Evaluation of different response lengths for an LLM

Max Kaufmann
University of Maryland
kaufmann@terpmail.umd.edu
December 9th, 2024

Abstract

This project aims to evaluate and grade responses generated by large language models (LLMs). Unlike prior research, this determines how much of a response is needed to accurately evaluate. The primary objectives are (1) create a model capable of accurate pairwise evaluations of different responses, (2) determine the minimal response length required for accurate assessments. This work can be used to select the best response an LLM gives, which has different applications.

1 Introduction

Full conversations with AI are now possible due to recent advancements powered by large language models (LLMs). The popular use case for these models are for text generation, which output human-like responses. They can be applied to a diverse set of tasks. They have reached a level to where many people use them for question answering, solving complex problems, summarization, translation, and much more. However, the responses an LLM gives can vary and can be non-deterministic. Some of the outputs will not be optimal and could be irrelevant, incoherent, factually incorrect, or even harmful. Also, the best model can change depending on the query. Being able to evaluate different responses can improve accuracy and help align models [1]. Typical evaluation methods analyze results after they have been completed. Two common uses of evaluation are for training the model, like reinforcement learning from human feedback (RLHF), and for ranking items in search, like retrieval-augmented generation systems.

However, there is a lack of research into evaluating a partial response. Prompt-guards can be used to stop the response if it is detected to be harmful, but this doesn't adjust responses [2]. Evaluations of partial responses could be used to adjust in real-time interactions, where responses are streamed to a user or in Voice AI. Another advantage of this would be to reduce computational resources. If the response of an LLM can be judged before the full output, compute resources are not spent on generating suboptimal text. The price and resources needed to power large models are a bottleneck in their performance and scaling them.

This project examines the performance of three different models that can evaluate responses, and how their performance changes when the length of the response is shortened.

2 Related Work

2.1 Methods to evaluate text

There was a need to evaluate and rank text-responses long-before the recent evolution of language models. Early approaches relied heavily on surface-level metrics derived from statistical comparisons to reference texts. For example, BLEU (Bilingual Evaluation Understudy) and ROUGE (Recall-Oriented Understudy for Gisting Evaluation) emerged as in machine translation and summarization, respectively. These metrics operate by measuring the overlap of n-grams between a candidate response and a human-written reference. While they offered a quantifiable and reproducible way to gauge

model quality, their reliance on lexical overlap limited their capacity to capture the semantic meaning or conversational appropriateness. For example, they struggle to consistently match paraphrases.

In the context of search engines, ranking search results is a critical step so that the top of the results contain the most relevant and useful information. Traditional methods relied on keyword matching and statistical metrics, such as Term Frequency-Inverse Document Frequency (TF-IDF) and BM25, which prioritized documents based on the frequency and distribution of query terms. However, these approaches often struggle with capturing the semantic relationships between the query and potential results. In order to improve, ranking methods can be done to reorder which results are best. Modern ranking methods have significantly evolved with the advent of machine learning and deep learning techniques. Models like neural re-rankers utilize embeddings to measure semantic similarity between a query and documents, enabling more nuanced interpretations of user intent. Additionally, factors such as user behavior (click-through rates) and previous searches can provide contextual information to refine rankings further. By leveraging these advancements, search engines can rank results more effectively, providing users with a seamless and satisfying search experience that transcends simple keyword matching.

Models for ranking can fall intro three categories: pointwise, pairwise, and listwise. Pointwise is when a score for each item is computed, which could then be used to numerically rank. Regression models can be used for pointwise. Pairwise are when two items are compared simultaneously, and the output determines which is better. This is similar to Bradley-Terry style models, which are probabilistic models for pairwise comparisons. Listwise can compare multiple items simultaneously, and output the rank. These are apart of learning-to-rank methods, which are most commonly gradient boosted decision trees. While those may be faster, neural network approaches can offer greater performance [3].

For text tasks, sequential processing improves performance because the order of words determines meaning. Recurrent Neural Networks allow for such data. They do this by maintaining a hidden state that contains information about steps. However, vanishing gradients hurt performance of these models. An improvement was Long Short-Term Memory Networks, which could remember long-term dependencies in the sequential data. More recently, a new architecture called transformers came out. This architecture relies entirely on self-attention mechanisms to process input sequences, eliminating the need for recurrent layers. The model's key innovation is the the multi-head self-attention mechanism, which allows it to capture dependencies across sequences more efficiently and in parallel. This attention mechanisms can create relationships between each of the inputs, and how important that relationship is [4]. This approach is the foundation of the current state-of-the-art models in natural language processing and other fields as well.

The current transformer architecture that powers ChatGPT is known as Generative Pre-Training. This approach first trains a transformer model on a large corpus of text data, and then uses unsupervised learning to predict the next words in a sequence. After that training stage is done, the model undergoes supervised fine-tuning for downstream tasks.

Another transformer are Bidirectional Transformers (BERT). Unlike other transformers which process the data uni-directionally (left-to-right or right-to-left), BERT can use the context on both sides to make predictions. BERT is good for masked language modeling, which is like fill in the blank. The masked words in the input are hidden, and the goal of the model is to predict what they are [5]. Another use case of BERT is to do ranking. Called BERTScore, this produces a similarity score for each the embedding of each token in the candidate sentence against the embedding of each token in the reference sentences. This goes beyond comparing comparing simple embeddings because it utilizes the surrounding words to create contextual embeddings [6].

A great improvement to these models is to use reinforcement learning from human feedback. For example, a model called InstructGPT had less than 1% of the GPT-3 parameters, but scored better on human evaluations. InstructGPT was fine-tuned using reinforcement learning from human feedback. The first stage was to create a reward-model. the reward-model is trained on pairwise classification that comes from human annotations of which response is better, and outputs a scalar reward. Thus the reward model is an evaluator, where responses that are more aligned are scored better. Then, this reward model is used to further fine-tune the GPT's response. This reward model plays a crucial part because its ability to evaluate responses determines the style of the response [7].

Reward-models can be transformers themselves. The Skywork-Reward-Gemma-2 comes from google's gemma model. This model was fine-tuned using a careful dataset. While the model can still do text-generation because the architecture comes from gemma, in order to evaluate the score the logit on the BOS Token (start of sequence) is used as the output. Then for two different responses, the one with the higher score is predicted to be better [8]. Another reward-model that is a transformer is the PairRM. This comes from Microsoft's DeBERTaV3. DeBERTa is similar to BERT, but it uses disentangled attention. This separates the content embeddings and position embeddings into two distinct vectors, and attention mechanism processes the two separately. The advantage is that DeBERTa has relative position biases, which helps it better understand the relationships between words when compared to absolute positions. PairRM does pairwise classification. It uses special tokens to separate the prompt, first response, and second response. The model was trained to output a positive number if the first response is better than the second response, and a negative number if the second response is better, thus acting like a binary classifier. PairRM was fine-tuned on six different human-preference datasets [1].

Rather than having a specific reward-model, LLMs have been used to evaluate text. One study that popularized this was LLM-as-judge, which utilized prompt engineering to do pointwise score of a response. This was shown to give better results than BLUE, ROGUE, and other metrics for their respective task [9]. The prompt-engineering can influence how well these perform. Different tasks did better with different prompts. Also, Chain-of-thought, where the LLM is first asked to generate reasons for an evaluation before giving a score, give better response [10]. Pairwise comparisons can also be used with LLMs, and can perform better than pointwise [11]. Another approach to uses LLMs for evaluation is fine-tuned models specifically for judging. One such is Skywork-Critic-Llama-3.1-8B SFR-Judge, which comes from Meta's LLama 3. During the fine-tuning process, a specific prompt template was used for the pairwise comparison. This model would then output "[[A]]" if the first response was better, else it should do "[[B]]" [12]. Another fine-tuned LLM for judging was SFR-Judge. This was fine-tuned to judge three ways: pointwise, pairwise, and binary classifications. Unlike the Skywork-Reward-Gemma-2 or Skywork-Critic-LLAMA-3.1, SFR-Judge can generate an explanation why it judged the way they did. This helps mitigate the black-box nature of many other judge models because a human could look at the explanation SFR-Judge gave, instead of just a numerical score [13]. Another way to fine-tune is to do self-taught models. One such Llama model was fine-tuned by Meta, and only used synthetic training data to train itself [14]. These models that were fine-tuned for evaluating text performed better than with the original weights of the model.

2.2 Datasets to evaluate text

The performance of a LLM heavily depends on the dataset it was trained on. One example is Hammer 2.0 models, which are fine-tuned from Qwen2.5 models. The Hammer 2.0 models were trained on function calling datasets. The 7B version was able to outperform conversational-based models 10-100 times its size on Berkeley Function-Calling leaderboard [15]. RLHF datasets exist in many different forms. One such is a dataset specifically for summarizing text, and using human annotations to decide which which summary was better. This resulted in GPT models performing better when asked to summarize [16]. The largest RLHF dataset comes from Chatbot Arena, which crowd sources human preferences by having people vote which response they prefer on a website. As of now, it has over 1 million pairwise comparisons. This dataset was compared to a subset of expert raters, and the two were shown to be in agreement, which adds credibility to crowd sourcing human annotation [17]. Datasets also exist for multi-way comparisons, which can help with ranking. The Nectar dataset provides a 7-wise comparison framework—each prompt is paired with seven distinct responses—allowing. The advantage of this dataset is that it allows to examine the researchers fine-grained differences in quality and preference. A clear advantage of such multi-response settings is that they encourage models to identify nuanced quality gradients rather than just picking a "winner" between two candidates. However, managing and interpreting 7-way comparisons can be more computationally and analytically complex, and it may be more challenging to create a clear gold standard for evaluation, especially when human evaluators themselves vary in their preferences [18]. Another dataset for training reward models is to encode responses into different attributes. For example, UltraFeedback scores responses on instruction-following, truthfulness, honesty, and helpfulness. The advantage here is the dataset's fine-grained guidance, which can lead to highly specialized reward models. For example, if a doctor used an AI assistant truthfulness would be much more important than for an AI assistant a comedian uses [19].

Its commonly believed that the more data a machine learning model is trained on, the better it will perform. However, specialized datasets that curate the best examples were shown to lead to better performances. Skywork-Reward is a dataset that uses specific filtering strategies. This dataset was used with the Skywork-Reward-Gemma-27B model, which is currently the top model on the RewardBench dataset [8].

Another dataset specifically for LLM judges is LLM-BAR. This dataset is contains examples that is meant to mislead an LLM evaluator. For example, a more engaging tone. This dataset is important because it shows that while LLMs can be used to judge, they have biases based on the style the response was given in. For example, for two responses where one is correct and one is wrong, if the wrong was response was in the style people preferred, then LLM judges would have worse accuracy on that compared to two responses in the same style [20]. On par with this, LLM judges can suffer from self-enhancement bias, which is when a model prefers responses from itself over other models [9]

A problem with human preferences is that they focus on the style and format a person prefers. The RewardBench dataset was created to help mitigate this. It provides binary pairwise comparisons, where one response was verifiably better than another [21].

3 Approach

3.1 Chosen Dataset

For real-time conversations, there is an inherent limitation due to the need for low-latency and immediately give a response. At the very least, responses should be evaluated for their objectiveness. Rewardbench does that by having a verifiably correct answer. The dataset consistent of 5 sections: Chat, which tests on basic open-ended questions, Chat Hard, which tests on trick questions and subtitles, Safety, which tests on LLM's refusal to answer, Reasoning, which tests on code and reasoning abilities, and Prior Sets, which test on preexisting human preference datasets. For this research, all 5123 items in the dataset were used to calculate accuracy.

3.2 Chosen models

The first model that was tested was Skywork-Reward-Gemma-2-27B. This model is finetuned, and does pointwise comparison. This model has around 27 billion parameters. The second model was using Skywork-Critic-llama-3.1-8b. This uses a fine-tuned LLM-as-judge and a pairwise comparison approach The model has 8 billion parameters. The third model was using PairRM. This is finetuned using pairwise comparison, and outputs a numerical binary classification. The model size is the smallest, with 400 million parameters. The rationale for picking these models is that they are all of different sizes, and use a different method.

3.3 Testing methology

The metric measured was binary accuracy. The RewardBench has a total of 5123 samples. The lengths of the response taken were 100%, 75%, 50%, 25%, 10%, and 5%. The part of the response that was came from the start. To get the length, the characters were multiplied by the percent taken, and rounded down. The average response was 921 characters, which for Llama models will be about 230 tokens. However, by taking the proportion before tokenization, this standardized it across models. The compute ran on NVIDIA GPUs hosted on runpod.io, and the configuration on hugging face of each model was used.

4 Results

Figure 1: Response Lengths vs Accuracy for models

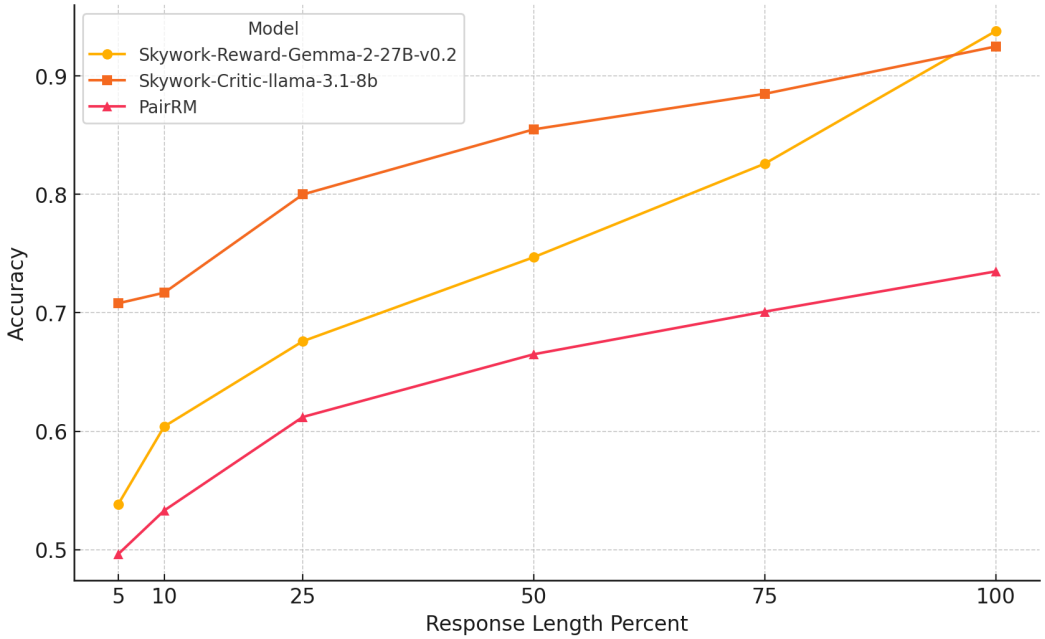

Table 1: Accuracy and P-value comparisons across models and response lengths.

Model	Response Length Percent	Accuracy	P-value
Skywork-Reward-Gemma-2-27B-v0.2	100	0.938	5e-324
	75	0.826	5e-324
	50	0.747	5e-285
	25	0.676	4e-143
	10	0.604	4e-50
	5	0.538	7e-08
Skywork-Critic-llama-3.1-8b	100	0.925	5e-324
	75	0.885	5e-324
	50	0.855	5e-324
	25	0.8	5e-324
	10	0.717	2e-219
	5	0.708	9e-200
PairRM	100	0.735	3e-257
	75	0.701	1e-186
	50	0.665	6e-125
	25	0.612	1e-75
	10	0.533	2e-06
	5	0.496	6e-01

5 Conclusion

This research showcases the capability of large language models to do pairwise comparisons between good and bad responses. The models can also be accurate, even when the input was heavily truncated. For all three models, there was statistical evidence that accurate evaluations can be made with as little as 10% of the original response length, which is about be 18–19 words. The data also showed a clear trend that as response length shortens, so does accuracy.

6 Further Work

The first and simple improvement of this is testing with different models to see if different methods of fine-tuning or model size made a difference. Also, it could be insightful to analyze different datasets like UltraFeedback or Nectar.

The goal of this research was initially to create a general framework that could be used to evaluate and adjust responses for a conversation over voice. An LLM capable of assessing its responses on-the-fly can lead to better interactions by interjecting feedback or having a chance to correct and clarify itself. An LLM continues generating tokens without considering new ideas while generating. When people speak, they assess themselves while talking. If an LLM could mirror a human conversation in this way, the responses could be adjusted and improved.

However, to minimize latency, the text-generation model and evaluation model should be deployed in the same region. Also, all of the models used here are fine-tuned. This makes them unpopular and there isn't a generic serverless inference for this. In order to use these models, they would have to be self-hosted, which is quite costly. Running the tests for these models for just a couple hours costed $80 dollars. For someone that wants to deploy a system like this, they might have to pay thousands of dollars a month to have these models ready to go 24/7.

The PairRM model tested is apart of a project called LLM-Blender, which uses an ensemble of models to generate responses, pick the best-n responses, and then fuse them together to create a response that uses the best parts of each response. The evidence that ensembles of models can improve responses and evaluators can use just a sentence to determine the better result indicates responses could be improved mid-generation in a real-time setting.

References

[1] Jiang, Dongfu. "LLM-Blender: Ensembling Large Language Models with Pairwise Ranking and Generative Fusion." Allen Institute for Artificial Intelligence, June 2023. https://arxiv.org/pdf/2306.02561.

[2] OpenAI. "GPT-4o Mini: Advancing Cost-Efficient Intelligence," July 2024. https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/.

[3] Burges, Christopher. "From RankNet to LambdaRank to LambdaMART: An Overview." Microsoft Research Technical Report, January 2010. https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/MSR-TR-2010-82.pdf.

[4] Vaswani, Ashish. "Attention Is All You Need." Google Brain, 2017. https://arxiv.org/abs/1706.03762.

[5] Devlin, Jacob. "BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding," October 2018. https://arxiv.org/abs/1810.04805.

[6] Zhang, Tianyi. "BERTSCORE: EVALUATING TEXT GENERATION WITH BERT," February 2020. https://arxiv.org/pdf/1904.09675.

[7] OpenAI. "Training Language Models to Follow Instructions with Human Feedback," March 2022. https://arxiv.org/abs/2203.02155.

[8] Liu, Chris. "Skywork-Reward: Bag of Tricks for Reward Modeling in LLMs." Skywork, October 2024. https://arxiv.org/abs/2410.18451.

[9] Zheng, Lianmin. "Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena," June 2023. https://arxiv.org/abs/2306.05685.

[10] Kaufmann, M. "AI Self Eval," May 2024. https://github.com/mkaufmann84/ai-self-eval.

[11] Qin, Zhen. "Large Language Models Are Effective Text Rankers with Pairwise Ranking Prompting," March 2024. https://arxiv.org/pdf/2306.17563.

[11] Wang, Tianlu. "Self-Taught Evaluators," August 2024. https://arxiv.org/pdf/2408.02666.

[12] Skywork. "Skywork/Skywork-Critic-Llama-3.1-8B," September 2024. https://huggingface.co/Skywork/Skywork-Critic-Llama-3.1-8B.

[13] Wang, Peifeng. "DIRECT JUDGEMENT PREFERENCE OPTIMIZATION," September 2024. https://arxiv.org/pdf/2409.14664.

[14] Wang, Tianlu. "Self-Taught Evaluators," August 2024. https://arxiv.org/pdf/2408.02666.

[15] MadeAgents. "Hammer2.0-7b," September 2024. https://huggingface.co/MadeAgents/Hammer2.0-7b/tree/main.

[16] OpenAI. "Learning to Summarize from Human Feedback," February. https://arxiv.org/pdf/2009.01325.

[17] Chiang, Wei-Lin. "Chatbot Arena: An Open Platform for Evaluating LLMs by Human Preference," March 2024. https://arxiv.org/abs/2403.04132.

[18] Zhu, Banghua. "Berkeley-Nest/Nectar," November 2023. https://huggingface.co/datasets/berkeley-nest/Nectar?row=0.

[19] Cui, Ganqu. "Openbmb/UltraFeedback," 2023. https://huggingface.co/datasets/openbmb/UltraFeedback?row=1.

[20] Zeng, Zhiyuan. "Evaluating Large Language Models at Evaluating Instruction Following," October 2023. https://arxiv.org/abs/2310.07641.

[21] Lambert, Nathan. "RewardBench: Evaluating Reward Models for Language Modeling," March 2024. https://arxiv.org/abs/2403.13787.