The computational trend of NLP research is shifting from feature engineering to representation learning to pretraining-finetuning to very recently prompt engineering with large language models (LLM). Large language models (or other generative models trained on other modalities) allow the extraction of diverse and intrinsic knowledge from human-written texts/images/videos and their pairs. This assignment requires you to **explore the limits and capabilities of large language models** by designing your own prompts to interact with LLMs, observing their outputs, understanding their shortcomings, or creating your own datasets.

Follow the steps below and submit your prompt JSON file and PDF report to Canvas. Below are three steps you have to follow where each step has specific deliverable to submit.

- Step 1: Choosing Models
- Step 2: Understand Current Prompting Techniques
- Step 3: Designing Your Own Prompts

Before you start the homework, you are encouraged to understand the course material on prompting and augmenting/instructing LLMs. The lead TA for this assignment is Shirley Hayati (hayat023@umn.edu). Please communicate with the lead TA via Slack, email, or during office hours. This homework is team-based. Your team should work together.

Step 1: Comparison between Large Model vs. Smaller Model

In this section, you must choose at least one local model to use and one commercial model to use (e.g., ChatGPT, Gemini, Claude, Deep). For example, you can choose Qwen-2.5 as your local model and Gemini for your commercial model. You can use **LangChain** to compare and contrast multiple local LLMs, but it's not mandatory. (With LangChain, it is also possible to use APIs from OpenAI/Anthropic/TogetherAI/etc., but we will not be doing that for the purposes of this assignment.)

Commercial model Google Gemini offers free API to use with limited usage. You can access it with Google AI Studio (https://aistudio.google.com/welcome). OpenAI's ChatGPT (https://chat.openai.com/chat)'s interface version is currently free and you should be able to login using your usual Google credentials. For OpenAI API, we are currently in the process of getting instructional funding approved, so we advise you to use free commercial models like Gemini or use the interface version of ChatGPT if you still prefer ChatGPT.

Here is a list of examples tasks provided by OpenAI[1], such as Question Answering, Summarization, and Text-to-Command. Try playing around with those existing prompts and examine the outputs. For those using free ChatGPT, take a look at these resource for examples:

- ChatGPT prompt book: https://lifearchitect.ai/chatgpt-prompt-book/

- Official Prompt Engineering guide by OpenAI: https://platform.openai.com/docs/guides/prompt-engineering

Note that for your submission, you are not allowed to use any online examples, whether provided here or not.

Comparing LLM performances This assignment requires you to compare two LLMs: ChatGPT and one open-sourced smaller model such as LLaMA, Qwen, Gemma, Mistral, Phi, Alpaca or Falcon. For the smaller models, make sure you choose the latest instruction-tuned version (e.g., LlaMa 3.2-3B-Instruct or Qwen2.5-0.5B-Instruct)

[1] https://beta.openai.com/examples/

When loading weights from open-sourced models like LLaMA3.2, we suggest loading 3B or less-sized models. In particular, we recommend looking at Llama3.2-1B-Instruct, Llama3.2-3B-Instruct, Qwen2.5-0.5B-Instruct, Qwen2.5-1.5B-Instruct, and Qwen2.5-3B-Instruct.

Your task is to make five pairs of a prompt and expected output: (Prompt, Expected Answer). Then you must compare and report the output from the two models using "their APIs." For example, a prompt-answer pair could be ("Task: is this text positive or negative? \n Input: Today's weather is sunny and great! \n Output: ", Positive) where the actual output from ChatGPT is (Negative), which doesn't match my expected answer. You are not allowed to use any examples in this homework description or any existing papers/datasets/the Internet. Be creative!

You must discuss the differences and similarities in the outputs of each model.

Step 2: Understand Current Prompting Techniques

This assignment requires you to understand the state-of-the-art prompting techniques. In addition to simple zero-shot and few-shot prompting, refer to the following articles and papers (you can find links to the original articles in the Reference section):

- Some prompting tricks like Chain-of-thoughts [WWS+22b], Tree of Thoughts (ToT) [YYZ+23], self-consistency [WWS+22a], reAct [YZY+22], self-refine [MTG+23], and more, and applications to human-GPT3 collaboration for text editing [RKKK23] and poetry writing [CPH22].
- Stress test of GPT3 on various aspects: commonsense reasoning [MD20], hypes and ethics [BGMMS21], and planning [VOSK22].
- Discrete and soft prompting methods: Auto-prompting methods [SRLI+20, ZWF+21] and Prefix/prompt-tuning [LL21, LARC21][2]
- Risks of using LLM-generated data for NLP tasks [DLMB+24, DQL+22, MDPA23, KLR+23]

Your task is to read some of the papers (minimum 3 different prompting techniques) from the list above and understand the differences and limitations of those methods. Write one section (2-3 paragraphs) in your report about comparison of the prompting techniques you choose.

You may also find *Prompt Engineering Guide* and *Learn Prompting* useful for grasping the overall picture of the field.

[Optional] You can try those prompting techniques as you wish and report results or failure cases. Bonus point will be given based on the creativity of the tasks and findings of failure cases.

Step 3: Designing your own Prompts

The last step involves designing your own creative prompts for a task! You can choose one task among the following options:

- Task 3a: Data generation
- Task 3b: Role-playing

You must design and discover your own prompts. Please be creative! We will compare the similarity of your prompts with any publicly available prompts in the links provided or other links we didn't provide. If we find similar prompts in our database, your submission will be considered cheating.

[2]These methods require fine-tuning of large language models so I don't recommend to use them for this assignment

You should make use of **at least two different types of prompting techniques** for each category you choose from above. For example, if you choose to use prompting for "data generation," the two types of prompting techniques could be: "few-shot" and "self-refine." In total, you need to create at least **1 task category** x **2 types of prompting** x **50 prompts per type** = **100 prompts**. Each "prompt" here can be a unique pair of input-output to generate new data points (task 3a) or a combination of different personas (task 3b).

Store all prompts and outputs from each setup, and submit them in your JSON file.

You can choose one model from Step 1.

(Task 3a) Data generation for complex tasks Large-scale LLMs, such as ChatGPT, are useful for generating new synthetic data for model distillation, which is training smaller model with training data generated by stronger LLMs (e.g., ChatGPT). Your goal is to generate new dataset for one specific task. The task could involve instruction-tuning data (e.g., Alpaca) or focus on a highly specific domain, such as academic writing, mental health conversations or legal processing, where obtaining real-world data is particularly challenging.

Here, you write at least 50 seed input-output pairs and prompt an LLM to generate at least 500 more instances.

- **Iterative data generation**: Some relevant papers related to LLM for data generation Self-instruct, Alpaca.
- **Challenging tasks**: You must design a task which are challenging for LLMs to tackle, such as compositional instructions, multilingual instructions, complex constrained instructions. You must not generate dataset for really common NLP problems, such as sentiment analysis, machine translation, general text summarization, named entity resolution, part-of-speech tagging, simple text classification (spam detection, topic classification) etc.
- **Evaluation metrics**: There are different ways to examine the quality of generated data. You can do manual inspection of randomly sampled generated data (e.g., 100-200 instances), implement LLM as judge to examine the data quality, voting by multiple LLMs, or even fine-tune a smaller model with your generated dataset and report the improved F1-score/BLEU/etc over the baseline (non-fine-tuned model). You can select one evaluation metric and justify your decision of choosing that metric (e.g., cite how relevant papers also use similar metric, explain the nature of your designed task.)

In your report:

- Explain why your task is challenging
- Explain what type(s) of prompting technique you use for data generation
- Explain and justify the evaluation metric you use to examine the quality of generated data

(Task 3b) Role-playing LLMs have been widely explored for role-playing, where the model adopts specific personas or perspectives, such as a doctor, teacher, or interviewer, to simulate realistic dialogues or scenario-based reasoning. Effective prompting plays a crucial role in guiding the models behavior, ensuring that it maintains the assigned role and adheres to contextual goals.

In this assignment, your task is to choose an LLM and assign at least 50 different combinations of personas through prompting. Each persona can include basic demographic information, such as country, age, gender, or educational background, and other history of the person's activities. You may also add task-specific persona attributes, such as profession, intentions, background context, political beliefs, or preferences (e.g., favorite movies or hobbies).

After assigning personas, you will test how the model performs some specific tasks. For example, you can make LLM role-play a classroom setting, where it interacts as a teacher with various persona-based LLM students, adapting responses and behavior according to the assigned persona. Or you can make an LLM makes decision for some hypothetical social situations link. Be creative!

Your goal is to observe how different personas affect the model's ability to perform the task. It is also possible that perhaps your model is unable to follow the given persona you prompt, but you can explain the failure cases.

- **LLMs for role-playing**: Some relevant papers about LLM for role-playing role-play with LLMs, role-play prompting, Character-LLM.
- **Evaluation metric**: There are different metrics you can choose to evaluate the outcome of your LLM role-playing such as
  - persona consistency. This examines whether the model maintains the assigned persona traits throughout the conversations. It could be manual inspection of the conversation content, or examining the style of the dialogue of a teacher LLM (e.g., formality).
  - goal fulfillment. For example, if your LLM role plays as a teacher and students, do the teacher really execute the teaching well? You can make LLM students rate the teacher LLM's teaching quality.

  You can do manual inspection, use LLM as judge, make a final goal for the LLM to achieve, or run an existing classifier (e.g., formality classifier) to examine the role-playing outcome. You can select one evaluation metric and justify your decision of choosing that metric (e.g., cite how relevant papers also use similar metric, explain the nature of your designed task.)

In your report:

- Explain what persona variations you assign to the LLM and what tasks the LLM need to execute
- Explain what type(s) of prompting technique you use for LLM role-playing
- Explain and justify the evaluation metric you use to examine the outcome of the LLM role-playing.

General Advice for Prompting It is NOT permitted to use existing datasets or other sources of data for this particular homework. Check Google for existing or similar examples/prompts before submitting them. When we find the same or similar examples/prompts in other sources, you will lose points. Here are some notes and tips for your prompt design:

- You have to provide a reasonable quality of task description and examples in your prompts, and make sure that LLMs' failure does not come from the quality of your prompt design, but is mainly caused by the lack of inherent capabilities of LLMs. You can find high-quality prompts through trial-and-error with LLMs in free commercial interface (e.g., Gemini, ChatGPT) or in your Python code.
- We are scientists! Try different tasks and prompt examples, and see if LLMs always fails deterministically.
- Once again, you cannot use examples from the Internet, this homework description, predefined prompts, or previous papers. It will be treated as *cheating* if we find the same prompt used before. Note that instructors already have a huge list of adversarial prompts from previous classes. Check the class page for our academic integrity policy
- Here are some additional tips you may consider during prompting:
  - Find novel tasks you/LLMs can't do.
  - Find tasks that LLMs can do a better job than humans.
  - Find cases where even humans do not agree with each other and see how LLMs can handle this human disagreement
  - Find unseen (probably not seen in the training data) but realistic cases
  - Find unseen and unrealistic cases

Deliverable

Please upload the following files to Canvas by Nov 2, 11:59pm:

- JSON file containing your designed prompts and outputs,
- your report (in PDF),
- a Python code or Jupyter Notebook script along with documentation in your report pdf on how to run it.

JSON file Your designed prompts and outputs should be contained in a JSON file and pushed to the homework repository in a way that all your input is visible. Your JSON file name should follow this naming convention: **csci5541-f25-hw5-{TEAM-NAME}-3{a/b}.json** The violation of this format will receive a penalty in your grading.

You can simply create a JSON file using the following script:

```python
import json
data = [{'name': 'John Doe', 'age': 30}, {'name': 'Jane Doe', 'age': 25}]
with open('data.json', 'w') as f:
    json.dump(data, f, indent=4)
```

This code will create a JSON file called data.json that contains the following data:

```json
[{
  "name": "John Doe",
  "age": 30
},
{
  "name": "Jane Doe",
  "age": 25
}]
```

Your prompt consists of a combination of task instruction, examples only if few-shot (i.e., input-output pairs), and input task.

Code a Python file or Jupyter Notebook script that contains your code for evaluating your prompt and outputs, or how you make your agents interact with each other, etc. You must name your Python file with this naming convention: `code_csci5541_f25_hw5_{TEAM-NAME}_3{a/b}.py` or

`code_csci5541_f25_hw5_{TEAM-NAME}_3{a/b}.ipynb`.

- Code implementation of the workflow and how data is collected, etc.
- Code for evaluation metric which takes your JSON file as an input and print the output metric.

Report Maximum four pages PDF total. Your report needs to include the following content:

- Findings from Step 1 and Step 2.
- Explanation of the selected topic (e.g. Task 3a data generation), the problem set up, etc.
- Describe which LLMs you choose for Step 1 and Step 3 and properly cite them.
- Explanation of your prompts and how the LLM was prompted
- Explanation of evaluation metric
- Analysis on the LLMs' outputs and overall results.

- Challenges you encountered during your homework and your general thoughts on language model prompting. What are the takeaways or other interesting things you learned through this assignment?

Rubric: 15 points total

- Report (total: 8 points)
  - Properly cites all papers used in the report (1 point)
  - Step 1: Description of LLMs you tried for this and cite them correctly (1 point)
  - Step 1: Provide five examples of prompts, their expected outputs, and LLM-generated responses, and include a comparison between at least two models (1 point)
  - Step 2: Description and comparison of at least three prompting techniques and properly cite them (1 point)
  - Step 2 and 3: Lists limitations of current prompting techniques (Step 2), and what can be improved with these prompting techniques when applying them to your chosen topic in Step 3 (1 point)
  - Step 3: Problem setup explanation: clearly describe which task you choose (3a/b), proper citation, different prompts you try, evaluation metric (1 point)
  - Step 3: In-depth analysis of the LLM's outputs and comparison among prompts (1 point)
  - Step 3: Discussion of challenges and takeaways (1 point)
- JSON file (total: 5 points)
  - Includes all entities and their values specified (1 point)
  - Follow all formats (1 point)
  - Contains at least 100 pairs of prompts and outputs (2 types of prompting x 50 prompts per type) (3 points)
- Code implementation (total: 2 points)
  - Code runs without errors for the whole pipeline, including data processing, metrics, etc. (1 point)
  - Code contains metric (0.5)
  - Code contains the whole pipeline (prompts, data processing, output generation) (0.5)

Awards: Your designed prompts will also be considered for the following awards and you will receive 1 extra point if you win.

- Best Research Application: Discover a finding that may lead to further research or publication, e.g., a new benchmarking dataset for task 3a.
- Best Creativity Case: Discover a creative scenario and corresponding prompt in your task, e.g., discovering new scenarios for task 3b.
- Best Mistake Case: Discover an interesting case where LLM fails by not giving you the output you want or expect

References

[BGMMS21] Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the dangers of stochastic parrots: Can language models be too big? In Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency, FAccT '21, page 610623, New York, NY, USA, 2021. Association for Computing Machinery. https://doi.org/10.1145/3442188.3445922.

[CPH22] Tuhin Chakrabarty, Vishakh Padmakumar, and He He. Help me write a poem: Instruction tuning as a vehicle for collaborative poetry writing. arXiv preprint arXiv:2210.13669, 2022. https://arxiv.org/abs/2210.13669.

[DLMB+24] Debarati Das, Karin De Langis, Anna Martin-Boyle, Jaehyung Kim, Minhwa Lee, Zae Myung Kim, Shirley Anugrah Hayati, Risako Owan, Bin Hu, Ritik Parkar, Ryan

Koo, Jonginn Park, Aahan Tyagi, Libby Ferland, Sanjali Roy, Vincent Liu, and Dongyeop Kang. Under the surface: Tracking the artifactuality of llm-generated data, 2024.

[DQL+22] Bosheng Ding, Chengwei Qin, Linlin Liu, Yew Ken Chia, Shafiq Joty, Boyang Li, and Lidong Bing. Is gpt-3 a good data annotator?, 2022.

[KLR+23] Ryan Koo, Minhwa Lee, Vipul Raheja, Jong Inn Park, Zae Myung Kim, and Dongyeop Kang. Benchmarking cognitive biases in large language models as evaluators, 2023.

[LARC21] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 3045–3059, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. https://aclanthology.org/2021.emnlp-main.243.

[LL21] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 4582–4597, Online, August 2021. Association for Computational Linguistics. https://aclanthology.org/2021.acl-long.353.

[MD20] Gary Marcus and Ernest Davis. Experiments testing gpt-3's ability at commonsense reasoning: results, 2020. https://cs.nyu.edu/~davise/papers/GPT3CompleteTests.html.

[MDPA23] Anders Giovanni Mller, Jacob Aarup Dalsgaard, Arianna Pera, and Luca Maria Aiello. The parrot dilemma: Human-labeled vs. llm-augmented data in classification tasks, 2023.

[MTG+23] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. Advances in Neural Information Processing Systems, 36:46534–46594, 2023.

[RKKK23] Vipul Raheja, Dhruv Kumar, Ryan Koo, and Dongyeop Kang. Coedit: Text editing by task-specific instruction tuning, 2023.

[SRLI+20] Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 4222–4235, Online, November 2020. Association for Computational Linguistics. https://aclanthology.org/2020.emnlp-main.346.

[VOSK22] Karthik Valmeekam, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. Large language models still can't plan (a benchmark for llms on planning and reasoning about change). arXiv preprint arXiv:2206.10498, 2022. https://arxiv.org/pdf/2206.10498.pdf.

[WWS+22a] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models, 2022.

[WWS+22b] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models, 2022. https://arxiv.org/abs/2201.11903.

[YYZ+23] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models, 2023.

[YZY+22] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2022.

[ZWF+21] Tony Z. Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. Calibrate before use: Improving few-shot performance of language models, 2021. https://arxiv.org/abs/2102.09690.