CSE 447 Winter 2024 Final Project

Joey Krueger	**John Nguyen**	**William Hu**	**Angela Lee**
447	447	447	447
jkru3@uw.edu	john123@uw.edu	whu3@uw.edu	leeang22@uw.edu

Project Information:

Project Type	Default 447 Project
Task Name	ProtoQA
Benchmark Link	URL to ProtoQA

Joey Krueger: Architect of the multi-model pipeline. Wrote in sections 1, 2, 6, 7, and 10. Helped poster create the poster, and led presentation discussion.
John Nguyen: Co-implemented the multi-model pipeline code, wrote sections 3, 4, 5, 8, helped create the poster.
Angela Lee: Played with hyper-parameters, and contributed to the 2.
William Hu: Tested prompts and prompt engineering, and contributed to 9.

1 Introduction

Large language models are great at generating coherent answers, but generating great QA responses in Family Feud Style questions is a task that dives deep into the human psyche. In our approach to ProtoQA, we gave multiple language models different tasks along a refinement pipeline with the goal of creating an ideal set of responses that would enable us to outrank A12's ProtoQA GPT2 baseline in a novel way.

When we considered this project, there were a few things that drew our interest:

- This project would require a high degree of Commonsense reasoning. As we would come to find out, the authors of the paper "ProtoQA: A Question Answering Dataset for Prototypical Common-Sense Reasoning" have even formulated a series of cases for Commonsense reasoning in an NLP context
- The people answering family-feud questions are not often known for their PG-rated reputation. This in itself makes the data of the project entertaining to explore, but also carries with it a challenging in overcoming the censorship and biases held by pre-trained models
- The benchmarks for this project were straightforward enough to quickly develop a solution that could be easily tested, but complex where we could find many different meta-parameters and modularily expand upon the development of our solution.

Since our access to GPU was limited, we realized we had two options for methods in implementing our solution:

1. We fine-tune a smaller model (like GPT2 or BERT) that has already been used in previous attempts in outranking the ProtoQA benchmark.
2. We use a larger model, with the tradeoff that we wouldn't be able to fine-tune the model since the GPU cost would be too great.

Many previous attempts with GPT2, BERT, BART, and T5 had already been attempted with mixed results while newer and bigger models like Claude had shown promise in outpacing past attempts. We decided to stick with the method of using a larger model, which would give us the benefit of focusing on utilizing multiple models in a pipeline to further process our responses. In doing this, we found that we were actually able to do better than the benchmark for a fine-tuned GPT2 implementation with just the pre-trained and prompt tuning.

Our contribution to the project was in the development of a multimodel pipeline that generated responses in three stages:

- We made a model that generated output. Rather than fine-tune it for this specific purpose, we focused on feeding it a prompt that would consistently return our data in an enumerated list of simple responses. The prompt itself contained a meta-parameter instructing the model how many generations to create.
- We made a model that judged the quality of output. When generating output with a lower *generations* meta-parameter, we found that sometimes it would format responses in undesirable ways, and we used a model to reformat responses that were overly verbose, redundant, or otherwise non-sensible.
- We made a model to rank the outputs in context of the question. Since the order of the responses was important, we believed there was no reason to believe that this would be in the best order on generation, and used another model with zero-shot classification to reorder it.

We also provide empirical findings for how this setup performed at every stage of the pipeline.

2 Background

Members of our group had background in generating ranked results with LLMs before. In a previous project, prompt engineering was used in conjunction with a set of metrics to numerically rank large paragraphs of text on a set of metrics. This inspired the use of utilizing a zero-shot classifier used in the third model.

Some of us were also intrigued by the training set which felt like something pertaining as much to the domain of psychology as data science. Anyone who has played games like Apples to Apples, Mad Libs, or Cards Against Humanity knows, the "best" answers to Family Feud are not always the most correct (and rather, the most entertaining). Evaluating answers is a subjective judgment influenced by various cultural factors. While commonsense reasoning in machines has no official categorization to evaluate upon, a model with high accuracy can achieve a tremendous breakthrough in the quality of responses. The authors of the benchmark's paper alluded to awareness of that fact in the dataset they generated, as respondents to the questions were split into "clusters", indicating that different groups of humans were likely to share results that were similar across many different questions.

Although 3 of the 4 members of our group had taken 446 previously, our experience training and working with language models was largely limited. However, we found some HuggingFace frameworks that could generate useful results. The base zephyr-7b-beta model is a fine-tuned model that can work for a variety of question and answer applications. We largely focused on using and extending on this model to generate possible answers for Family Feud questions and further improving on their quality.

3 Generation Processing

3.1 Prompt Tuning

In order to establish quality generations from our model, we created a series of prompts for our zephyr-7b-beta chatbot to comply with. Using the 'system' role parameter of apply_chat_template, we steered the models' generation toward a simple list of answers, reducing unnecessary explanations.

Our first prompt, ***'you are on a family feud gameshow and you will ONLY generate [n] answers to the following question I ask'***, provided a good baseline for our model to generate only a ranked list of n answers, however, we noticed too high of a complexity in the generations.

For instance, when asked, 'Name something that is hard to guess about a person you are just meeting?' Our model responded:
'One thing that is hard to guess about a person you are just meeting is the person's hidden secret'
'A person's age is normally something hard to guess when you are just meeting them'
This was on the right track to meeting our target: 'Secret', 'Age'

We tweaked our final prompt with the intention of generating simple yet diverse answers and landed on ***'you are on a family feud gameshow and you will ONLY generate [n] simple, different answers (no adjective, just the noun) with no explanations to the following question I ask'***. Here our model frequently generated varied answers corresponding with the appropriate labels for the question. For instance, when asked, 'Name something that is hard to guess about a person you are just meeting?' Our model answered:
'Secret', 'Belief (Religion)', ..., 'Age'
Target: 'Secret', 'Belief', ..., 'Age'

This gave our prediction a much better initial response to improve upon.

3.2 Generation Filtering

This prompt tuning helped steer the generations toward desirable answers, however as seen in the last example, we still had to filter out any extra explanations since the target format for answers was not always guaranteed. This involved removing anything between parenthesis '(', ')' and removing anything after ':'

'Belief (Religion)' $\rightarrow$ 'Belief'

'Belief: Catholicism, Christianity, Buddhism, Hinduism' $\rightarrow$ 'Belief'

3.3 Generation Length

To encourage our model to generate more diverse generations, we also attempted to ask our model to generate n results for each question where n = 10, 20, 30, 40, and 50.

3.4 Temperature

For the generation of our model, we set the temperature to 0.1 to encourage the model to only generate the most obvious answers. This ensured that our model would not generate any answers with low probability since they are unlikely to be an answer to match the true labels.

4 Evaluation

For evaluation, generated answers from our model must be matched to answers proposed by the authors. Exact Match similarity, which gives points only if our answer matches the true label exactly, is the simplest approach for evaluation. However, generations may be marked incorrect although they present a clear synonym of a reference answer. To propose a solution to matching the model's generated ranked list of answers and the true label answers presented by the authors, we utilized WordNet Similarity for evaluation. The evaluation criteria and implementation are copied from the ProtoQA baseline, and is described here: "We take a similar approach, tokenizing a proposed answer string and comparing it to the tokenization of the answers in each answer cluster. Since some words in WordNet are multi-word phrases (eg. "chewing gum") we furthermore perform this matching on all possible partitions of the tokenization. For each answer in an answer cluster we return the maximum (over all possible partitions) of the average number of matched tokens. The assignment of answers to clusters proceeds as in the exact match case" (Boratko et al., 2020).

5 Method

We explore three different models for this task: a QA-based model which provides a ranked list of answers given a question through the use of the model's chatbot feature, a Judgement Model that refines the generations from the QA-based model, and a Zero Shot Classification model that ranks the answers from the judgement model for evaluation. Using the base chatbot templates for models 1 and 2, we were able to reduce computation and resources needed to fine-tune a large model.

5.1 Brainstorm Model – QA-based model

We report a brainstorm model that utilizes the base zephyr-7b-beta model for basic question-answering tasks. As this dataset is in the form of questions and answers, we predicted that

the base model's pretrained parameters would perform well with providing a baseline to work on top of. To generate results for the benchmark with this approach, we gave the model each question and asked our model to generate a ranked list of answers. After going through the generation processing steps as outlined above, we would evaluate our model on the model's generations.

5.2 Judgement Model – The quality model

Although the generations produced by the Brainstorm Model exceeded initial expectations, there was still room for improvement. Specifically, on questions where our Brainstorm model's generations started to get repetitive, or the model produced answers that were too specific. We utilized the same zephyr-7b-beta model, but instead changed the prompt for the model's 'system' role to be:
Given a list of [n] answers, I want you to replace any answers that are a synonym of another answer with a new answer for the question, but keep the answers simple with no adjectives, only the noun; Return ONLY [n] answers for every question with no explanations.

Intended Purpose:
When asked, 'Name one thing you would bring with you if it was raining outside?'
Brainstorm model:
'umbrella', ..., 'hat', ..., 'cap', ..., 'a lightweight umbrella'
After Judgement Model:
'umbrella', ..., 'hat', ..., [replace since cap is similar to hat], ..., [replace since lightweight umbrella is same as umbrella]

Originally, we wanted to implement a 'system' role where we provide the entire evaluate criteria for the model to emulate and then proceed to replace answers, however this proved to be too complex for the model to comprehend and we noticed a dramatic loss in quality. For simple generations, our model is able to recognize patterns and will replace text that it sees repetitive.

5.3 Zero Shot Classification – The ranking model

For the final experiment, we attempted to rerank the generations given after going through the Judgement Model. This involved using the model DeBERTa-v3-base-mnli-fever-anli to assign labels (the ranked list of answers) to each question and have our model infer a score for each word. This allowed us to rerank the words based on the scores and evaluate our model on the newly ranked list of answers. Since zero shot classification leverages a pretrained model's classification ability where the amount of labeled data is small, we attempted to model this situation with our benchmark. In the previous step, we attempted to try to do this task simultaneously by including in the prompt to rank the outputs generated, however, this proved to be too complex for the model and we noticed a drop in performance for the Judgement Model.

Intended Purpose:
When asked, 'Name one thing you would bring with you if it was raining outside?'
After Judgement Model:
'umbrella', ..., 'hat', ..., 'boots', ..., 'raincoat'
After Zero Shot Classification:
'raincoat', 'umbrella', 'hat', 'boots', ...

The output above is subjective, but this is an appropriate ranking we would hope our model to output.

6 Experiments

6.1 Datasets

We evaluated on https://github.com/iesl/protoqa-data/blob/master/data/dev/dev.crowdsourced.jsonl, a dev set that contains 50 questions, to measure the quality of our metrics. This is directly linked from a github repo linked to the ProtoQA benchmark on A12.

6.2 Baselines

The primary baseline metrics we compared our model's performance too were included in Table 3 of the ProtoQA paper.

6.3 Code

We created a few code-bases for our model. A simplified version with detailed instructions that includes the results we submitted to ProtoQA can be found here. This does not contain results generated by the judgement model and the ranking model, since these results were sub-optimal:
ProtoQA Submission Codebase

We also called on code from the following repo (provided by the benchmark) for it's evaluation metrics
ProtoQA Evaluator

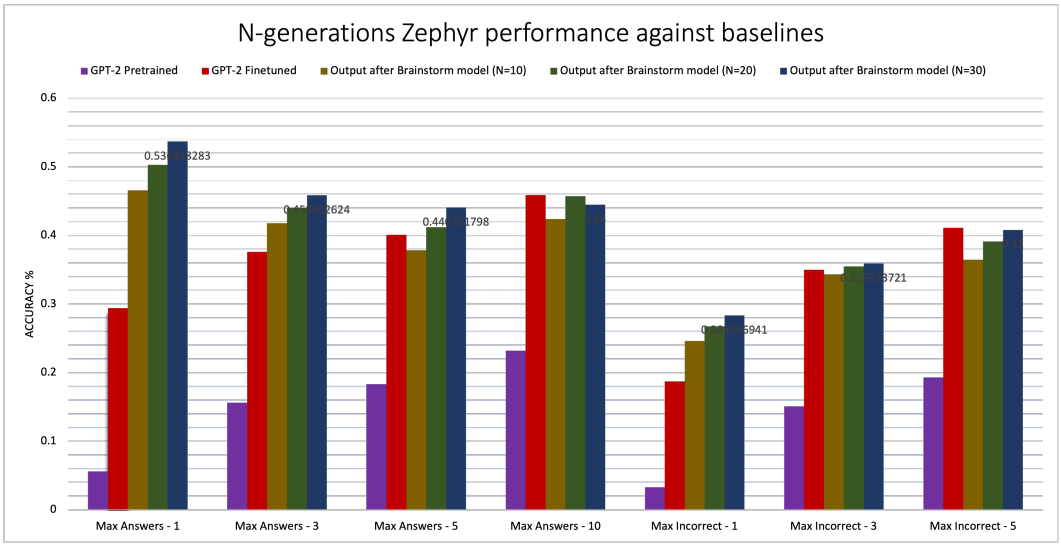
A more detailed example of our codebase (which we created copies of for each) n value can be found here, which includes the all results we included in our findings for this report.
N=10 Generations
N=20 Generations
N=30 Generations
N=40 Generations
N=50 Generations

7 Results

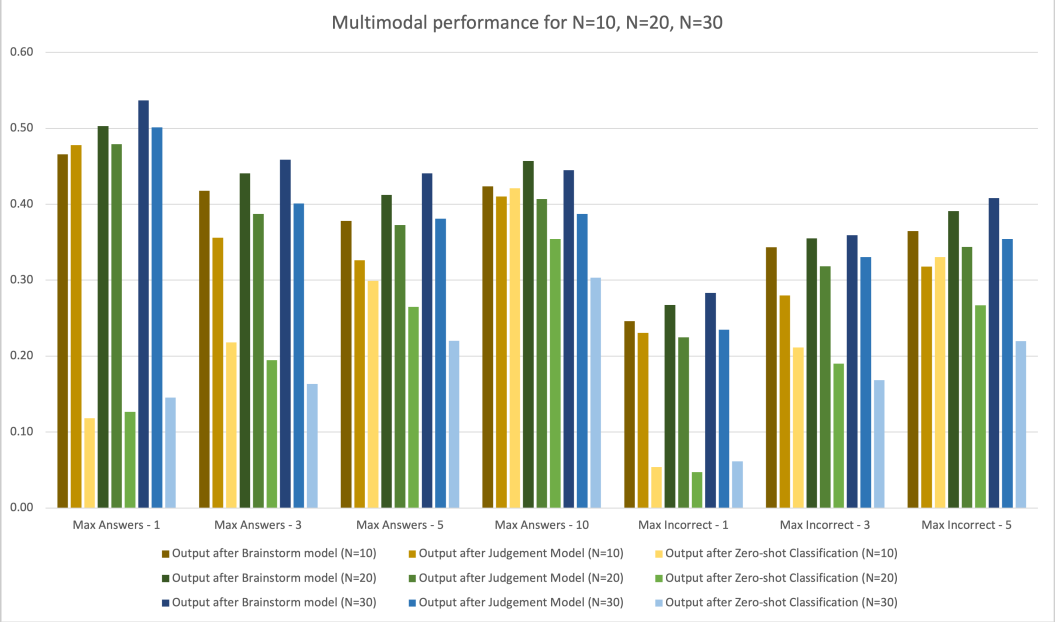
Pretrained Zephyr-7b-beta outperformed the finetuned GPT-2 baselines on almost every metric except for "Max Answers - 10" and "Max Incorrect -5".

One surprising finding found was that the accuracy of the top 10 generations increased when telling our model to generate 20 tokens and it increased even further when it generated 30. At 40, the performance started to decline and at 50, we couldn't even get it to generate that number consistently.

One possible explanation for this initial increase is due to the how the decoder model prioritizes it's generations when it is told to generate more results. It seems possible that an n of 30 (which was our optimal n value) increases the diversity of responses at the beginning of the generation when compared to an n of just 10.



Unfortunately, we were not able to yield any further optimization of our metrics with our additional models as seen in the table below. In only one case (where $n = 10$, Max Answers - 1) were we able to get the judgement model to improve the generations of the brainstorm model, and in all other cases it did significantly worse, regardless of the number of responses generated.



While these initial results are not promising, we think there is significant room for improvement with fine-tuning as outlined in section 8.

8 Constraints

Due to computational resources, our researchers were not able to successfully finetune our large model on the training corpus given by the ProtoQA benchmark. Although attempts to finetune smaller models were made, no results were proven successful as finetuning smaller models still didn't allow for the same quality of basic question answering as the base zephyr-7b-beta model. We wished to also tune the Zero Shot Classification model on the training data to get better at assigning scores for certain questions, however, that also proved to be cost restrictive. We hypothesize that finetuning these models will provide an

increase in accuracy during prediction, and will work better for the intended purposes that we had hoped for.

9 Discussion

Because ProtoQA is focused on prototypical Commonsense reasoning, the implication of creating a solution to solve this benchmark with high accuracy and precision is that language models will have the ability to answer questions that previously thought of as needing the humanistic thinking to generate creative and common answers. Our results showed that using a pre-trained judgement model has comparable to sub-optimal results to using only the brainstorm model. Fine-tuning is clearly necessary if accuracy is to improve with this method, if at all.

Most interestingly, our pre-trained zero-shot classification model significantly reduced our metric's accuracy by a high amount, with two notable trends:

1. As n increased, the accuracy of the zero-shot ranking model decreased. This is likely because it clusters similar results in the top-k position (which is a detriment to diversity of responses)
2. As max answers considered increases, so too does the output of the zero-shot ranking model. This makes sense because probability would tell us that the larger the sample size we look at, the more likely we are to encounter accurate samples. AKA: our "improved" findings likely have little to do with the increasing quality of the zero-shot classifier's capabilities and more to do with regression towards the mean. Not the best measure of quality!

As previously stated however, we believe there is potential for fine-tuning to actually make both of these models significantly better and perhaps even outperform the standard generation model.

10 Conclusion

In this paper, we covered our experiment in optimizing a ProtoQA solution for Commonsense reasoning. Given our background, we figured the project would have a dataset with answers that would be interesting to try to replicate with numerous implementation decisions to expand upon. We outline the decisions that led us to stick with a pre-trained version of Zephyr-7b-beta, which yielded surprising results in combination with 1. prompt tuning, 2. filtering, and 3. (and perhaps most surprisingly), increasing the length of generated results (which we only took the first 10 of). While our multi-model method yielded sub-optimal results to our initial generations, we believe that this experiment provided insights into some of the weaknesses of using pre-trained models for the purpose of judgment and ranking of quality Commonsense reasoned responses.

However, our experiments are no-where near conclusive, and while we were not able to demonstrate improved results through our Judgement and Ranking models, we believe that with less computational constraints, they present a baseline that can be improved upon through fine-tuning in future work.

References

Boratko, M., Li, X. L., Das, R., O'Gorman, T., Le, D., & McCallum, A. (2020). ProtoQA: A Question Answering Dataset for Prototypical Common-Sense Reasoning. arXiv:2005.00771v3 [cs.CL]. Retrieved from https://doi.org/10.48550/arXiv.2005.00771

Acknowledgments

This template is modified from the COLM 2024 paper template. Instructions are written by Liwei Jiang, Alisa Liu, and Yegor Kuznetsov.

Special thanks to the entire TA staff for CSE 447, especially Yegor Kuznetsov, who helped us greatly in committing to a direction for our project.