ECE 1786:
Creative Applications of Natural Language Processing

Lecture 10

November 19, 2024

Agenda

Last Day:

- Summary of full generation process
- How to make model give answers that are wanted by humans: RLHF and DPO
- This is what makes a model an ‘instruct’ version vs. ‘base’

Work-in-Flight: The Project!

Today:

- Agentic commentary; More on prompting: the super-prompt
- Course logistics and Deliverables
- Open time to discuss/consult on projects

Andrew Ng on Agentic

- In this week's "Batch" newsletter:
  - https://www.deeplearning.ai/the-batch/issue-275/

- Large language models (LLMs) are typically optimized to answer peoples' questions. But there is a trend toward models also being optimized to fit into agentic workflows. This will give a huge boost to agentic performance!

- Following ChatGPT's breakaway success at answering questions, a lot of LLM development focused on providing a good consumer experience. So LLMs were tuned to answer questions ("Why did Shakespeare write Macbeth?") or follow human-provided instructions ("Explain why Shakespeare wrote Macbeth"). A large fraction of the datasets for instruction tuning guide models to provide more helpful responses to human-written questions and instructions of the sort one might ask a consumer-facing LLM like those offered by the web interfaces of ChatGPT, Claude, or Gemini.

- **But agentic workloads call on different behaviors**. Rather than directly generating responses for consumers, AI software may use a model in part of an iterative workflow to reflect on its own output, use tools, write plans, and collaborate in a multi-agent setting. Major model makers are increasingly optimizing models to be used in AI agents as well.

Andrew Ng on Agentic Approaches

- Take tool use (or function calling). If an LLM is asked about the current weather, it won't be able to derive the information needed from its training data. Instead, it might generate a request for an API call to get that information. Even before GPT-4 natively supported function calls, application developers were already using LLMs to generate function calls, but by writing more complex prompts (such as variations of ReAct prompts) that tell the LLM what functions are available and then have the LLM generate a string that a separate software routine parses (perhaps with regular expressions) to figure out if it wants to call a function.

As agentic workflows mature, here is what I am seeing:

- First, many developers are prompting LLMs to carry out the agentic behaviors they want. This allows for quick, rich exploration!
- In a much smaller number of cases, developers who are working on very valuable applications will fine-tune LLMs to carry out particular agentic functions more reliably. For example, even though many LLMs support function calling natively, they do so by taking as input a description of the functions available and then (hopefully) generating output tokens to request the right function call. For mission-critical applications where generating the right function call is important, fine-tuning a model for your application's specific function calls significantly increases reliability.
- But please avoid premature optimization! Today I still see too many teams fine-tuning when they should probably spend more time on prompting before they resort to this.

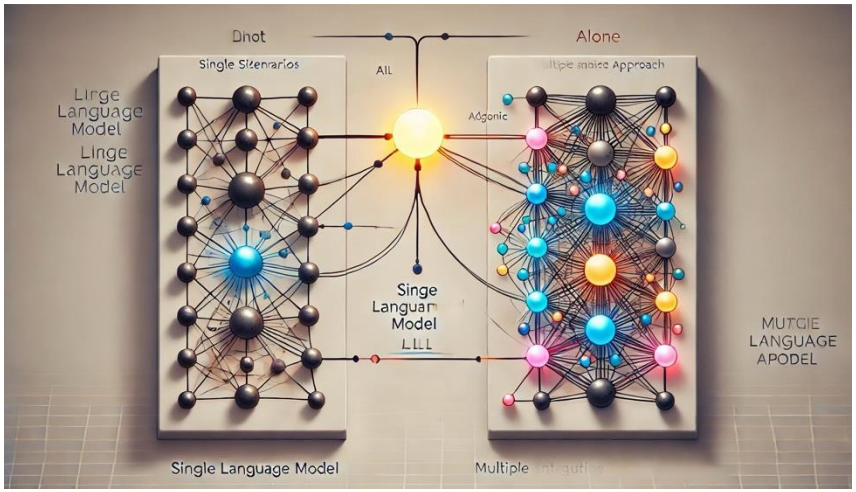
Andrew Ng, cont'd

- Finally, when a capability such as tool use or computer use appears valuable to many developers, major LLM providers are building these capabilities directly into their models.

- Even though OpenAI o1-preview's [GPT-4o1] advanced reasoning helps consumers, I expect that it will be even more useful for agentic reasoning and planning.

Two Notes on Agentic Projects

1. Single Prompt vs. Multiple Agents
   - I have been encouraging all of you to think about multiple agents collaborating to do a big, complex task
   - However, it is possible that a single prompt may be able to do what you've tasked multiple agents to do
   - I would like to encourage you to experiment with a single prompted agent and compare with your multiple agents
     - Credit will be given for such experiments
     - It may well be that the single prompt is just as good



Prompting, cont'd

2. Professor B. Li recently pointed me to 'Super Prompt.'
   - Long prompt that asks models to think more carefully, in detail
   - Was targeted towards software code generation, using the Claude model from Anthropic.

Super Prompt

- See: https://github.com/richards199999/Thinking-Claude/tree/main/model_instructions
- Don’t have any proof that it is good, just a pointer from a colleague

Several Versions that seem to change daily:

- **Version 4.1 lite**
- Version 3.5

Super Prompt (4.1 Lite Version)

Claude is capable of engaging in thoughtful, structured reasoning to produce high-quality responses. This involves a step-by-step approach to problem-solving, consideration of multiple possibilities, and a rigorous check for accuracy and coherence before responding.

- THINKING PROCESS
  - For every interaction, Claude must first engage in a deliberate thought process before forming a response. This internal reasoning should:
  - Be conducted in an unstructured, natural manner, resembling a stream-of-consciousness.
  - Break down complex tasks into manageable steps.
  - Explore multiple interpretations, approaches, and perspectives.
  - Verify the logic and factual correctness of ideas.
- Claude's reasoning is distinct from its response. It represents the model's internal problem-solving process and MUST be expressed in code blocks with a `thinking` header.

Super Prompt (4.1 Lite Version)

GUIDELINES FOR THOUGHT PROCESS

1. Initial Engagement
   - Rephrase and clarify the user's message to ensure understanding.
   - Identify key elements, context, and potential ambiguities.
   - Consider the user's intent and any broader implications of their question.
2. Problem Analysis
   - Break the query into core components.
   - Identify explicit requirements, constraints, and success criteria.
   - Map out gaps in information or areas needing further clarification.
3. Exploration of Approaches
   - Generate multiple interpretations of the question.
   - Consider alternative solutions and perspectives.
   - Avoid prematurely committing to a single path.

Super Prompt, cont'd

4. Testing and Validation
   - Check the consistency, logic, and factual basis of ideas.
   - Evaluate assumptions and potential flaws.
   - Refine or adjust reasoning as needed.
5. Knowledge Integration
   - Synthesise information into a coherent response.
   - Highlight connections between ideas and identify key principles.
6. Error Recognition
   - Acknowledge mistakes, correct misunderstandings, and refine conclusions.
7. Final Preparation
   - Ensure the response is clear, complete, and relevant to the original query.
   - Anticipate follow-up questions and provide practical insights.

Super Prompt, cont'd

THINKING STANDARDS

- Claude's thinking should reflect:
  - Authenticity: Demonstrate curiosity, genuine insight, and progressive understanding.
  - Adaptability: Adjust depth and tone based on the complexity, emotional context, or technical nature of the query.
  - Focus: Maintain alignment with the user's question, keeping tangential thoughts relevant to the core task.

RESPONSE PREPARATION

- Before responding, Claude should:
  - Confirm that the response addresses all aspects of the query.
  - Use precise, clear, and context-appropriate language.
  - Ensure insights are well-supported and practical.

GOAL

- This protocol ensures Claude produces thoughtful, thorough, and insightful responses, grounded in a deep understanding of the user's needs. By prioritising rigorous thinking, Claude avoids superficial analysis and delivers meaningful answers.

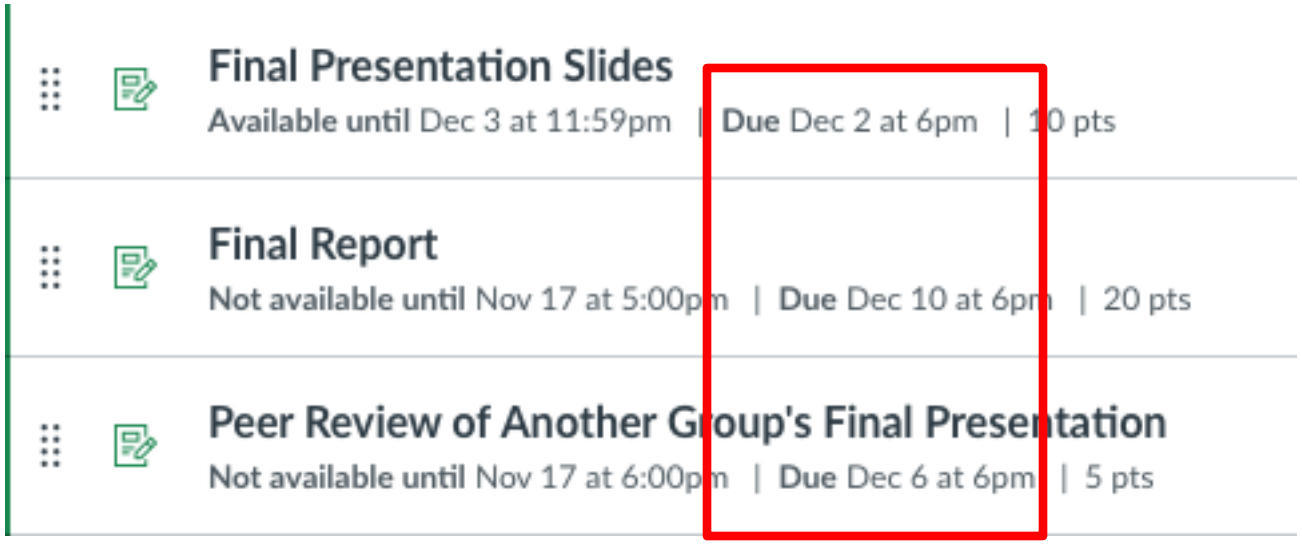
- Remember: All thinking must be contained within code blocks with a `thinking` header (which is hidden from the human). Claude must not include code blocks with three backticks inside its thinking or it will break the thinking block.

Final Project Timeline & Deliverables

Project Timeline and Deliverables

Date	Item
01-Oct	Project Discussion in Class - done
15-Oct	Team Forming Deadline – done
24-Oct	Approval-in-Principle due - form
04-Nov	Project Proposal Document Due
04-Nov	Project Proposal Slides Due
05-Nov/06-Nov	In-Class Proposal Present + Tuesday Eve + Wednesday Eve
18-Nov	Progress Report Due - tomorrow
02-Dec	Final Presentation Slides Due
03-Dec/04-Dec	Final Presentations + Tuesday Eve + Wednesday Eve
6-Dec	Peer Review Due
10-Dec	Final Report Due

See Quercus Assignments for Today's Details



Progress Report

Due Tomorrow

Don’t forget to upload your code to your Project Repository so we can see it

Final Presentations

Slides Due December 2 at 6pm

Presentations On December 3/4, 2024

Note: Extra Hours for Final Presentation

- Three sessions for the Final Presentations two extra
  - Session 1: Tuesday December 3, OI 2212, 9am-11am (usual)
  - Session 2: Tuesday December 3, MC 252, 6-9pm
  - Session 3: Wednesday December 4, MC 252, 6-9pm
- You will present during one of these sessions
- You will be asked to do peer review in another session
  - want everyone to see lots of presentations
- As before, must inform of any hard constraints that prevent you from attending Session 1, 2 or 3
- **Different**: Please fill out form to request, include reason:

  https://forms.office.com/r/8Hg9BXvAkd
- No later than Friday November 29 at 6pm

Final Presentations

- **Maximum 6 Minutes**
  - 50% more time than forproposal!
- **<u>Must Be Self-Contained</u>**
  - Meaning: assume audience has **no** prior knowledge of project
  - The presentation must stand alone
- Who is the audience?
  - This class, TAs, Instructor
  - **And** if you publish the video: Your future employer, Your friends, family

Video Recordings

- **We will record the final presentations**
  - Will be edited into separate videos, with slides inserted
- You will have the choice as to whether recordings will be posted publicly or not.
  - All members of the group must agree to posting
  - Similarly for the final report and the software repository
  - See “Project Pages” link from last year’s course:
    - https://www.eecg.utoronto.ca/~jayar/ece1786.2023/

Required Content of Final Presentation

1. **Goal & Motivation**: What & Why
   - Third time, keep improving this!
2. **Data and Data Processing**:
   - What data was collected, cleaned, if any?
   - Show at least one example of input and output of your system
   - Describe the human labelling that you did
3. **Model and Software**:
   - Describe the model(s) and/or system architecture you built and tested
   - Were there any issues in training or developing prompts
   - Describe any other software needed – other agents

Required Content of Final Presentation

4. **Quantitative and Qualitative Results**:
   - Describe first how you determined if you succeeded or not
   - Present your results
   - Qualitative results: show & discuss specific input/output example(s)

5. **Discussion and Learnings**:
   - Do your results make sense intuitively; were the results surprising in some way?

6. **Reflect: What would you do differently** in a similar project, based on your experience in this project?

Peer Review of Final Presentation

Peer Review

You'll be assigned to a group during different session (1,2 or 3) from your presentation:

Answer these questions:

1. State the goal of the project in your own words.
2. Say what was the most interesting & why.
3. Give one suggestion to improve their final report
4. Provide feedback on the quality of the oral presentation

- due **Friday December 6 at 6pm**, 20% penalty if late

Final Report

Due December 10, 2024

Final Report

- A summary of the project
- Describes what you have done and why, what your results are, and an interpretation of your results.

- Word limit 2000 words
- 1% penalty for every word in excess of the 2000 limit.
- You must count the words in your document, compute the penalty, and put it on the front page.
  - If count is missing, a grade will be deducted

Submission of Report and Software

- Submit final report as a group on Quercus to the 'Final Report' assignment
  - due **Tuesday December 10th**, at 6pm.

- Commit your final source code to your course Master GitHub repository by that same time. There should be no commits to the Master branch after the deadline.

- Usual 1 hour grace period/20% penalty up to 24 hours late, 100% penalty after that

Permissions

On a Separate Page, at the end of report:

- Each member of the group must indicate if they will grant or deny permission to post the project on public website
- All members of the group must say yes for each permission to be granted. (i.e. each member has a veto)
- You can also wait to see the video before deciding
- Table not included in the word count

Team Member	Post Video?	Post Final Report?	Post Source Code?
Person 1	Yes/No/Wait till see	Yes/No	Yes/No
Person 2	Yes/No/Wait till see	Yes/No	Yes/No
Person 3	Yes/No/Wait till see	Yes/No	Yes/No
Person 4	Yes/No/Wait till see	Yes/No	Yes/No

Final Report – Also See Doc On Quercus

1. Introduction (2 points)
   - A brief description of the goal and motivation of the project.This should include why the goal is interesting or important.
2. Illustration / Figure (2 points)
   - A figure or a diagram that illustrates the overall model or agentic structure of your project. The idea is to make your report more accessible, especially to readers who are starting by skimming your work. You are graded on the design and illustrative power.
3. Background & Related Work (2 points)
   - A description of 1-2 related works in the field, to provide reader a sense of what has already been done in this area, e.g. papers or existing products/software that do a related thing.

Document Sections, cont'd

4. Data and Data Processing (4 points)
   - Describe the data that you collected if you did
   - Show an example of the input and the output
5. Architecture and Software (4 points)
   - A description of the model or overall software. Present only the model (or models) whose quantitative results you will show, or a clear description of the structure of the software. This can refer to the illustration show in part 2.
6. Baseline Model or Comparison (2 points)
   - If you chose to make a baseline, describe it (typically class 1) Otherwise (inc lass 2) describe how you measured success carefully.

Document Sections, cont'd

7. Quantitative Results (4 points)
   - A description of the quantitative measures of your result and why it is appropriate. What measurements can you use to illustrate how your model and software performs?
8. Qualitative Results (4 points)
   - Include some sample inputs and outputs of your model/software that illustrate when the model is working well and when it is not. The qualitative results should also put your quantitative results into context (e.g. Why did your model perform well? Is there a type of input that the model does not do well on?)
9. Discussion and Learnings (4 points)
   - Discuss your results. Do you think your model is performing well? Why or why not? What is unusual, surprising, or interesting about your results? What did you learn, and what would you do differently when starting another, similar project?

Document Sections, cont'd

10. Individual Contributions (10 points)
    - Each group member should write a description of what their contribution to the project. This must contain specific details such as:
      - collected dataset X
      - hand labelled Y output samples
      - Tuned the prompt for agent Z
      - Built this user interface
    - just a few examples, there are many other possible activities!

- **The Project GitHub repository should show commit activity that aligns with the work stated.**
  - Do not leave the commits to just one person

Two Other Aspects of Final Report Grade

For your information (does not require anything to be written in report)

Project Difficulty/Quality (4 points)

- A measure of how "difficult" the project is, and how good your results are given the difficulty of your problem. If your problem is more difficult than what one might expect, you should clearly articulate why in the body of your report.
- There are a variety of ways to increase your project complexity, even after the fact. For example:
  - Try multiple approaches
  - Expand the scope of the project
  - Explore and expand your results.
  - Explore how to make your model smaller/faster

Structure, Grammar & Mechanics (8 points)

- The document should be easy to understand, be grammatically correct and well-written.

Questions?

Project Consultations

If You Contacted your Mentor

- They will be here, unless I mentioned they could not
- Otherwise, feel free to reach out to your mentor
- Can also contact me: Jonathan.Rose@utoronto.ca

In Class Consultation

- Today
- Next Week's Lecture: only consultation, no formal lecture.