LLM's vs JEE Advanced: Indian Teenagers Worst Nightmare!

Ali Hasan Mohiuddin
G01389462
amohiud@gmu.edu

Anjan Kumar Sripati Panditaradhyula
G01407403
asripati@gmu.edu

1 Introduction

1.1 Task / Research Question Description

As we are very much familiar with the pattern of the JEE-Advanced exam, we did have have clarity in what the authors of "Have LLMs Advanced Enough? A Challenging Problem Solving Benchmark For Large Language Models" (1) were trying to achieve with variations in GPT-4.The problem they addressed was; "How well do large language models (LLMs), particularly GPT-4, perform on a new, more challenging benchmark dataset called JeeBench, which consists of 515 difficult pre-engineering mathematics, physics, and chemistry problems from the highly competitive IIT JEE-Advanced exam?"

In simpler terms, the authors want to know how good GPT-4 and other LLMs are at solving complex reasoning problems in subjects like math, physics, and chemistry, and whether they can handle the kind of challenging problems found in competitive exams like IIT JEE-Advanced. They also want to understand the strengths and weaknesses of these models when faced with such problems. We are trying to validate whether this is the case, that is, whether the model can perform efficiently and better than the prior work done in this regard.

1.2 Motivation and Limitations of existing work

The motivation behind the research is driven by the significant improvements in the performance of large language models (LLMs) on reasoning benchmarks over the past years. The authors recognize the need for harder benchmarks to evaluate the problem-solving abilities of LLMs accurately. To address this need, they use JEEBENCH, a more challenging benchmark dataset that requires complex logical and mathematical reasoning on top of deep in-domain knowledge.

Limitations : While existing benchmarks have shown improvements in LLMs' performance, they may not be sufficiently challenging to evaluate the true problem-solving abilities of these models. There may be shortcomings in existing LLMs' capabilities, such as errors in algebraic manipulation, difficulty in grounding abstract concepts into mathematical equations accurately, and failure to retrieve relevant domain-specific concepts.

The prior work had shortcomings where the LLMs could not perform as well on challenging benchmarks that require long-horizon reasoning and deep domain knowledge.

1.3 Proposed Approach

The authors plan to evaluate various open-source and proprietary LLMs on the JEEBENCH dataset. They aim to measure the performance of these models using metrics such as accuracy, reasoning capability, and ability to handle long-horizon reasoning.

Model Techniques:

The authors mention using techniques like self-consistency, self-refinement, and chain-of-thought prompting to enhance the performance of LLMs on the benchmark dataset. They also introduce a post-hoc confidence-thresholding method over self-consistency to enable effective response selection, especially for handling risk assessment introduced by negative marking in the exam.

They highlight avenues for further research, such as exploring the effectiveness of self-critique approaches in improving LLM performance and addressing the limits of LLMs' reasoning abilities. It also emphasizes the importance of building LLMs faithful to mathematical logic and developing better methods for estimating LLM performance compared to humans.

1.4 Likely challenges and mitigations

The challenge would be to know how much information can be inculcated in the dataset, that is, a complex math problem requires an in depth analysis and all set of mathematical formulae. It also depends on how well the model interprets the problem and if it is able to gain human like understanding which is a difficult task. All we are trying to do is understand what problems the model is facing while answering certain question and try to prompt it accordingly.

2 Related Work

GSM8K (Cobbe et al., 2021) (2) emphasized on multi-step mathematical reasoning. The dataset consists of 8.5K high quality linguistically diverse grade school math word problems. They find that even the largest transformer models fail to achieve high test performance, despite the conceptual simplicity of this problem distribution. To increase performance, they proposed training verifiers to judge the correctness of model completions. At test time, many candidate solutions were generated and the one selected was ranked highest by the verifier. They demonstrated that verification significantly improves performance on GSM8K. Our dataset, that is, the JEEBENCH contains advanced math problems, where the GPT-4 model was able to solve only 20% when compared to solving 80% of the MATH problems of GSM8K. This shows that there are still bigger problems that needs to be addressed.

Dolphin18K (Huang et al., 2016) (3) considers a huge dataset of size 18k, which was obtained from CQA (community question answering) web pages like the math category of Yahoo. The only problem here is that, even though a model like SVM was used to extract answers, it doesn't have data where it can show with absolute proof that the correct answers were extracted. This is where the JEEBENCH dataset makes the difference, where it has solutions to advanced math solving problems and can prompt the machine learning model in such a way where it can solve more complex problems.

MATH (Hendrycks et al., 2021b) (4), though fell short of what they wanted to present paved a way on how to proceed further with research in solving math problems using large language models. They considered 12,500 problems from high school math competitions which relied on long horizon reasoning, that is, a step by step solutions were included in the dataset, that can be used to train the machine learning model. Even though, they were successful in collating the dataset, they were able to achieve very low accuracy on the real world math examples. This shows that even further research is required while choosing a set of problems and the type of prompts that needs to be asked to a model.

3 Experiments

3.1 Datasets

The authors in their GitHub repository provide the dataset which comprises of 515 questions from JEE Advanced from 2016 till 2023. The authors have acknowledged the efforts of Dr. Parag Singla of the JEE Office with trying to get approvals for research purposes. Apart from that, questions are freely available on the JEE Advanced website. [1]

3.2 Implementation

The implementation was cloned from the original repository. It was then studied to understand the structure of the repository and then the code itself. Since the dataset was not huge, it was relatively easy to go through to understand the format in which it was being passed to the model.

The code is executed in two phases: First the model is given (using OpenAI's API) questions in the form of prompts and then the model responds which is recorded. From this response, the exact answer, depending upon if the question is of MCQ, MCQ(multiple), Integer or Numeric is extracted. In the second phase, the model is evaluated based on the extracted answer which is compared to the gold key in the dataset.

Models evaluated in the re-run include GPT-3, GPT-3.5, GPT-4-0613 under different modes such as Normal, CoT (Chain of Thought), One shot and Self-Refinement.

Due to reasons that will be discussed in later sections, only 114 of the 515 responses could be recorded and those we used to evaluate the models. The code was originally meant to have all 515 responses used which was not possible. Also, only GPT-4-0613 was generated again by the team whereas the rest of the data from GPT-3, GPT-3.5 and other models were used from the original data

[1] https://jeeadv.ac.in/archive.html

given by the original authors since it was not possible to generate them.

The authors code was changed to accommodate the 114 responses being used and then it was evaluated in the second phase.

The github link for the same is pasted below https://github.com/somethingdevs/cs678-jeebench.git

3.3 Results

Following is the table of which compares the different models with evaluation

3.4 Discussion

The results show that our re-run of the experiment yields numbers similar to the authors', in fact slightly better. There might be various factors contributing to this. Our dataset consisted of 114 responses instead of the authors' original 515. Due to this, the dataset could have been more skewed towards certain types of questions. We note that the models perform better on single correct and multi-correct multiple-choice questions. This is evident in the author's work and in ours as well. So, the subset we considered could have contained more of those questions.

Overall, GPT-4 outperformed GPT-3 and GPT-3.5 in terms of overall problems solving. Chemistry was clearly the subject that got the most correct responses. This could be due to Chemistry generally being more concept-based and not having as much problem solving as Math and Physics. Math performed the worst, which is expected since JEE Advanced questions are typically heavy on problem-solving.

Moving forward, we did face a couple of difficulties along the way. Initially, when we tried to compute the results, we always seemed to get the same numbers as the author. We studied the code related to result generation and discovered that the code kept producing results with the same data that the author had generated once. The API was not called again, for obvious reasons, i.e., cost associated with API token usage. To address this and gain a new perspective on the evaluation of the models, we ran the GPT-4-0613 model with a total of 515 questions given as prompts.

The volume of prompts led to us being rate-limited and also incurred extra costs, which allowed us to complete only 114 questions. As a result, we were unable to generate our own responses to the dataset for the other models, hindering us from conducting any form of sensitivity analysis.

The next issue was that the code expected all models to run with 515 data objects being presented to them. This meant that the code would throw errors for the GPT-4-0613 model. To solve this, we essentially trimmed down the number of questions used to evaluate the models. Instead of using all 515 objects, we used 114 objects, but all the models were able to run on them. These were the main fixes that we had to make to the code to allow it to run with the available amount of data.

3.5 Resources

The paper initially seemed pretty straightforward to recreate. However, when we attempted to generate our own responses without using the authors', we encountered rate limitations and costs. Generating 114 responses from GPT-4-0613 cost us roughly $5, so the amount it could take to generate 515 responses from 6-7 models would have been rather significant.

In terms of computation, the generation of responses took roughly 10 minutes, which is not very significant. The main time spent was behind fixing the code to only take our limited responses instead of the entire dataset. Apart from that, it took us multiple days just to go through the list of papers from EMNLP 2023 and narrow down our options. Since our team consists of 2 people, the work progressed a bit slower, but due to the nature of and interest in the paper, we were able to finish things on time.

During the paper selection phase, a few papers did not provide datasets that could be used. Additionally, we encountered some issues when attempting to install the requirements and discovered some content that was specific only to the authors' local environment. This prompted us to message some of the authors (one actually responded and thanked us for showing interest in their work!).

Finally, development time was not as heavy as it should have been. However, the time taken to fix bugs due to dataset issues did hinder progress. We also spent hours manually extracting the single correct answer from the model response.

3.6 Error Analysis

Error analysis was conducted on the code using two primary methods. Firstly, we analyzed the responses generated by the model. These responses,

Model	Chem	Math	Phy	Integer	Single-Correct	Multi-Correct	Total
Random	0.141	0.14	0.137	0.000	0.250	0.135	0.139
GPT-3	0.128	0.082	0.113	0.038	0.143	0.113	0.105
GPT-3.5	0.358	0.141	0.194	0.192	0.2	0.259	0.226
GPT-4	0.52	0.288	0.419	0.192	0.514	0.425	0.399
GPT-4+CoT	0.655	0.402	0.573	0.385	0.571	0.575	0.531
GPT-4+(1-shot) CoT	0.493	0.283	0.452	0.269	0.543	0.363	0.397
GPT-4+CoT+Self Critique	0.703	0.359	0.516	0.423	0.514	0.557	0.513
GPT-4-0613	0.628	0.223	0.315	0.154	0.514	0.401	0.379

Table 1: Comparison of model results

which were answers to the dataset's questions, revealed several outliers. A limit of 2048 tokens was imposed on the model's responses. Responses exceeding this limit were truncated, and their values were set to 'None'.

Secondly, the reduction in dataset size introduced new errors due to the code not accommodating varying response lengths. We adjusted the code to handle the reduced response lengths consistently across all models. This modification not only highlighted the vulnerability of the system when provided with insufficient or potentially biased data but also led to inflated performance metrics that may not accurately reflect the model's reliability.

4 Plan for Checkpoint 2

For Checkpoint 2, we plan to going down the multi-lingual path track. Normally, the JEE Advanced paper is offered to students in English and Hindi. However, India is a much more culturally and linguistically diverse land with many regional languages. Our own mother tongues (Urdu and Telugu) are not offered, so we decided to proceed with trying to convert the dataset to some common languages like Hindi, Urdu, Telugu, and Chinese. To reduce the costs from the API usage, we might need to cut down the dataset and the number of questions we translate.

5 Checkpoint 2 Approaches

5.1 Approach for training a multilingual model

For the multilingual model, we used three types of models: GPT-3.5 Turbo, GPT-4 0613, and mBERT. For mBERT, we built it from scratch using pre-trained multilingual-bert-base-uncased and its Tokenizers. To create the dataset, we used 3 ways, DeepL Translate, Google Translate and GPT-3.5 turbo. We majorly used GPT-3.5 to convert it into different languages by the help of a Python script that gave it the prompt and question to translate. This was done on only 100 questions from 102 to 202 of the 515 questions in the dataset. This was uniformly done over all the languages.

Since the testing data was already separated, we then split the training and validation data on an 80-20 basis. The first step of the pipeline was to preprocess the data. We used Pandoc to remove LaTeX from the dataset. Then, we categorized and one-hot encoded the answers into categorical variables. Next, we tokenized the questions using the Transformers library. The input features and labels were converted into PyTorch tensors and used to create TensorDatasets and DataLoaders.

The hyperparameters were chosen through trial and error. Although all the values chosen resulted in roughly similar outcomes, the batch size was kept low due to its high computing usage.

Metrics used to evaluate the model include loss, calculated via cross-entropy, and accuracy. We also conducted separate evaluations based on language, calculating accuracy and loss for each language as well.

5.2 Approach for analyzing a multilingual model

First, we identify the language of each question in the test set. Once the languages are identified, we split the original test set into subsets based on language. For example, create separate test sets for each language present in the dataset (e.g., English, Spanish, Chinese & Arabic). We ensure that each subset contains questions only in the corresponding language. We then tokenize and encode the questions in each language-specific test set using BERT's multilingual tokenizer.Next, we compare the model's performance across different languages to identify any variations or language-specific challenges. We look for patterns in the model's predictions and errors across languages to gain insights into its multilingual capabilities. Depending on the results, we considered fine-tuning

or adapting the model for specific languages where performance is sub-optimal. Fine-tuning involves further training the model on language-specific data to improve its performance on that language. Finally, it's essential to continuously monitor and evaluate the model's performance on new data such as Hindi. This ensures that the model remains effective and generalizes well to diverse linguistic contexts.

6 Checkpoint 2 Experimental Results

6.1 Multilingual Model Performance

As seen in the table below, GPT-3.5 Turbo and GPT-4 0613 Normal performed well in categorizing problems in Hindi. However, our mBERT model displays a notable limitation in handling math-related tasks. This is particularly because questions involving numeric or integer responses are treated as classification problems rather than regression tasks, which can lead to less precise outcomes. However, given that such questions constitute a small proportion of the dataset, we proceeded with this approach for its simplicity and practicality.

GPT-3.5-turbo and GPT-4-0613 both do bad overall at Math tasks as well. This is explained in the paper with the reasoning that Math involves Concept Recollection on top of Problem Solving. These models don't do extremely well for advanced level problem solving. Its for the same reason that GPT models score the highest in Chemistry. Keep in mind mBERT is trained on 415 rows of data and then tested on 100 rows whereas GPT is already trained and just tested on 100 rows.

6.2 Analysis of Model Multilinguality

A careful error analysis of the models GPT-3.5 Turbo and GPT-4 0613 reveals the following issues: In the Arabic language, Chemistry scores the lowest and Math the highest, which is contrary to what is normally expected. An inspection of the responses indicates extensive incorrect use of LaTeX, which may have contributed to incorrect answers. Additionally, the right-to-left (RTL) nature of Arabic could further complicate the rendering of LaTeX. Finally, many questions did not translate well from English and this is visible with many questions being left in English. In the Chinese language, the model performs decently. LaTeX converts well and integrates smoothly with the translated text. However, the scores for numeric-type questions are very poor, likely due to the model's inability to process problem-solving effectively. In the English language, the model performs best, adhering to the general trends discussed during the first checkpoint. It does, however, perform poorly in the numeric section, though it still outperforms all other languages. In the Hindi language, the model performs very well, ranking second to English across multiple sections. Questions translated from English to Hindi yield consistently stable results. There are a few irregularities with LaTeX, but they are not significant.

For the mBERT model, the issues of translation was common among all the languages. On top of this, data pre-processing could have been done better which could have lead to much better resutls.

7 Discussions

We had to manually inspect the quality of the dataset due to issues caused by LaTeX formatting. Concerned that mBERT might not be able to interpret these formats, we converted the dataset into a list of dictionaries. LaTeX was then removed from the text using Pandoc, but this process was applied only to the English data, not to all languages. In the future, the text should be "de-LaTeXed" before translation to enhance accuracy. We also should have treated the questions in a bit more complicated manner. Instead of treating them all as classification problems, Integer and Numeric should have had regression applied on them. To improve the overall quality of the dataset, manual translation from experts would could significantly boost model accuraucy as well. Given that the JEE Advanced exam is one of the most challenging exams, a larger training set is required. The model needs extensive training on these questions, as the exam presents new challenges each year. Therefore, it is crucial to continually expand the dataset to include new additions and concepts. The dataset size is limited because the JEE Advanced exam was only recently converted to digital format, resulting in a novel dataset.

8 Workload clarification

In the first checkpoint, Both Ali and Anjan worked equally on sifting through the list of conference papers. Ali mainly focused on development and

Language	Accuracy
Chinese	0.0732
Arabic	0.0642
English	0.0642
Hindi	0.0732

Table 2: mBERT Results

Subject	**Model**	**English**	**Arabic**	**Hindi**	**Chinese**
Chemistry	GPT-3.5	0.23	0.128	0.182	0.27
	GPT-4	0.473	0.203	0.338	0.453
Physics	GPT-3.5	0.268	0.115	0.33	0.232
	GPT-4	0.42	0.200	0.25	0.33
Math	GPT-3.5	0.179	0.143	0.407	0.236
	GPT-4	0.264	0.364	0.436	0.343

Table 3: Model Accuracy Across Different Languages and Models

bug fixes, while Anjan concentrated more on writing a clean report. Ali contributed to the third section as well as the latter part of the discussion since that aligned more with his contributions. Anjan researched related works and facts associated with the paper, extracted more information, and completed the remaining parts of the paper.

In the second checkpoint, Anjan focused on accumulating the dataset. He helped translate and manually annotate the data in Arabic, Chinese, and Hindi. Ali assisted in translating the data using GPT-3.5 Turbo. He gathered the metrics and generated the tables. Finally, he implemented the entire pipeline to build the mBERT model from scratch and analyzed its results. He also refactored the code and updated the instructions for easier reproducibility.

9 Conclusion

The paper "Have LLMs Advanced Enough? A Harder Problem Solving Benchmark For Large Language Models" by Arora et. Al focuses on evaluating different LLM's using questions from the JEEBench dataset, which includes problems from India's JEE Advanced exam. The paper was reproducible with certain limitations. The findings indicate that GPT-4 significantly outperforms GPT-3 and GPT-3.5. Additionally, results showed that Chemistry resulted in the most correct answers among all the models. This could be attributed to the fact that Chemistry generally contains concept focused and lesser problem solving questions than the others.

In Checkpoint 2, we built the mBERT classifier from scratch. To create the dataset, we used three methods: DeepL Translate, Google Translate, and GPT-3.5 Turbo. We primarily utilized GPT-3.5 to convert it into different languages using a Python script that supplied the prompt and question for translation. This process was applied to a set of 100 questions out of the 515 questions in the dataset and was uniformly conducted across all languages. We focused mainly on the Hindi language, as the exam is conducted in both Hindi and English. As observed from the tables, our model did not exhibit much accuracy compared to large language models such as GPT-3.5 and GPT-4. This is primarily because the GPT models have already been extensively trained, whereas our model was trained on 415 rows of data and tested on the remaining 100.

References

[1] D. Arora, H. G. Singh, and Mausam, "Have LLMs Advanced Enough? A Challenging Problem Solving Benchmark For Large Language Models," *arXiv preprint arXiv:2305.15074*, 2023. [Online]. Available: https://arxiv.org/abs/2305.15074.

[2] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, C. Hesse, and J. Schulman, "Training Verifiers to Solve Math Word Problems," *ArXiv*, vol. abs/2110.14168, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:239998651

[3] D. Huang, S. Shi, C.-Y. Lin, J. Yin, and W.-Y. Ma, "How Well Do Computers Solve Math Word Prob-

lems? Large-Scale Dataset Construction and Evaluation," *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pp. 887–896, Berlin, Germany, August 7-12, 2016.

[4] D. Hendrycks, C. Burns, S. Kadavath, A. Arora, S. Basart, E. Tang, D. Song, and J. Steinhardt, "Measuring Mathematical Problem Solving With the MATH Dataset," *arXiv preprint arXiv:2103.03874*, 2021. [Online]. Available: https://arxiv.org/abs/2103.03874.