples, the model improved to 100% accuracy:

```
Processing completed:
  email_id  success   classification  response_sent
0      001     True        complaint           True
1      002     True          inquiry           True
2      003     True         feedback           True
3      004     True  support_request           True
4      005     True            other           True
5      006     True            other           True
6      007     True            other           True
7      008     True            other           True

Classification Accuracy: 1.00
```

- **Accuracy**: 100%
- **Outcome**: The model successfully classified all emails correctly with no misclassifications.

5. Problems Encountered

- **Issue 1: Inconsistent Results**
    - In the earlier iterations, the model often misclassified emails, particularly in edge cases where emails might belong to multiple categories or are unclear.
- **Issue 2: Uncertainty Handling**
    - The model didn't always know how to handle ambiguous or mixed emails, which led to poor results in some cases.

6. Improvements Made

- **Improvement 1: Fine-Tuning the Prompt**
    - The prompt was fine-tuned by adding examples, rules, and structured instructions for dealing with ambiguous or complex emails.

    - This reduced the randomness in results by providing clear guidelines and examples for the model.

- **Improvement 2: System and User Prompts**
    - System-level instructions were added to clarify the role of the AI in classifying the emails. The user prompt became more specific, helping the AI focus better on what was needed.
- **Improvement 3: Reduce Randomness**
    - By setting the temperature to 0 in the model call, randomness was reduced, leading to more consistent and reliable outputs.
- **Outcome:**
    - The model reached **100% accuracy** in classification after iterating over the prompts.
    - Failures occurred mainly for very ambiguous or unclear emails, where fine-tuning was required.

7. Further Fine-Tuning

- **Objective**: Fine-tune the model to handle specific cases better, improve accuracy, and reduce the number of misclassifications.
- **Approach**:
    - Implement more explicit instructions for complex scenarios.
    - Evaluate misclassifications and retrain the model to handle similar cases better.