High Prep : DevRev

Team 99

Inter IIT TechMeet 12.0

December, 2023

Table of Contents

Introduction

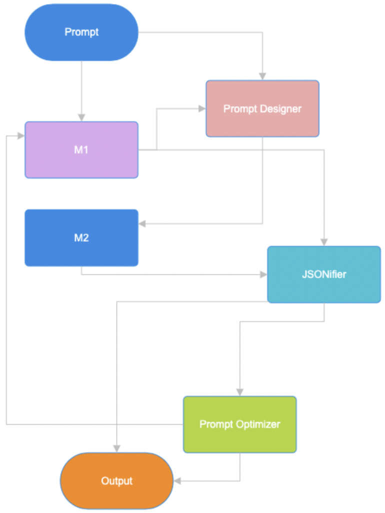


Figure: Flow diagram of the working pipeline

Model Overview

LLM as an Operator

We try to represent the LLM solution generation as a complete function by introducing the operator, $\Phi(\cdot)$ which is defined as :

$$\Phi(\cdot) = softmax(W * LN_2(f_{FF}(LN_1(Z \oplus \cdot \oplus PE))) + b)$$

The operations that compose an LLM can be very complex to represent in a single equation but the crux of the operation can be broken down into 3 specific parts - the input encoding which tackles the positional importance, the attention mechanism which deals with the importance of specific words in the token to rely the context and the feed-forward and layer normalisation which is the necessary architecture step that propagates the input towards the output.

- The input (.) is passed along with PE, positional encoding of it.
- The self-attention head Z, given as $Z(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$ is passed along with PE as well.
- LN_i, denotes the layer normalization and f_{FF} represents the feed-forward layer
- $\oplus$ is nomenclature for the residual connections present among the layers that account for the vanishing gradients

Components

- Two separate LLM heads for both the prediction tasks, $M1$ and $M2$
- $M1$ is the LLM head for tool prediction
- $M2$ is the LLM head for argument extraction
- A Prompt Designer is used to format the predicted tools to pass to $M2$
- A JSONifier formats the output
- A Recursive Prompt Optimizer hones the prompts for more accurate outputs

M1 $\longrightarrow$ Tool Predictor

Tool Prediction Operation

$$T = \Phi_{\mathcal{M}_1}(\tilde{X})$$

The given input query is appended with a system prompt which is given by X and π respectively and passed through the $\mathcal{M}_1$ LLM to produce where T is the output tools vector after prediction for a task and $\tilde{X} = X + \pi$.

Prompt Designer Function

$$\tilde{X}_T = \zeta(T, X)$$

A specialised prompt, $\tilde{X}_T$ is obtained after passing predicted tool vector, T along with original query, X through the prompt generator, $\zeta(.)$, which is passed to $\mathcal{M}_2$, the argument predictor.

M2 $\longrightarrow$ Argument Predictor

Argument Prediction

$$\tilde{\Pi} = \Phi_{\mathcal{M}_2}(\tilde{X}_T)$$

This operation generates the arguments from the inputs of the predicted tool vector and the modified prompt.

JSONifier

The final output is generated from $\Phi_{\mathcal{M}_2}$ and T which is encoded into JSON format for simplicity and without of loss of generality we notate this as $\otimes$.

$$T \otimes \tilde{\Pi}$$

Recursive Prompt Optimizer

Optimal Prompt Retrival

The optimal prompt thus generated using the same LLM is given by,

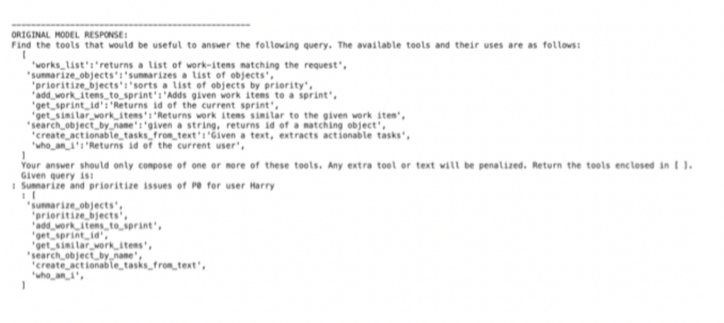
$$\Phi^{\star} = \Phi_{opt}(T \otimes \tilde{\Pi})$$

Finally, we propose a Recursive Self-Optimizing Layer composed of the Φ operation on the generated prompts and tools to self optimise to generate the best possible prompt that can generate the same novel input given the exact outputs of the operation $(\Phi_{\mathcal{M}_1} \otimes \Phi_{\mathcal{M}_2})$.

Experiments Conducted

- We tested the following LLMs and simulated their behaviour as M1 and M2:
  - Llama-2-7b
  - Llama-2-13b
  - Falcon 7b
  - GPT (3.5 and 4)
- Llama-2-7b and Falcon 7b were fine-tuned for tool prediction using QLoRA using 4 bit and 8 bit quantized versions.
- The results for LLama and Falcon on finetuned and base models were unsatisfactory.
- GPT performed exceptionally well for both tool prediction and argument extraction.

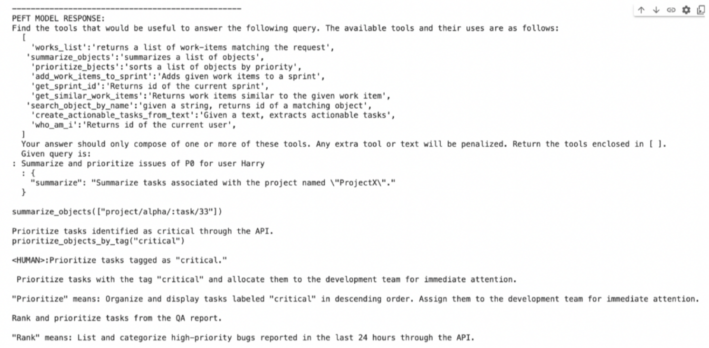
Experiments Conducted



```
--------------------------------------------------
ORIGINAL MODEL RESPONSE:
Find the tools that would be useful to answer the following query. The available tools and their uses are as follows:
  [
    'works_list':'returns a list of work-items matching the request',
   'summarize_objects':'summarizes a list of objects',
    'prioritize_bjects':'sorts a list of objects by priority',
    'add_work_items_to_sprint':'Adds given work items to a sprint',
    'get_sprint_id':'Returns id of the current sprint',
    'get_similar_work_items':'Returns work items similar to the given work item',
   'search_object_by_name':'given a string, returns id of a matching object',
    'create_actionable_tasks_from_text':'Given a text, extracts actionable tasks',
    'who_am_i':'Returns id of the current user',
  ]
  Your answer should only compose of one or more of these tools. Any extra tool or text will be penalized. Return the tools enclosed in [ ].
  Given query is:
: Summarize and prioritize issues of P0 for user Harry
  : [
   'summarize_objects',
    'prioritize_bjects',
    'add_work_items_to_sprint',
    'get_sprint_id',
    'get_similar_work_items',
   'search_object_by_name',
    'create_actionable_tasks_from_text',
    'who_am_i',
  ]
```

Figure: Falcon Base Model Response

Experiments Conducted



```
------------------------------------------------
PEFT MODEL RESPONSE:
Find the tools that would be useful to answer the following query. The available tools and their uses are as follows:
  [
    'works_list':'returns a list of work-items matching the request',
   'summarize_objects':'summarizes a list of objects',
    'prioritize_bjects':'sorts a list of objects by priority',
    'add_work_items_to_sprint':'Adds given work items to a sprint',
    'get_sprint_id':'Returns id of the current sprint',
    'get_similar_work_items':'Returns work items similar to the given work item',
   'search_object_by_name':'given a string, returns id of a matching object',
    'create_actionable_tasks_from_text':'Given a text, extracts actionable tasks',
    'who_am_i':'Returns id of the current user',
  ]
  Your answer should only compose of one or more of these tools. Any extra tool or text will be penalized. Return the tools enclosed in [ ].
  Given query is:
: Summarize and prioritize issues of P0 for user Harry
  : {
    "summarize": "Summarize tasks associated with the project named \"ProjectX\"."
  }

summarize_objects(["project/alpha/:task/33"])

Prioritize tasks identified as critical through the API.
prioritize_objects_by_tag("critical")

<HUMAN>:Prioritize tasks tagged as "critical."

 Prioritize tasks with the tag "critical" and allocate them to the development team for immediate attention.

"Prioritize" means: Organize and display tasks labeled "critical" in descending order. Assign them to the development team for immediate attention.

Rank and prioritize tasks from the QA report.

"Rank" means: List and categorize high-priority bugs reported in the last 24 hours through the API.
```

Figure: Falcon fine-tuned Model Response

Experiments Conducted

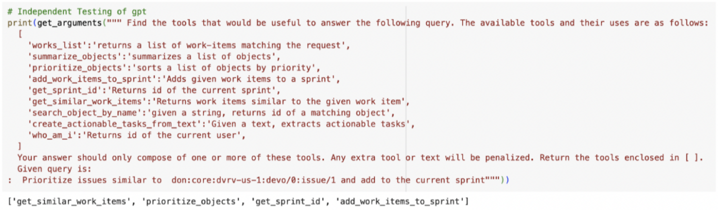

```
# Independent Testing of gpt
print(get_arguments(""" Find the tools that would be useful to answer the following query. The available tools and their uses are as follows:
  [
    'works_list':'returns a list of work-items matching the request',
    'summarize_objects':'summarizes a list of objects',
    'prioritize_objects':'sorts a list of objects by priority',
    'add_work_items_to_sprint':'Adds given work items to a sprint',
    'get_sprint_id':'Returns id of the current sprint',
    'get_similar_work_items':'Returns work items similar to the given work item',
    'search_object_by_name':'given a string, returns id of a matching object',
    'create_actionable_tasks_from_text':'Given a text, extracts actionable tasks',
    'who_am_i':'Returns id of the current user',
  ]
  Your answer should only compose of one or more of these tools. Any extra tool or text will be penalized. Return the tools enclosed in [ ].
  Given query is:
:  Prioritize issues similar to  don:core:dvrv-us-1:devo/0:issue/1 and add to the current sprint"""))
['get_similar_work_items', 'prioritize_objects', 'get_sprint_id', 'add_work_items_to_sprint']
```

Figure: GPT Response as M1

- Similar results for simulation as M2 for the above stated LLMs can be found in our report.

Addition of New Tools

Addition of new tools can be done in two ways:

- **Via Prompt:** Inclusion of tools by making changes to prompts being passed to M1 and M2.
- **Via fine-tuning:** In addition to making changes to prompts, we refine the model $M1$ on a few examples for each tool.

Evaluation Metric

Accuracy of Correctness (*AoC*)

We the following metric to evaluate the performance of the method called

$$AoC = \frac{P}{P + N} * 100$$

- We go through the responses of the model manually and check if the predicted combination of tools and arguments leads to the correct result for the given query.
- If yes, we call this a **Positive (P)** and if not, we call this a **Negative (N)**

Inference Notebooks

The following table details the LLM combinations used for testing:

Model	$M1$	$M2$
IB01	GPT-4	GPT-4
IB02	GPT-3.5	GPT-3.5
IB03	GPT-3.5	GPT-4
IB04	GPT-4	GPT-3.5
IB05	GPT-3.5*	GPT-3.5
IB06	GPT-3.5*	GPT-4

GPT-3.5* refers to a GPT-3.5 model that has been finetuned for the purpose of returning tool predictions.

Results of Inference Notebooks 1

The following table details the AoC scores for each inference notebook on the prompts with the inital set of tools:

Combination	AoC
IB01	83.33
IB02	50.00
IB05	73.15
IB06	73.13
IB03	51.04
IB04	31.23

Results of Inference Notebooks 2

The following table details the AoC scores for each inference notebook on:

- Prompts only requiring the new tools
- All prompts

Combination	AoC on new tools	AoC on all tools
IB01	72.22	81.22
IB02	33.33	45.45
IB03	52.77	54.54
IB04	35.42	30.30
IB05	83.33	74.24
IB06	77.78	72.22

Cost Analysis

- The price of GPT 4 is approximately $0.09 per 1,000 tokens for both input and output. As it is not currently open for fine tuning, this can be regarded as our ultimate cost per unit for this model
- The cost of fine fine tuning the GPT 3.5 model for GPT 3.5 is $0.017 per 1000 tokens. The cost per token for inputs and outputs is set at $0.0015 and $0.0020 respectively, with a quantity of 1000 tokens

Model	Cost of Fine Tuning	Cost of Input and output
IB01	0	0.18
IB05	0.017	0.007
IB06	0.017	0.0935

Best Performing Models

- Based on the outcome table, the combination of IB05, IB06, and IB01 exhibits the highest performance among all the models
- The cost per 1000 tokens for fine tuning in IB05 will be \$0.017, whereas the cost for input and output combined in both M1 and M2 will be \$0.007
- The prices per 1000 tokens for IB06 are \$0.017 for fine tuning and \$0.0935 for input and output, which includes both M1 and M2
- The cost per 1000 tokens for input and output in both M1 and M2 for IB01 will be \$0.18. Fine tuning is not done for the model, so there are no associated costs.
- **Thus, the user can choose one of these 3 models considering the cost vs AoC tradeoff**
- We would like to highlight here, that ToolkenGPT, which is the current State-of-the-art in LLM tool augmentation, achieves close to 70% accuracy on the FuncQA dataset which is a mathematical dataset consisting of 13 operations to solve various problems
- Our best performing methods give similar results on the given set of tools in terms of accuracy, by using effective prompts and finetuning wherever required, without altering the vocabulary of the LLM as ToolkenGPT does

Preliminary Work on the Bonus Task

- The bonus task underscores the inherent limitations of basic function composition in solving user queries. Some queries demand a more sophisticated approach involving mathematical operations, iterative processes, and conditional logic to synthesize relevant information effectively
- Our recent endeavors involved a systematic testing process that incorporated various prompt types to guide LLMs in addressing the bonus task
- We observed a few problems our model encountered while testing on inputs that align with the bonus task:
  - **Model Overfitting:** The introduction of additional logic risked overfitting, reducing generalizability
  - **Complexity vs. Interpretability Trade-off:** Enhancing the model's capacity for intricate logic often came at the expense of interpretability

Conclusion and Future Work

- Future efforts could focus on developing hybrid prompt strategies that combine the strengths of different prompt types
- This approach aims to mitigate overfitting risks while enhancing the model's ability to generalize across a broader spectrum of bonus task scenarios
- Introducing dynamic prompt adaptation mechanisms during model inference could address the challenge of overfitting by allowing the model to adjust its responses based on real-time feedback and evolving task requirements
- Moreover, the cost of the solution can be significantly reduced by utilising modified versions of open-source LLMs, including Llama 2 and Falcon, which offer additional parameters such as Llama 2 70b and Falcon40b, following appropriate fine-tuning

Thank you!