Every day, professionals wade through hundreds of emails, from urgent client requests to promotional offers. It's like trying to find important messages in a digital ocean. But AI can help you stay afloat by automatically sorting emails to highlight what matters most.

You've been asked to build an intelligent email assistant using Llama, to help users automatically classify their incoming emails. Your system will identify which emails need immediate attention, which are regular updates, and which are promotions that can wait or be archived.

## The Data

You'll work with a dataset of various email examples, ranging from urgent business communications to promotional offers. Here's a peek at what you'll be working with:

### email_categories_data.csv

| Column | Description |
|---|---|
| email_id | A unique identifier for each email in the dataset. |
| email_content | The full email text including subject line and body. Each email follows a format of "Subject" followed by the message content on a new line. |
| expected_category | The correct classification of the email: `Priority`, `Updates`, or `Promotions`. This will be used to validate your model's performance. |

```
# Run the following cells first
# Install necessary packages
!pip install llama-cpp-python==0.2.82 -q -q -q
```

```
# Download the model

model_path = "!wget -q https://huggingface.co/TheBloke/TinyLlama-1.1B-Chat-v0.3-GGUF/resolve/main/tinyllama-1.1b-chat-v0.3.Q4_K_M.gguf?
download=true -O model.gguf"
```

```
# Import required libraries
import pandas as pd
from llama_cpp import Llama
```

```
# Load the email dataset
emails_df = pd.read_csv('data/email_categories_data.csv')
# Display the first few rows of our dataset
print("Preview of our email dataset:")
emails_df.head(2)
```

```
Preview of our email dataset:
```

| index | email_id | email_content | expected_category |
|---|---|---|---|
| 0 | 1 | Urgent: Server Maintenance Required\nOur main server needs immediate maintenance due to critical e... | Priority |
| 1 | 2 | 50% Off Spring Collection!\nDon't miss our biggest sale of the season! All spring items half off. Limited t... | Promotions |

Rows: 2

```
# Start coding here
# Use as many cells as you need

# Download the model file
!wget -q https://huggingface.co/TheBloke/TinyLlama-1.1B-Chat-v0.3-GGUF/resolve/main/tinyllama-1.1b-chat-v0.3.Q4_K_M.gguf?download=true -O
model.gguf

# Define the model path
model_path = "model.gguf"

# Initialize the Llama model
llm = Llama(model_path=model_path)

prompt = """ You classify emails into Priority, Updates, or Promotions.

Example 1:
Urgent: Password Reset Required
Your account security requires immediate attention. Please reset your password within 24 hours.
Response: Priority

Example 2:
Special Offer - 50% Off Everything!
Don't miss our biggest sale of the year. Everything must go!
Response: Promotions
```

```
Example 3:
Canceled Event - Team Meeting
This event has been canceled and removed from your calendar.
Response: Updates

Example 4:

"""
```

```
llama_model_loader: loaded meta data with 20 key-value pairs and 201 tensors from model.gguf (version GGUF V2)
llama_model_loader: Dumping metadata keys/values. Note: KV overrides do not apply in this output.
llama_model_loader: - kv   0:                       general.architecture str              = llama
llama_model_loader: - kv   1:                               general.name str              = py007_tinyllama-1.1b-chat-v0.3
llama_model_loader: - kv   2:                       llama.context_length u32              = 2048
llama_model_loader: - kv   3:                     llama.embedding_length u32              = 2048
llama_model_loader: - kv   4:                          llama.block_count u32              = 22
llama_model_loader: - kv   5:                  llama.feed_forward_length u32              = 5632
llama_model_loader: - kv   6:                 llama.rope.dimension_count u32              = 64
llama_model_loader: - kv   7:                 llama.attention.head_count u32              = 32
llama_model_loader: - kv   8:              llama.attention.head_count_kv u32              = 4
llama_model_loader: - kv   9:     llama.attention.layer_norm_rms_epsilon f32              = 0.000010
llama_model_loader: - kv  10:                       llama.rope.freq_base f32              = 10000.000000
llama_model_loader: - kv  11:                          general.file_type u32              = 15
llama_model_loader: - kv  12:                       tokenizer.ggml.model str              = llama
llama_model_loader: - kv  13:                      tokenizer.ggml.tokens arr[str,32003]   = ["<unk>", "<s>", "</s>", "<0x00>", "<...
llama_model_loader: - kv  14:                      tokenizer.ggml.scores arr[f32,32003]   = [0.000000, 0.000000, 0.000000, 0.0000...
llama_model_loader: - kv  15:                  tokenizer.ggml.token_type arr[i32,32003]   = [2, 3, 3, 6, 6, 6, 6, 6, 6, 6, 6, 6, ...
llama_model_loader: - kv  16:                tokenizer.ggml.bos_token_id u32              = 1
llama_model_loader: - kv  17:                tokenizer.ggml.eos_token_id u32              = 2
llama_model_loader: - kv  18:            tokenizer.ggml.unknown_token_id u32              = 0
llama_model_loader: - kv  19:               general.quantization_version u32              = 2
llama_model_loader: - type  f32:   45 tensors
llama_model_loader: - type q4_K:  135 tensors
llama_model_loader: - type q6_K:   21 tensors
llm_load_vocab: special tokens cache size = 262
llm_load_vocab: token to piece cache size = 0.1684 MB
llm_load_print_meta: format           = GGUF V2
llm_load_print_meta: arch             = llama
```

```python
# Function to process messages and return classifications
def process_message(llm, message, prompt):
    """Process a message and return the response"""
    input_prompt = f"{prompt} {message}"
    response = llm(
        input_prompt,
        max_tokens=5,
        temperature=0,
        stop=["Q:", "\n"],
    )

    return response['choices'][0]['text'].strip()
```

```python
test_emails = emails_df.head(2)

results = []

for idx, row in test_emails.iterrows():
    email_content = row['email_content']
    expected_category = row['expected_category']

    result = process_message(llm, email_content, prompt)

    results.append({
        'email_content': email_content,
        'expected_category': expected_category,
        'model_output': result
    })
```

```
llama_print_timings:        load time =    9948.63 ms
llama_print_timings:      sample time =       0.16 ms /     1 runs   (    0.16 ms per token,  6289.31 tokens per second)
llama_print_timings: prompt eval time =    9948.39 ms /   167 tokens (   59.57 ms per token,    16.79 tokens per second)
llama_print_timings:        eval time =       0.00 ms /     1 runs   (    0.00 ms per token,      inf tokens per second)
llama_print_timings:       total time =    9949.45 ms /   168 tokens
Llama.generate: prefix-match hit

llama_print_timings:        load time =    9948.63 ms
llama_print_timings:      sample time =       0.16 ms /     1 runs   (    0.16 ms per token,  6250.00 tokens per second)
llama_print_timings: prompt eval time =    2900.01 ms /    30 tokens (   96.67 ms per token,    10.34 tokens per second)
llama_print_timings:        eval time =       0.00 ms /     1 runs   (    0.00 ms per token,      inf tokens per second)
llama_print_timings:       total time =    2900.57 ms /    31 tokens
```

```
# Create a DataFrame with results
results_df = pd.DataFrame(results)

result1 = "Priority"
result2 = "Promotions"
```

```
results_df.head(2)
```

|  | email_content | expected_cat... | mod... |
|---|---|---|---|
| 0 | Urgent: Server Maintenance Required\nOur ... | Priority |  |
| 1 | 50% Off Spring Collection!\nDon't miss our b... | Promotions |  |

Rows: 2