Investigating Optimal AMA Prompting Strategies for Small Foundation Models

Aayush Agrawal
Computer Science
aayush2k@stanford.edu

Andrew Hojel
Computer Science
ahojel@stanford.edu

Oscar O'Rahilly
Computer Science
oscarfco@stanford.edu

Abstract

In this paper, we build on the ASK ME ANYTHING (AMA) prompting strategy developed by Arora et al. [1]. This strategy breaks difficult prompts down into prompt chains, which consist of multiple prompt steps where outputs from one step are passed into the context of the next step. The strategy then takes the outputs from multiple prompt chains and uses Majority Vote or Weak Supervision to make a final prediction. We wanted to investigate how changing the hyper parameters of AMA would affect overall performance. Throughout this paper we focused on the SuperGLUE WSC task, which evaluates a machine's ability to understand the meaning of a sentence by focusing on relationships between words in the sentence. We ran an initial experiment substituting values from the prompt chain of a 6B parameter model (GPT-J-6B [9]) into the prompt chain of a 1.3B parameter model (GPT-Neo-1.3B [2]) and found that better results within the prompt chain improve downstream results by up to 9.6%. The rest of the paper focused on improving the performance of the 1.3B parameter model. We experimented with changing the number of in-context examples for prompts within prompt chains, varying the number of prompt chains, varying the outputs of in-context examples, and using ChatGPT [8] to neurally generate in-context examples. Our findings could be relatively counter intuitive in the sense that there were not typically clear trends in the effects of varying the number of in-context examples or number of prompt chains. In addition, we observed that shuffling the outputs of the in-context examples improved performance. Using the ideal hyper parameters in terms of number of shots per chain and number of prompt chains results in a 14.1% improvement over the original majority vote results when using GPT-Neo-1.3B. The results from neural in-context example generation, which we believe we are the first to experiment with, were very promising. Some neurally generated in-context examples were able to outperform the original human-written prompt chain accuracy by 4.7%. Overall, we can conclude that tuning the hyper parameters when using the AMA prompting framework can be un-intuitive but can yield significant increases in overall performance. In conjunction with the fact that neural prompt chains generated by more powerful models can achieve results similar to the human-written prompt chains used in the original AMA paper, this paper motivates the development of a more automated strategy for experimenting with the hyper parameters of AMA. This tuning could avoid the need for significant human intervention by utilizing larger foundation models to help generate in-context examples.

CS330 (Deep Multi-Task and Meta Learning) Final Project, Fall 2022

1 Introduction

The objective of this project is to improve the performance of "smaller" (<10B parameter) foundation models (a large language model (LLM) trained on a massive dataset that can be adapted to a wide range of downstream tasks). Currently, the top performing foundation models are extremely large (100Bs of parameters), which results in higher latency, significant cost, and difficulty running these models locally. By improving the performance of smaller foundation models, we could provide an alternative that is lower latency, lower cost, and could potentially be run locally. The value of being able to run a foundation model locally is that it is more privacy preserving (don't send model prompts to a server), can be done offline, and is potentially faster given that it does not rely on networking bottlenecks.

To approach the problem of improving the performance of smaller foundation models, we investigated ASK ME ANYTHING (AMA) prompting developed by Arora et al [1]. AMA proposes a prompting strategy (discussed in more detail in Related Work 2 and Technical Approach 3.3) that can significantly improve the performance of small foundation models. Given the potential to deploy AMA for a variety of tasks, we investigated different approaches to tuning AMA prompting, including tuning hyperparameters and attempting to use neurally generated in-context examples generated by larger models. AMA was published this year, so there is no known work further investigating AMA prompting.

In addition to investigating strategies for improving AMA performance via tuning, we are also interested in creating a more automated approach for deploying AMA for a given task. Currently, multiple components of AMA need to be manually written or configured. We are excited about the potential to search different configuration of AMA automatically.

2 Related Works

The inspiration for this project comes from work by Arora et al. [1] titled ASK ME ANYTHING (AMA) prompting, which investigates the affect of producing multiple effective, yet imperfect, prompts recursively using the foundation model itself, followed by majority vote or weak supervision to aggregate the results of these various prompts into a final prediction given an input. Arora et al. [1] find that open-ended question-answering prompts (QA) outperform more closed ended questions that result in a yes or no; therefore, the strategy converts inputs into an open-ended QA style when generating synthetic prompts. The strategy helps avoid the need to create a "perfect prompt" to use a foundation model for a new task, replaced by a strategy for aggregating a batch of imperfect synthetic prompts for a more robust prompting strategy. Arora et al. [1] found that AMA enables the open-source GPT-J-6B to exceed the performance of few-shot GPT3-175B on 15 of 20 popular benchmarks.

Another relevant prior work is Decomposed Prompting by Khot et al. [5], which proposes decomposing complex tasks into sub-tasks, which are delegated to prompting-based LLMs to solve the sub-tasks. This process optimizes each prompt for the given sub-task and allows LLMs to approach more difficult problems. Khot et al. [5] find that Decomposed Prompting can improve the performance of few-shot GPT3-175B on various tasks.

This research is interesting when approaching tasks that require complex chain of reasoning, but for simpler tasks such as reasoning and extracting information from a context, the Decomposed Prompt strategy may provide less performance benefit.

In the Selection-Inference framework, Creswell et al. [3] propose a strategy of breaking down questions that require multi-step reasoning into selection (selection of a reasonable subset of information for the next inference step) and inference (infers a new piece of intermediate evidence using the distilled information from the selection step). Creswell et al. [3] find that a 7B parameter model using 5-shot in-context examples can outperform a 280B parameter model on 10 logical reasoning tasks without finetuning. Crewswell et al. [3] find that an important next step is to build on the framework and create processes to develop Selection-Inference in a more automated fashion.

The three different prompt decomposition approaches above make use of a recursive prompt decomposition strategy to significantly improve model performance (with AMA and Selection-Inference targeting small <10B parameter models). There exists empty space in research regarding how one can

take these strategies and apply them to new problems, properly tune hyperparameters, and adequately search potential recursive prompts. This motivates our desire in this paper to explore AMA more deeply.

In addition to further investigating recursive prompting strategies, in this paper we explore perturbing outputs of in-context examples and the use of larger foundation models to aid in the generation of in-context examples. Min et al. [7] investigate how in-context learning works and interpret it through a lens of bayesian inference. An interesting conclusion from their exploration was that varying the outputs of in-context examples within the distribution of possible in-context examples had minimal impact on overall performance. In terms of neurally generating in-context examples, Wu et al. [10] explored the creation of interfaces the aid in the development and iteration of prompt chaining. However, the interface developed by Wu et al. [10] did not explore the use of foundation models to recursively aid in any portions of the development. We were unable to find any other research into generation of in-context examples using foundation models.

3 Approach

3.1 Dataset

For this paper, we decided to focus on the SuperGLUE Winograd Schema Diagnostic (super_glue_wsc) task proposed by Levesque et al. [6], which is one of the 20 tasks investigated by Arora et al. [1] in the AMA paper.

SuperGLUE WSC is a set of tasks aimed at evaluating a machine's ability to understand the underlying meaning of a sentence by focusing on the relationships between the words in the sentence.

In the Winograd Schema diagnostic proposed by Levesque et al. [6], each of the questions consists of four primary features:

1. Two parties are mentioned in a sentence by noun phrases. The parties can be males, females, inanimate objects, or groups of people and objects.
2. A pronoun or possessive adjective is used in the sentence references one of the parties, but it is also properly formatted to reference the second party.
3. The questions requires determining the referent of the pronoun or possessive adjective. The two possible answers refer to the two parties mentioend in the sentence.
4. There is a word (called the *special word*) that appears in the sentence and possibly the question. It can be replaced by another word, however everything still makes sense but the answer changes.

An example Winograd Schema question is the following:

```
The trophy would not fit in the brown suitcase because it
was too big. What was too big?
    Answer 0: the trophy
    Answer 1: the suitcase
```

To elaborate on the fourth feature of WSC quetsions, we can see in the following example how changing the *special word* from `feared` to `advocated` changes the response to the question.

```
The city councilmen refused the demonstrators a permit because
they [feared/advocated] violence.
```

If the word is `feared`, then `they` most likely refers to the city council. However, if the word is changed to `advocated`, then `they` likely refers to the demonstrators.

Some unique / challenging characteristics of WSC is that the set of question can be:

- Easily disambiguated by a human reader (it is essentially intuitive and there is no ambiguity)
- Not solvable by simple techniques such as selectional restrictions
- Google-proof, in the sense that there is no statistical test over the text corpora that can disambiguate between the two parties mentioned

3.2 Models & Hardware

Throughout this paper, we use two different models with their base parameters (no finetuning is done). The two models are GPT-Neo-1.3B and GPT-J-6B. Both models were trained on the Pile, a large curated dataset created by EleutherAI for the purpose of training language models proposed by Gao et al. [4].

GPT-Neo-1.3B, proposed by Black et al. [2], is a 1.3B parameter model that was trained for 380 billion tokens over 362,000 steps. It was trained as a masked autoregressive language model using cross-entropy loss.

GPT-J-6B, proposed by Wang et al. [9], is a 6B parameter model trained for 402 billion tokens over 383,500 steps. It was trained as an autoregressive language model, using cross-entropy loss to maximize the likelihood of predicting the next token correctly during self-supervised learning.

We used Nvidia Tesla T4 GPUs to run all experiments. To run the experiments we used a `temperature = 0` so that the results would be deterministic and reproducible. In addition, we cached responses from the models for a given prompt to decrease the amount of compute needed to run all of the experiments.

3.3 Technical Approach

Throughout this paper, we do not perform any finetuning of either GPT-J-6B or GPT-Neo-1.3B. The primary contributions involve a range of experiments to the prompt chains used by AMA for recursive prompt generation. To better understand what this means, in Figure 1 we demonstrate what a typical AMA prompt chain looks like.

```
Extract the phrase containing the pronoun.

Passage: Jeff gave his son money because "he"
         wanted to buy lunch.
Extract: phrase containing "he": "he"
         wanted to buy lunch

...

Passage: The large ball crashed through the
         table because "it" was made of
         styrofoam.
Extract: phrase containing "it":
(model output #1) was made of styrofoam
```

(a) Extraction Step of Prompt Chain

```
Rewrite the input as a question.

Input: it was made of glass
Question: What was made of glass?

...

Input: (model output #1) was made of styrofoam
Question:
(model output #2) What was made of styrofoam?
```

(b) Question Step of Prompt Chain

```
Answer the question.

Passage: Mrs. Jenna told Fred she loved him.
Question: Who loved him?
Answer: Mrs. Jenna

 ...

Passage: The large ball crashed through the
         table because it was made of
         styrofoam.
Question: (model output #2) What was made of
          styrofoam?
Answer:
(model output #3) the table (True)
```

(c) Answer Step of Prompt Chain

Figure 1: Example of AMA Prompt Chain for SuperGLUE WSC

We can see in Figure 1 how outputs are recursively passed from each prompt in the chain to the next prompt. The AMA framework uses multiple prompt chains with varying in-context examples.

4 Experiments & Results

4.1 Substitution

The first experiment we ran was substituting GPT-Neo-1.3B extractions (`model output #1`) and questions (`model output #2`) into the prompt chains for GPT-J-6B (and vice versa).

This strategy would not be used in practice because with access to GPT-J-6B you would likely use it for every step of the prompt chain given that it has significantly better performance. However, we were interested in understanding how subsituting better or worse intermediary model outputs would affect overall performance of AMA to understand the difference between quality of prompt chain and overall model capability. In addition this could lay the ground work for teacher / student finetuning of smaller models using outputs of larger to create custom models for different portions of prompt chains. The results of the substition experiment can be found in Table 1

Model	Substitution	MV Accuracy	AAB Accuracy
GPT-Neo-1.3B	original	0.615	0.622
GPT-Neo-1.3B	extraction for GPT-J-6B	0.673	0.644
GPT-Neo-1.3B	question for GPT-J-6B	0.673	0.647
GPT-J-6B	original	**0.779**	**0.753**
GPT-J-6B	extraction from GPT-Neo-1.3B	0.731	0.689
GPT-J-6B	question from GPT-Neo-1.3B	0.645	0.663

Table 1: Results from substituting results between GPT-J-6B and GPT-Neo-1.3B using default AMA hyperparameters. MV stands for Majority Vote and AAB stands for Average Across Boost.

4.2 Varying Number of Shots within Prompt Chains

For this experiment, we varied the number of in-context examples per prompt in the prompt chain referred to in Figure 1. The default AMA has 3-shot in-context examples for extraction, 7-shot in-context examples for question generation, and 3-shot in-context examples for answer generation. The same number of in-context prompts are used across chain.

The results for varying the number of in-context examples per prompt can be found in Table 2.

K per Chain	MV Acc	MV Precision	MV Recall	AAB Acc	AAB Precision	AAB Recall
3,7,3 (original)	0.615	0.571	0.563	0.622	0.583	0.575
1,7,3	**0.673**	**0.644**	**0.636**	**0.641**	**0.607**	**0.600**
5,7,3	0.635	0.600	0.595	0.631	0.595	0.589
3,4,3	0.596	0.548	0.542	0.583	0.533	0.528
3,10,3	0.577	0.530	0.527	0.612	0.569	0.562
3,7,1	0.635	0.590	0.573	0.638	0.601	0.581
3,7,5	0.596	0.556	0.553	0.590	0.551	0.547

Table 2: Results from varying the number of shots per prompt in the chain where (a,b,c) is (k for extraction, k for question, k for answer) with default AMA hyperparameters. MV stands for Majority Vote and AAB stands for Average Across Boost.

4.3 Varying Number of Prompt Chains

For this experiment, we vary the number of chains used in AMA. The default AMA for SuperGLUE WSC used 3 prompt chains. The results for this experiment can be found in Table 3.

# Chains	MV Acc	MV Precision	MV Recall	AAB Acc	AAB Precision	AAB Recall
3 (original)	0.615	0.571	0.563	0.622	0.583	0.576
1	0.635	0.592	0.578	**0.635**	**0.592**	**0.578**
5	**0.644**	**0.603**	**0.586**	0.617	0.573	0.565

Table 3: Results from varying the number of prompt chains with default AMA hyperparameters. MV stands for Majority Vote and AAB stands for Average Across Boost.

4.4 Varying the Output of In-Context Examples

Inspired by Min et al.'s [7] work investigating the effect of different outputs for in-context learning examples and the effect on performance, we wanted to vary the ouputs of the in-context examples we provided. Min et al. [7] found that in-distribution outputs for in-context examples only slightly affected overall performance. These results are relevant to this paper because if the outputs of in-context examples for prompts do not need to be completely accurate in prompt chains, the opportunity

to neurally generate in-context examples to scale up the number of prompt chains and in-context examples becomes possible.

We experiment with three types of output perturbations:

1. Shuffled: shuffling the outputs of in-context examples.
2. In-Distribution: writing new outputs for in-context examples that are within the distribution of possible outputs but are not correct.
3. Random: random outputs for in context examples.

Examples of the different perturbations to in-context example outputs can be found in Figure 2

```
Rewrite the input as a question.

Input: it was made of glass
Question: Who drowned?

Input: he drowned
Question: Laugh at who?

Input: laugh at them
Question: What was made of glass?
```

(a) Shuffling In-Context Outputs

```
Rewrite the input as a question.

Input: it was made of glass
Question: Who was at the party?

Input: he drowned
Question: Where was the city?

Input: laugh at them
Question: Who was Jorge?
```

(b) In Distribution In-Context Outputs

```
Rewrite the input as a question.

Input: it was made of glass
Question: money is so funny?

Input: he drowned
Question: 5 + 12 + 31

Input: laugh at them
Question: hahahahahaha
```

(c) Random In-Context Ouputs

Figure 2: Different Perturbation Strategies for In-Context Examples

The results of varying the outputs of in-context examples can be found in Table 4.

Output Status	MV Acc	MV Precision	MV Recall	AAB Acc	AAB Precision	AAB Recall
gold (original)	0.615	0.571	**0.563**	0.622	**0.583**	**0.575**
in distribution	**0.635**	0.317	0.500	**0.628**	0.428	0.499
random	0.625	0.316	0.492	0.622	0.456	0.497
shuffle	**0.635**	**0.573**	0.522	0.619	0.564	0.523

Table 4: Results from varying the ouputs of the in-context examples in prompts within prompt chains with default AMA hyperparameters. MV stands for Majority Vote and AAB stands for Average Across Boost.

4.5 Tuned Hyper Parameters

For this experiment, we combined the best hyper parameters from experimenting with number of in-context example per prompt and the number of prompt chains. The results can be found in Table 5.

# Chains	K per Chain	MV Acc	MV Precision	MV Recall	AAB Acc	AAB Precision	AAB Recall
1	1,7,1	0.702	0.676	0.659	0.702	0.676	0.659

Table 5: Results for taking the results from varying the number of prompt chains and number of in-context examples per prompt.

4.6 Neurally Generated Prompt Chains

For this experiment, we wanted to attempt to generate in-context examples for different prompts in the prompt chain with a much more powerful model. We attempted to use ChatGPT [8], which is model developed by OpenAI on top of GPT3.5 (175B foundation model with instruction finetuning). We attempted to generate in-context examples for the extraction, question generation, and answer generation chains and experimented with different combinations of neurally generated in-context examples. The number of chain and number of in-context examples were following the AMA default values.

We were excited by the prospect of being able to recursively generate prompts from LLMs in order to improve performance. This is motivated by the varying performances of the manually-written prompt chains, as well as the fact that scaling up the number of prompts chains would be significantly easier through some form of automation. We experimented with neurally generating all three aspects of the prompt chain. The method behind generating these prompts was simply inputting the gold prompt chains in ChatGPT [8] and asking it to `generate more examples`.

The results of the neurally-generated, in-context examples can be found in Table 6.

Prompt Chain	MV Acc	MV Precision	MV Recall	AAB Acc	AAB Precision	AAB Recall
original	0.615	0.571	0.563	**0.622**	**0.583**	**0.575**
entirely neural	0.567	0.423	0.464	0.619	0.542	0.523
neural extraction & answer	0.606	0.557	0.550	0.599	0.551	0.545
neural extraction	**0.644**	**0.612**	**0.608**	0.619	0.542	0.523
neural answer	0.625	0.575	0.560	0.612	0.564	0.553

Table 6: Results from using ChatGPT to generate new in-context examples for prompts in the prompt chain with default AMA hyperparameters. MV stands for Majority Vote and AAB stands for Average Across Boost.

5 Discussion

It is worth noting that for all of the experiments we ran, we used majority votes for the final prediction of the model instead of using weak supervision (as Arora et al. [1] propose in their paper). Our rational for this decision was to better understand how updating prompt chains would affect downstream results without weak supervision. In AMA they only report a a 4.1% increase from using weak supervision on the SuperGLUE WSC task. However, using weak supervision may have a more significant impact on performance when there are more prompt chains (which we experimented with).

5.1 Substitution

The substitution results were relatively straight-forward. Improving the quality of the extraction and generation steps improved downstream performance (by up to 9.4% when substituting question and extraction) when subsituting values from GPT-J-6B prompt chains into GPT-Neo-1.3B prompt chains. And subsitution in the other direction (worse extractions and questions) decreased downstream performance (by up to 17.2% when substituting question and extraction). This clearly demonstrates the value of responses during different steps of the prompt chain. In addition, we see that model capability still plays a significant role in the overall performance given that even with substitutions from GPT-J-6B, GPT-Neo-1.3B could not match the overall results of GPT-J-6B.

Another interesting observation is that the extraction step seems to be more difficult given that the majority of the gain from substitution of GPT-J-6B into GPT-Neo-1.3B prompt chain comes from the extraction substitution. The question substitution does not improve on the result from extraction substitution.

5.2 Number of Shots

The results from varying the number of shots within prompt chains are relatively counter intuitive. For the extraction step, increasing and decreasing the number of in-context examples improved performance. However, for the question generation step both increasing and decreasing the number of in-context examples decreased performance. For the answer generation step, decreasing number of in-context examples improved performance and increasing the number of in-context examples decreased performance.

The lack of clear trends helps motivate the further investigation of strategies to help with tuning the AMA strategy given that some of the results are relatively noisy.

5.3 Number of Prompt Chains

There was an increase in performance both when increasing and decreasing the number of prompt chains. The result for decreasing the number of prmopt chains just demonstrates that the first prompt chain was the most effectively written. However, the goal of the AMA framework is to synthesize a final answer from multiple noisy model predictions. Therefore, tuning hyperparameters and choosing only one prompt chain goes against the ethos of the AMA strategy. In addition, the use of weak supervision may improve the performance more drastically as we increase the number of prompt chains.

5.4 Tuned Hyper Parameter

When we combine the results from exploring the number of in-context examples per prompt and number of chains, we are able to achieve our best result overall of an accuracy of 70.2%. This is exciting because it demonstrates that combining hyper parameters from two different aspects of our ablation was additive.

5.5 Randomization

When investigating randomization, we experimented with shuffling the prompt outputs, randomizing outputs to other in-distribution examples, and finally randomizing outputs to the fullest extent. When we shuffled the outputs within each prompt chain, we see that the accuracy metrics are not greatly affected. The gold output outperforms the shuffling in MV recall, AAB accuracy, AAB precision and AAB recall, while the opposite holds for MV accuracy and MV precision (although the improvements are miniscule). This result was extremely interesting as it suggests that prompt chains could be "position-invariant" when it comes to outputs. Furthermore, it also means that in-context prompts learn with a fundamentally different approach than supervised learning - which is reliant on (x,y) pairs. Even though the input-output pairs have been switched, the LLM is able to locate the right output at a different, unintuitive location in the prompt.

For in-distribution randomization, we found that our results were not consistent with the work of Min et al. [7]. While there was a small gain in MV accuracy and AAB accuracy, we saw falls in MV precision, MV recall, AAB precision, and AAB recall. This makes sense as upon further examination, we found that the model would almost always output False (the WSC dataset is imbalanced with nearly two-thirds false outputs in the test set); thus although we see a slightly higher accuracy, the model performs much worse than it seems.

For complete randomness, we found the exact same pattern as the in-distribution case. In fact, the results show that there was no clear improvement in performance by using an in-distribution random output versus complete nonsense. This seems to suggest that there are only certain tasks in which the findings of Min et al. [7] are consistent with.

5.6 Neural Generation of Prompts

When investigating the neural generation of prompts, we initially used ChatGPT [8] prompts for all three modules of the prompt chain: extraction, question generation, and answer generation. We noticed a sizable drop in all accuracy metrics, with precision and recall taking the biggest hits. After qualitatively analyzing the new prompts as well as manually examining outputs of the model during testing time, we noticed that the question module of the prompt chain was performing particularly poorly. Thus, we reverted back to the original question generation prompts and re-ran the experiment. This lead to an increase in all accuracy metrics from the previous values - although the metrics were still lower than the gold prompts.

At this stage, we decided to try and run with only extraction or only answer generation prompts being neurally generated. We see that only using neurally generated answers makes the model perform better in terms of MV accuracy and MV precision (albeit marginally), and worse in terms of MV recall, AAB accuracy, AAB precision, and AAB recall when compared to gold prompts. Overall, it still doesn't seem like an upgrade. However, when we used ChatGPT [8] extraction prompts, we saw a marked increase in MV accuracy, MV precision, and MV recall.

This was quite exciting as from our personal experiences writing the prompt chains, the extraction step is the hardest to construct. In addition, from our substitution experiment 5.1 it appeared that the extraction step was also the most difficult for GPT-Neo-1.3B. Furthermore, we saw through the substitutions experiment that the main performance increase from GPT-J-6B into GPT-Neo-1.3B prompt chain comes from the extraction substitution.

6 Error Analysis

In looking at instances where our models performed poorly, a universal pattern we observed was that model outputs that are technically correct, but do not match the correct output string exactly, are deemed wrong. For instance, in the example:

```
Passage:  The path to the lake was blocked, so we couldn't reach it.
Question: What was blocked?
Ground Truth Answer: The path
```

we saw multiple chains produce the output `"The path to the lake"` as opposed to the correct output string `"The path"`. Clearly the model's output is also correct, however because it does not match the output string exactly, it is wrong. This problem stems from the decision Arora et al. [1] made to remove the two output options for the WSC task in favor of an unrestricted output space. One avenue we therefore identified for future work was to develop a method that mapped outputs from the open ended spave to the two WSC output options, perhaps by using cosine similarity of the output embeddings.

Another instance that negatively affects performance, which is pervasive amongst our various experiments, is that errors produced early on in each prompt chain are propagating through the chain and causing greater errors in later steps of the chain. This can be illustrated well in the following example:

```
Passage:  Joe Joe's uncle can still beat him at tennis, even though he is 30 years older.
Question: Who is older?
Ground Truth Answer: Joe Joe's uncle
```

And corresponding Model outputs for each step in the chain

```
Extractions:
["he still can beat", "he can still beat him at tennis", "he is 30 years older"]
Question:
["Whose still can beat?", "Can he still beat him at tennis?", "Whose 30 years older?"]
Answers:
["Joe", "Ye", "Joe Joe"]
Predictions:
["False", "False", "False"]
```

Here, the first two chains fail to extract the correct relevant part of the passage, which results in inaccurately formed questions and later on, incorrect answers. Thus, a mistake at any point in the chaining process can negatively affect the remainder of the process. This finding provides an explanation as to why our efforts in scaling up the number of prompt chains was not as successful as we had originally hoped. Certain chains consistenly appear to be far more susceptible to errors early on than others. This results in certain prompt chains sigificantly outperforming others. This can be seen in the Table 7 that shows the average accuracy of the performance of each prompt using the baseline number of shots per step (3, 7, 3). The best performing chain (Chain 1) achieves nearly 7% points higher than the worst performing chain (Chain 8).

These results also highlight that constructing well performing prompt chains is an extremely difficult task. Even within the three sets of chains written by Simran et al, us, and ChatGPT [8] there is extremely high variance in overall accuracy. Therefore, even if you scale up the number of prompts using neural models like ChatGPT [8], there is no guarantee on whether the chains are high performing or in fact detrimental to overall accuracy. This fact motivates the use of another method other than

	Chain 1 (A)	Chain 2 (A)	Chain 3 (A)	Chain 4 (B)	Chain 5 (B)	Chain 6 (C)	Chain 7 (C)	Chain 8 (C)
Accuracy	0.635	0.625	0.606	0.615	0.606	0.625	0.596	0.577

Table 7: Average Accuracy of each prompt chain. A are the prompts presented by Simran et al, B are the chains written by us and C are the neraully generated chains.

Majority Vote to aggregate the answers of the different chains. Simran et al. [1] propose the use of Weak Supervision as one method to capture correlation amongst chains and give less importance to chains that are highly correlated. Another proposal that could warrant future exploration could be retrieving the accuracies of an extremely large number of chains at train time and then only selecting the top k performing chains for running inference.

7 Conclusion

In this paper, we have shown that tuning the hyper parameters when using the AMA prompting framework can be an un-intuitive procedure. However, with rigourous experimentation it is possible to see significant increases in overall performance. For instance, we saw that actually reducing in context examples from 3 to 1 for passage extraction yielded a 9.4% increase in overall accuracy and over 7.2% improvement in both precision and recall.

We also showed that neural prompt chains generated by more powerful models can be used as valid alternatives to those generated by humans. In fact, using neurally generated examples for the extraction step saw an a 4.7% rise in accuracy vs the original extraction examples proposed by Simran et al [1]. This finding motivates the development of a more automated strategy for experimenting with the hyper parameters of AMA to find the most optimal settings. This tuning could avoid the need for significant human intervention by utilizing larger foundation models to help generate in-context examples.

Finally, we highlighted several avenues for potential future work that build upon the work presented in this paper. For instance, we believe an important future study would be exploring ways in which we can classify correct model outputs as such even when they do not exactly match the WSC output or working to develop an automated hyper parameter exploration strategy.

8 Contributions

Overall, we agree that every teammate contributed equally to the project. Here is a breakdown by person:

8.1 Andrew

- Set up AMA and infrastructure to run GPT-Neo and GPT-J
- Ran the substitution experiment and k-per-chain experiment

8.2 Aayush

- Ran randomization experiment
- Generated neural prompts using ChatGPT and ran related experiments

8.3 Oscar

- Ran number of chains experiment
- Performed detailed error analysis of model outputs

9 Code

`https://github.com/oscarfco/prompting_LLM`

References

[1] Simran Arora, Avanika Narayan, Mayee F. Chen, Laurel Orr, Neel Guha, Kush Bhatia, Ines Chami, Frederic Sala, and Christopher Ré. Ask me anything: A simple strategy for prompting language models, 2022.

[2] Sid Black, Gao Leo, Phil Wang, Connor Leahy, and Stella Biderman. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow, March 2021. If you use this software, please cite it using these metadata.

[3] Antonia Creswell, Murray Shanahan, and Irina Higgins. Selection-inference: Exploiting large language models for interpretable logical reasoning, 2022.

[4] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The Pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

[5] Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. Decomposed prompting: A modular approach for solving complex tasks, 2022.

[6] Hector J. Levesque, Ernest Davis, and Leora Morgenstern. The winograd schema challenge. In *13th International Conference on the Principles of Knowledge Representation and Reasoning, KR 2012*, Proceedings of the International Conference on Knowledge Representation and Reasoning, pages 552–561. Institute of Electrical and Electronics Engineers Inc., 2012. 13th International Conference on the Principles of Knowledge Representation and Reasoning, KR 2012 ; Conference date: 10-06-2012 Through 14-06-2012.

[7] Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. Rethinking the role of demonstrations: What makes in-context learning work?, 2022.

[8] John Schulman OpenAI, Barret Zoph, Christina Kim, and Jacob Hilton. Chatgpt: Optimizing language models for dialogue, Nov 2022.

[9] Ben Wang and Aran Komatsuzaki. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. `https://github.com/kingoflolz/mesh-transformer-jax`, May 2021.

[10] Tongshuang Wu, Ellen Jiang, Aaron Donsbach, Jeff Gray, Alejandra Molina, Michael Terry, and Carrie J Cai. Promptchainer: Chaining large language model prompts through visual programming, 2022.