17-630 Prompt Engineering

Assignment: Prompt Composition

Complex tasks can require composing multiple prompts into a chain in which the output from one prompt is used to setup the input to a subsequent prompt. In this respect, error must be handled locally so that it does not propagate downstream. In this assignment, students will read e-mails to check for meeting invitations and call an API to check if a meeting time is available for scheduling. The chain will consist of three types of prompts: (1) a classification task to classify e-mails into one of several types; (2) an extraction task to extract the meeting dates and times; and (3) a function calling task to format the meeting request into a function call.

Learning Objectives

- Learn to orchestrate and coordinate the I/O across multiple prompts.
- Learn to perform function calling.

Assignment Deliverables

This is an individual assignment. To complete the assignment, perform the following steps: (1) read the training_data.json file and develop a prompt to classify e-mail messages based on meeting message type (see Further guidance) and report your accuracy; (2) for each meeting-related message, extract any datetime information; (3) for each "meeting invitation" message with extracted datetime information, generate a function call to check the employee's calendar availability. Save the function calls to a file "function-calls.json"

Develop a Jupyter Notebook that includes sections for each of the three above tasks. Students are encouraged to use the notebook provided in the assignment package.

Submit a single ZIP archive with the following files and exact filenames:

- notebook.ipynb – your Jupyter Notebook containing your code used to conduct your experiment. Please document major sections of your notebook.
- function-calls.json – contains the formatted functional
- questions.txt – Answers to the questions provided in the package.

Further Guidance

- The training data consists of a list of dictionaries, each containing the follows keys: "email", which is a l-separated list of the sender, subject, date and message body; "label", which is the meeting class; "dates", which is the list of meeting dates described in the message, if any; and "index", which is the unique index assigned to each message.
- Instruction-tuned models are known to be well calibrated with multiple choice questions. Classify the e-mails using a multiple-choice question. The training data includes labels for the following classes:
  - *Meeting invitation* - a message inviting the recipient to attend a meeting

    - *Meeting reminder* - a message reminding the recipient about an upcoming meeting
    - *Meeting update* - a message notifying the recipient that the date or time for a previously scheduled meeting has changed
    - *Meeting cancellation* - a message notifying the recipient that a previously scheduled meeting has been cancelled
- Calculate accuracy for the classification task by counting true positives, only if the predicted message label matches the exact class above.
- In the training data, the meeting dates are encoded in MM-DD-YYYY format and the meeting times are encoded using a 24-hour clock, e.g., 13:30-14:00 describes a meeting from 1:30-2:00 PM. Meeting messages may be vague or incomplete. If the message does not contain the meeting year or meeting time, encode the missing digits with zeros, e.g., “09-19-0000” or “00:00-00:00”. If a message contains more than one meeting date and time, the training data will contain this information in a semi-colon separated list.
- Calculate precision and recall for the meeting date and time extraction task by counting true positives for each date-time pair that matches an expected date-time pair. The timezone is not included in the ground truth and thus does not count toward precision and recall.
- The function to be called is named “check_availability” and accepts the following parameters: month, day, year, hour, minute and timezone. If the timezone is not present in the message, assume eastern standard time. Use the OpenAI tools parameter to format the function call, by asking a question about whether your user can attend a meeting during the given date and time. Save the responses in the file “function-calls.json” as a list of dictionaries with the following format: {“index”: [index], “call”: [response]} where [index] is the index of the meeting message, and [response] is the response from the OpenAI function call prompt. There may be more than one call per message index, if there is more than one meeting time listed. Each call should map to only one dictionary in the function-calls.json file.

Evaluation Criteria

- Correctness of the implementation of the experiment and JSON file.
- Thoughtfulness of the answers to the questions in questions.txt, including examples from your work to illustrate your answers.