PRECOG RECRUITMENT TASK

Tasks are centered around four central themes

- NLP & Responsible AI
- Computer Vision
- Graphs
- Math & AI

Applicants must choose **any one** theme and complete the task listed under that theme. The task consists of two parts - a programming task and a paper reading task. You must attempt both tasks.

You are encouraged to rely on the existing literature, blogs etc to inform your design choices; we only recommend you to write and justify your choices / assumptions. Note that you are absolutely not required to use the architectures provided in the task descriptions, if any. Develop your own! Make modifications, try something new! Whatever you try and fail at, write it in your report/presentation.

For some tasks there are bonus tasks - to demonstrate that you can go the extra mile (which is an important characteristic of being in our group)! We encourage you to try those bonus tasks.

To overcome compute constraints:

- You can use Google Colab, Kaggle for compute intensive jobs.
- Take a subset of data if the dataset is huge - ex. 25% / 50% / 75% of train data

You must attempt this task on your own. Please make sure you cite any external resources that you may use. We will also check your submission for plagiarism, including ChatGPT :)

Submission:

- Put all your code/notebooks in a GitHub repository. Maintain a README.md explaining your codebase, the directory structure, commands to run your project, the dependency libraries used and the approach you followed.
- The link to the GitHub repository will be asked for during the interview.
- Presentation / Report:
    - Programming Task : Document the process and compile a presentation / report summarizing your findings, methodologies, and any insights gained from the analysis in a presentation/report. Your report must contain your methodology, findings, results etc. On the first page, It must detail exactly what parts of the task you did, and what parts of the task you did not do and why.
    - Paper Reading: For paper reading task we expect you to create a video that summarizes your analysis of the paper - video should be less than 5 mins. You can use PowerPoint/Loom to quickly record the video over your slides. Keep the video/slides simple, emphasize the technical content, rather than production

quality. We are keen to understand your understanding of foundational concepts. Your analysis could include one or more of the following: your thoughts around the summary of the paper, major strengths, weaknesses of the paper, generalisability of these techniques, limitations, extended research directions based on the paper, methodological insights et cetera. To get you started, you can answer the following questions - only to get you started, you are free to improvise, BE CREATIVE!

Evaluation:

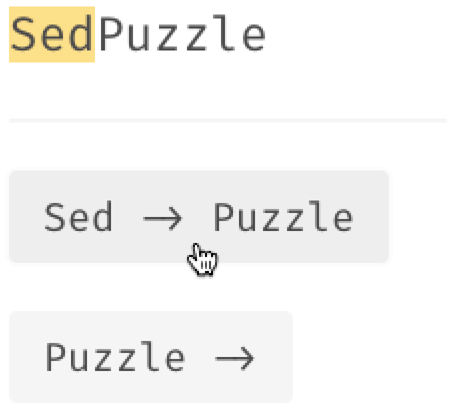
- You will be evaluated based on how you approach the problem, and not so much on the performance measures like accuracy etc. Although, we would expect you to beat random performance.
- How you present your code and findings. You should justify all the choices you make regarding data, model, and hyperparameters that you use. You should also be able to demonstrate theoretical understanding of the approaches used.
- Across all the tasks, your experiments will give you some quantitative measures to indicate the efficacy of your approach (Accuracy, Recall, MAE, MSE etc) - but dig deeper to analyze the predictions. Can you come up with any hypothesis for why your approach fails/succeeds? Can this analysis help you improve on your approach? Creativity in this analysis is what we are looking for!! - SURPRISE US!!.
- How do you handle and sample from large real-world datasets, and solve tasks with whatever resources you have access to..

For any doubts regarding this task please directly email to the address provided for each task, add the following email in cc - debangan.mishra@students.iiit.ac.in. If you don't get a reply within 24 hrs feel free to drop a reminder.

4. s/Math + AI/ Reasoning in LLMs

“A smooth sea never made a skillful sailor” ~ Franklin D. Roosevelt

The Unix tool `sed` is a simple yet powerful tool that can be used for pattern matching and text replacement. Amusingly, someone created a puzzle site, where you need to arrive at the blank string by replacing some text according to some patterns. We highly recommend you try some puzzles to get a feel of the problem before continuing.



Done? Now for the task…

Task 1: Dataset Curation

“It is a capital mistake to theorize before one has data.” ~ Sir Arthur Conan Doyle

To understand if LLMs can solve puzzles like the sed-puzzle, we need to collect a large enough collection so that we can reduce, if not eliminate, any statistical biases that exist in the data.

For this task, you need to create a dataset that is representative of the problems available on the sed-puzzle website. The puzzles in your dataset should have a variety of difficulty levels. Determining the difficulty is left to you ;)

Note: Do not scrape the internet for puzzles, we want you to *write your own generator* for it. This also means no creating puzzles by hand, except for a tiny fraction. Also, it is very likely that the puzzles have been scraped by companies that are training foundational models, so scraping them may not provide an accurate evaluation of the abilities of the language model

A few pointers to keep in mind for dataset generation:

- It should contain an initial state (non-empty string) and a non-zero list of possible transitions. The final state is always a blank string.
  Example: [Hello world, sed puzzle]

  Input: HELLOWORLD

  Available transitions: HELLO -> “”, WORLD -> “” (“” corresponds to empty string)
- Each datapoint in the dataset should be a valid puzzle (i.e. should have a solution, even if it may be very long). Invalid puzzles will be penalised. [Hint: how can you create a solution for a given puzzle?]
- The data points in the dataset should be saved as a JSON file. We have provided some starter code to do the same, but you are free to use that and/or modify it to suit your needs.
  For the above example, the corresponding JSON file will be

  ```
  {
        “problem_id”: “000”,
        "initial_string": "HELLOWORLD",
        "transitions": [
              {
                    "src": "HELLO",
                    "tgt": ""
              },
              {
                    "src": "WORLD",
                    "tgt": "”
              }
        ]
  }
  ```

- Keep the data points in the dataset limited to 100. You can generate more points if you wish, but we would like you to pick the 100 most representative data points to form your dataset.

Your dataset will be judged on the quality as well as the quantity of the dataset. There should be sufficient examples that are challenging to simple uses of LLMs; otherwise, it can give a false portrayal of the LLM capabilities.

To help you read and write puzzle files, we have provided some starter code, which can be found at https://github.com/precog-iiith/sed-solver.

Task 2: Getting LLMs to solve your puzzles

“A bad workman always blames his tools” ~ Apocryphal saying

LLMs are often bad at reasoning without additional effort. However, over the past years, there has been substantial research in trying to get them to logically reason about problems.

Your task is to evaluate the various methods of prompting on your puzzles. You need to try at least the following techniques and benchmark them across various samples from your dataset (how you implement them is up to you):

1. Zero-shot prompting
2. Few-shot prompting
3. Chain of Thought [CoT] prompting
4. Be creative :)

Make sure to document your progress as you go. Make sure to keep track of what prompts you submit, and what you observe. What sort of problems became solvable and what weren’t? This will be as important as the final results itself. Even if you do not have a positive result, we are okay as long as there is evidence of rigour.

Note: All experiments are to be performed on “traditional” (for the lack of a better word) LLMs. **Do NOT use o1/DeepSeek r1/qwq/other** models with “reasoning” for this. If you wish, you can add it as additional benchmarks.

You can approach this task in two ways:

1. **Prompting through web interfaces** (ChatGPT/Claude/Gemini): You can prompt various language models over available web/graphical interfaces and compare the results of the various models and prompts. You can attach screenshots of prompting in your report. Try comparing different models with different prompts and see what models perform better with what prompts and inputs.
2. **Prompting through APIs** [*Highly recommended*]: Gemini has a free (with limitations) API which you can use without having to add your credit card. We *DO NOT* recommend paying for API access to any provider. If you are able to run other LLMs locally, you are free to add them to your report, however, the inability to do so will not be held against you. The API documentation for Gemini Flash 1.5 can be found here.

To avoid rate-limits, you can try perfecting your prompt through the web interfaces before going the API route.

Your solutions should be saved as JSON files as well (you can use the starter code for this!). Using the previous example, the solution that we would expect would take the following form:

```
{
      “problem_id”: “000”
      “solution”: [0, 1]
}
```

The file should contain a solution array containing the **indices** in which the transitions are performed from first to last (i.e. first apply `HELLO -> ‘’` and then apply `WORLD -> ‘’`). Note that even the following is a valid solution to the above puzzle:

```
{
      “problem_id”: “000”
      “solution”: [1, 0]
}
```

The goal is to find any list of transitions that makes the initial string empty and *not* to minimize the number of transitions. To help compare with classical approaches, we have provided a baseline implementation in the starter code.

Task 3: Deriving a metric for evaluation

“In our lust for measurement, we frequently measure that which we can rather than that which we wish to measure...and forget that there is a difference.” ~ George Udny Yule

We need a way to compare different methods and models with each other. Try to come up with some quantitative way to assign a score to each output returned by LLMs and report it. What works and what does not work? Give examples of how your metric performs in different scenarios.

[Task 4 - Bonus]: Man v. Machine

“The supreme consideration is man. The machine should not tend to make atrophied the limbs of man.” ~ Mohandas Karamchand Gandhi

Can you find some examples that are easy for humans to solve, but difficult for LLMs to solve? What about the other way around?

Paper Reading Task: Solving olympiad geometry without human demonstrations

Have fun!

For any doubts regarding this task please directly email - sumit.k@research.iiit.ac.in, with cc to debangan.mishra@students.iiit.ac.in.