Lab Session 1: Maha Neta

Palacode Narayana Iyer Anantharaman

25th Oct 2024

What is System 2 Attention Prompting?

- Inspired by Cognitive Psychology: Modeled after System 2 thinking (slow, deliberate, logical reasoning) vs. System 1 (fast, intuitive, automatic thinking).

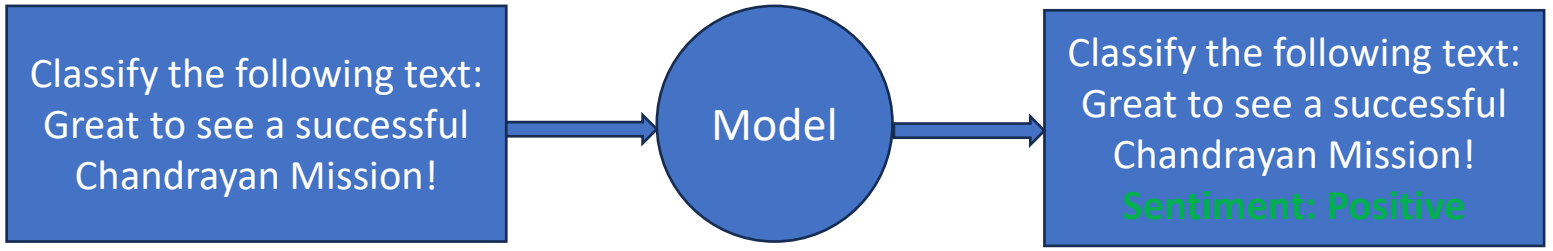
- Encourages Deeper Reasoning: Prompts the LLM to engage in step-by-step, reflective thought processes, leading to more accurate and logical responses.

- Moves Beyond Heuristics: Avoids quick, pattern-based answers by focusing on explicit problem-solving and careful evaluation.

Key Characteristics of System 2 Prompting

- Task Decomposition: Breaks complex problems into smaller, manageable steps.
- Iterative Reasoning: Revisits and refines intermediate conclusions before finalizing.
- Attention to Detail: Focuses on precision, encouraging the model to check for accuracy.
- Useful for Complex Tasks: Effective in scenarios requiring logical analysis, math, or structured reasoning.

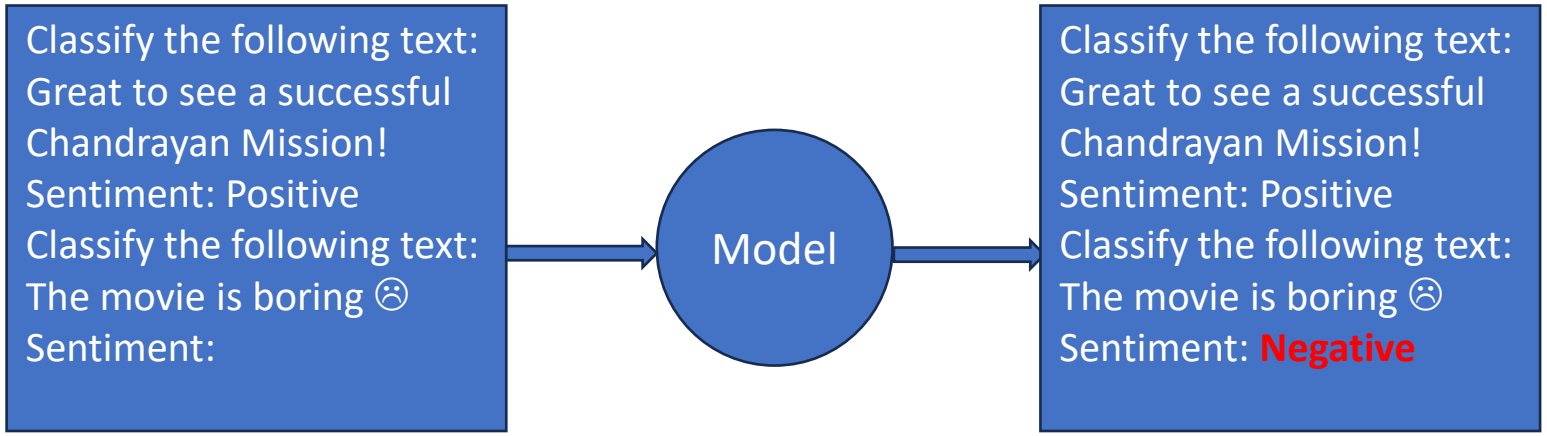
Prompt engineering: Zero shot inference



- Prompt engineering: Providing a prompt that can get the best response

- Zero shot inference: Provide a well crafted prompt to the foundational LLM, use the response from the model as the output. No additional training of the model is required

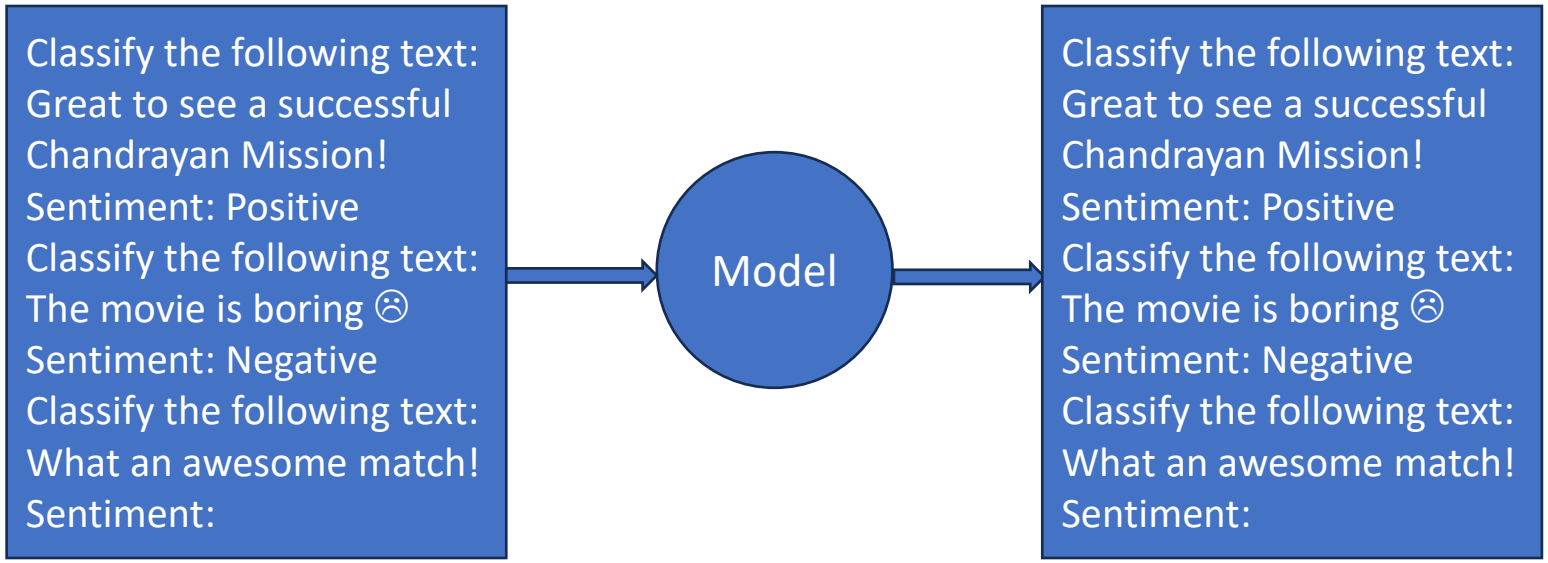
Prompt engineering: One shot inference



- Prompt engineering: Providing a prompt that can get the best response

- One shot inference: Provide a well crafted prompt to the foundational LLM along with one example, use the response from the model as the output. No additional training of the model is required

Prompt engineering: Few shot inference



- Prompt engineering: Providing a prompt that can get the best response

- Few shot inference: Provide a well crafted prompt to the foundational LLM along with a few examples, use the response from the model as the output. No additional training of the model is required

Few shot learning

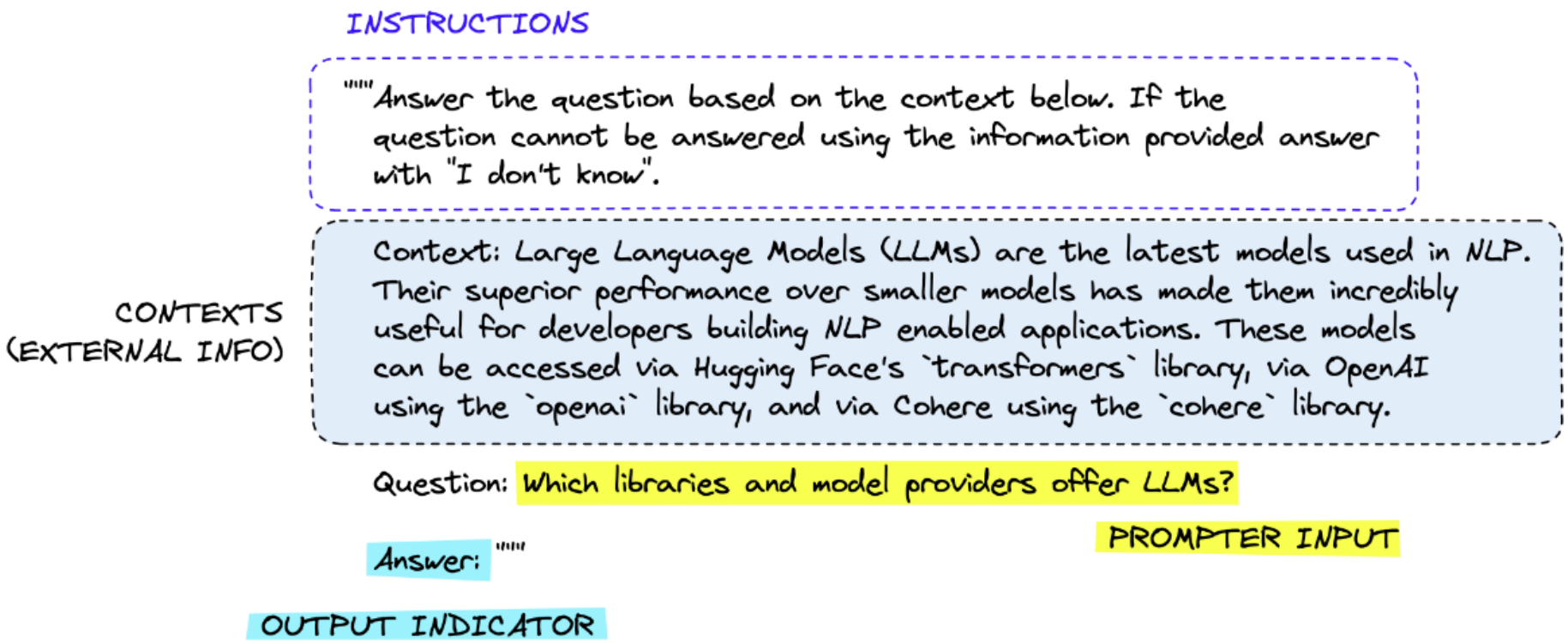
- The success of LLMs comes from their large size and ability to store “knowledge” within the model parameter, which is *learned* during model training.

- However, there are more ways to pass knowledge to an LLM.

- The two primary methods are:
  - **Parametric knowledge** — the knowledge mentioned above is anything that has been learned by the model during training time and is stored within the model weights (or *parameters*).
  - **Source knowledge** — any knowledge provided to the model at inference time via the input prompt.

- Langchain’s **FewShotPromptTemplate** caters to source knowledge input. The idea is to “train” the model on a few examples — we call this few-shot learning — and these examples are given to the model within the prompt

Example Prompt Structure

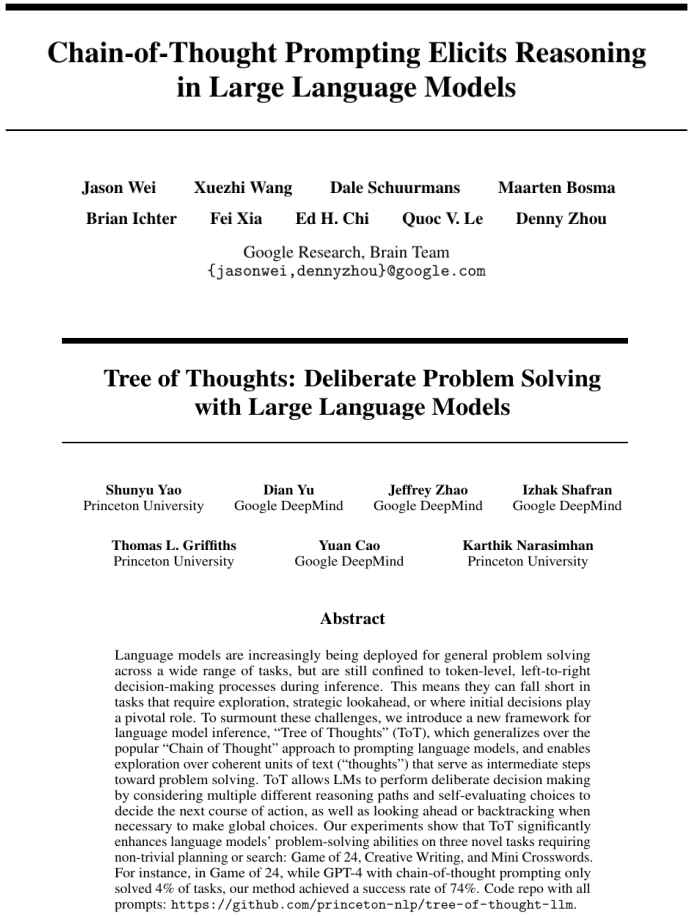


A typical prompt structure.

Ref: https://www.pinecone.io/learn/series/langchain/langchain-prompt-templates/

Advanced Prompt Engineering

- Chain of Thoughts
- Tree of Thoughts
- Graph of Thoughts
- ReAct Agents



Chain-of-Thought Prompting Elicits Reasoning in Large Language Models

Jason Wei Xuezhi Wang Dale Schuurmans Maarten Bosma
Brian Ichter Fei Xia Ed H. Chi Quoc V. Le Denny Zhou

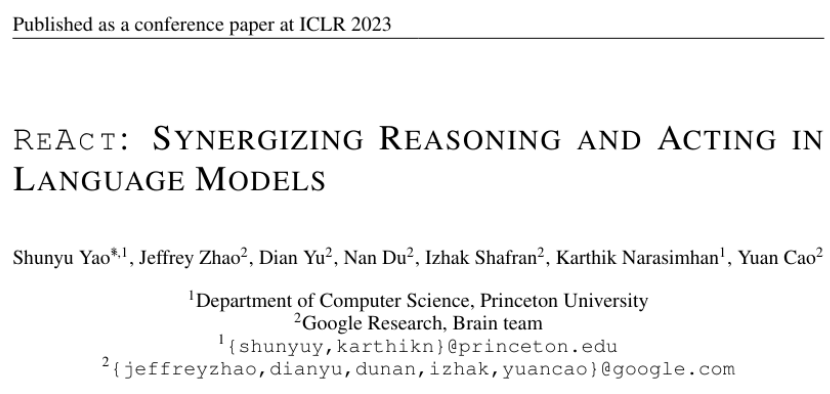
Google Research, Brain Team
{jasonwei,dennyzhou}@google.com

Tree of Thoughts: Deliberate Problem Solving with Large Language Models

Shunyu Yao Princeton University
Dian Yu Google DeepMind
Jeffrey Zhao Google DeepMind
Izhak Shafran Google DeepMind
Thomas L. Griffiths Princeton University
Yuan Cao Google DeepMind
Karthik Narasimhan Princeton University

Abstract

Language models are increasingly being deployed for general problem solving across a wide range of tasks, but are still confined to token-level, left-to-right decision-making processes during inference. This means they can fall short in tasks that require exploration, strategic lookahead, or where initial decisions play a pivotal role. To surmount these challenges, we introduce a new framework for language model inference, "Tree of Thoughts" (ToT), which generalizes over the popular "Chain of Thought" approach to prompting language models, and enables exploration over coherent units of text ("thoughts") that serve as intermediate steps toward problem solving. ToT allows LMs to perform deliberate decision making by considering multiple different reasoning paths and self-evaluating choices to decide the next course of action, as well as looking ahead or backtracking when necessary to make global choices. Our experiments show that ToT significantly enhances language models' problem-solving abilities on three novel tasks requiring non-trivial planning or search: Game of 24, Creative Writing, and Mini Crosswords. For instance, in Game of 24, while GPT-4 with chain-of-thought prompting only solved 4% of tasks, our method achieved a success rate of 74%. Code repo with all prompts: https://github.com/princeton-nlp/tree-of-thought-llm.



Published as a conference paper at ICLR 2023

REACT: SYNERGIZING REASONING AND ACTING IN LANGUAGE MODELS

Shunyu Yao[*,1], Jeffrey Zhao[2], Dian Yu[2], Nan Du[2], Izhak Shafran[2], Karthik Narasimhan[1], Yuan Cao[2]

[1]Department of Computer Science, Princeton University
[2]Google Research, Brain team
[1]{shunyuy,karthikn}@princeton.edu
[2]{jeffreyzhao,dianyu,dunan,izhak,yuancao}@google.com

Published as a conference paper at ICLR 2023

SELF-CONSISTENCY IMPROVES CHAIN OF THOUGHT REASONING IN LANGUAGE MODELS

Xuezhi Wang[†‡] **Jason Wei**[†] **Dale Schuurmans**[†] **Quoc Le**[†] **Ed H. Chi**[†]
Sharan Narang[†] **Aakanksha Chowdhery**[†] **Denny Zhou**[†§]
[†]Google Research, Brain Team
[‡]xuezhiw@google.com, [§]dennyzhou@google.com

Chain-of-Thought (CoT) Prompting

Standard Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The answer is 27. ✗

Chain-of-Thought Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5 + 6 = 11. The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

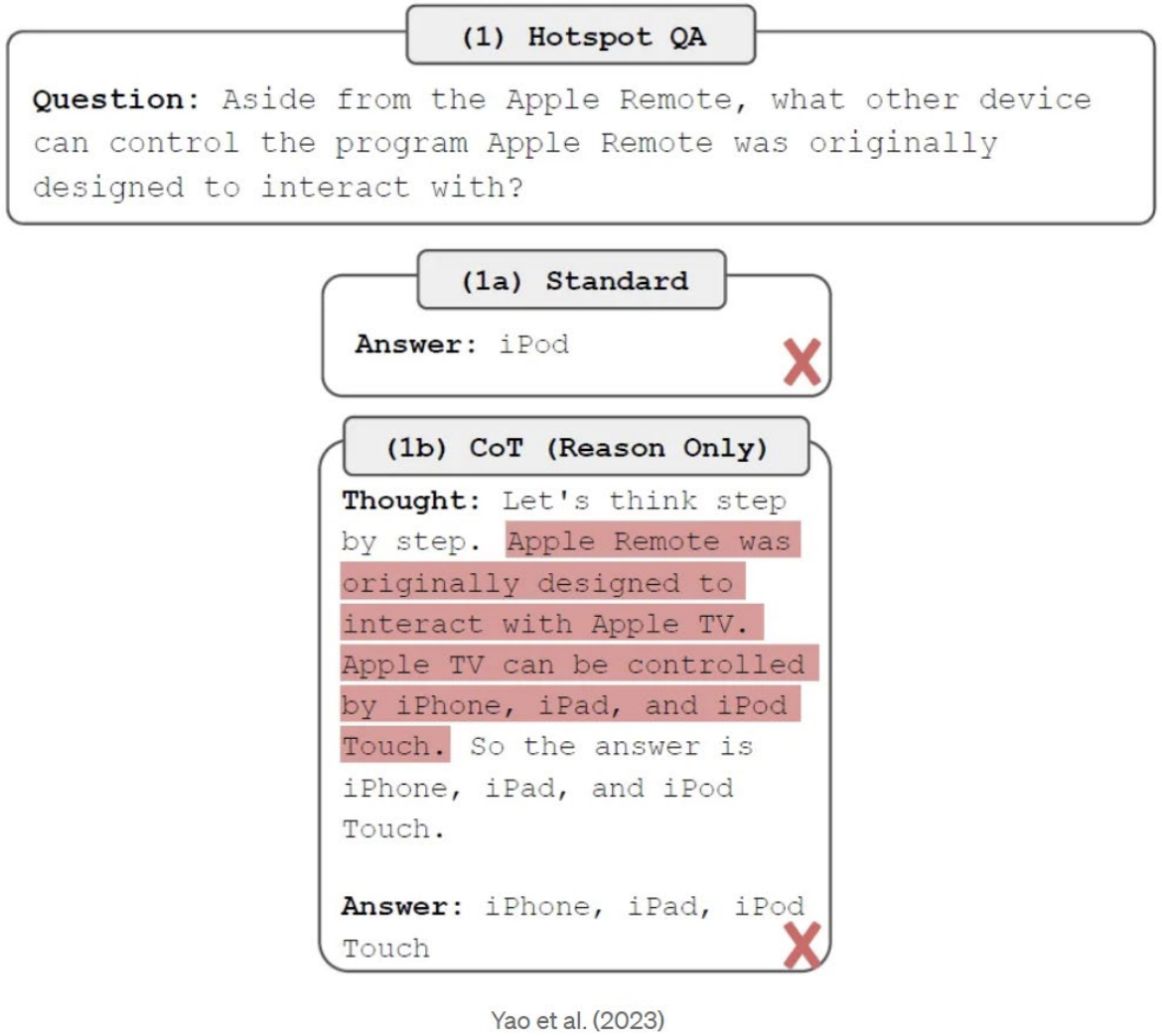
Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had 23 - 20 = 3. They bought 6 more apples, so they have 3 + 6 = 9. The answer is 9. ✓

Chain of Thoughts – see cot.py in my code

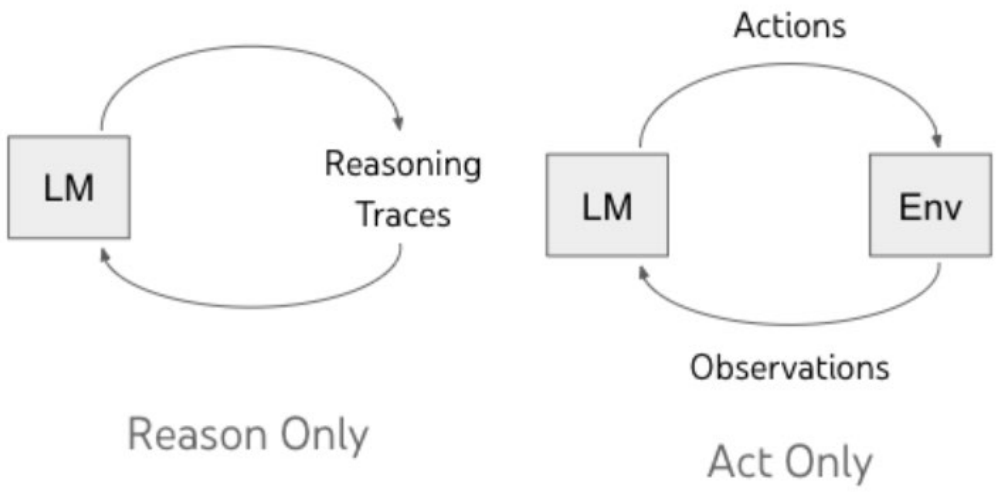
Limitations of CoT Reasoning

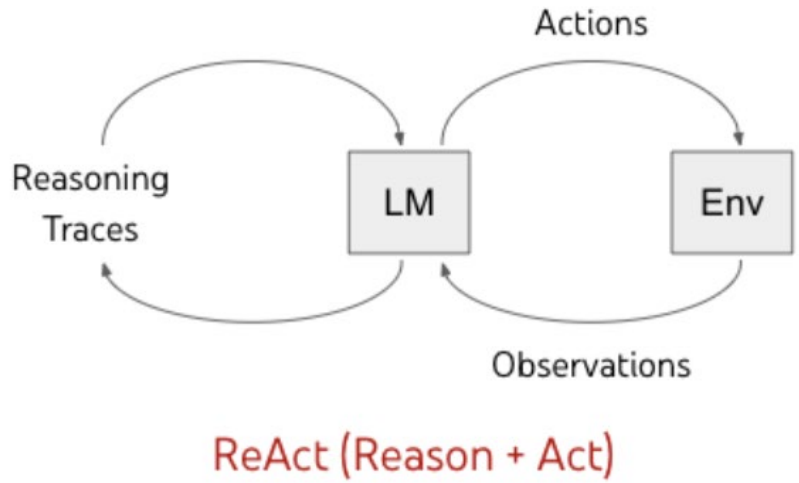
- Chain-of-thought reasoning is intended to combat reasoning errors.
- Giving the LLM one or more examples (few-shot learning) and illustrating how to reason through examples, helps to solve a different problem in a more accurate way.
- But it still suffers from hallucination, and hallucinated “facts” can propagate through the reasoning, causing the model to come to the wrong conclusion regardless.
- ReAct aims to solve this issue by allowing the LLM to take actions such as searching Wikipedia so that it can find facts and reason from those.



Yao et al. (2023)

ReAct Approach: Key Idea





Approach

1. An environment that takes a text action (out of a set of potential actions which can change based on the environment's internal state) and returns a text observation.

2. An output parser framework that stops the agent from generating text once it has written a valid action, executes that action in the environment, and returns the observation (appends it to the text generated so far and prompts the LLM with that).

3. Human-generated examples of intermixed thoughts, actions, and observations in the environment to use for few-shot learning.

Hackathon - Instructions

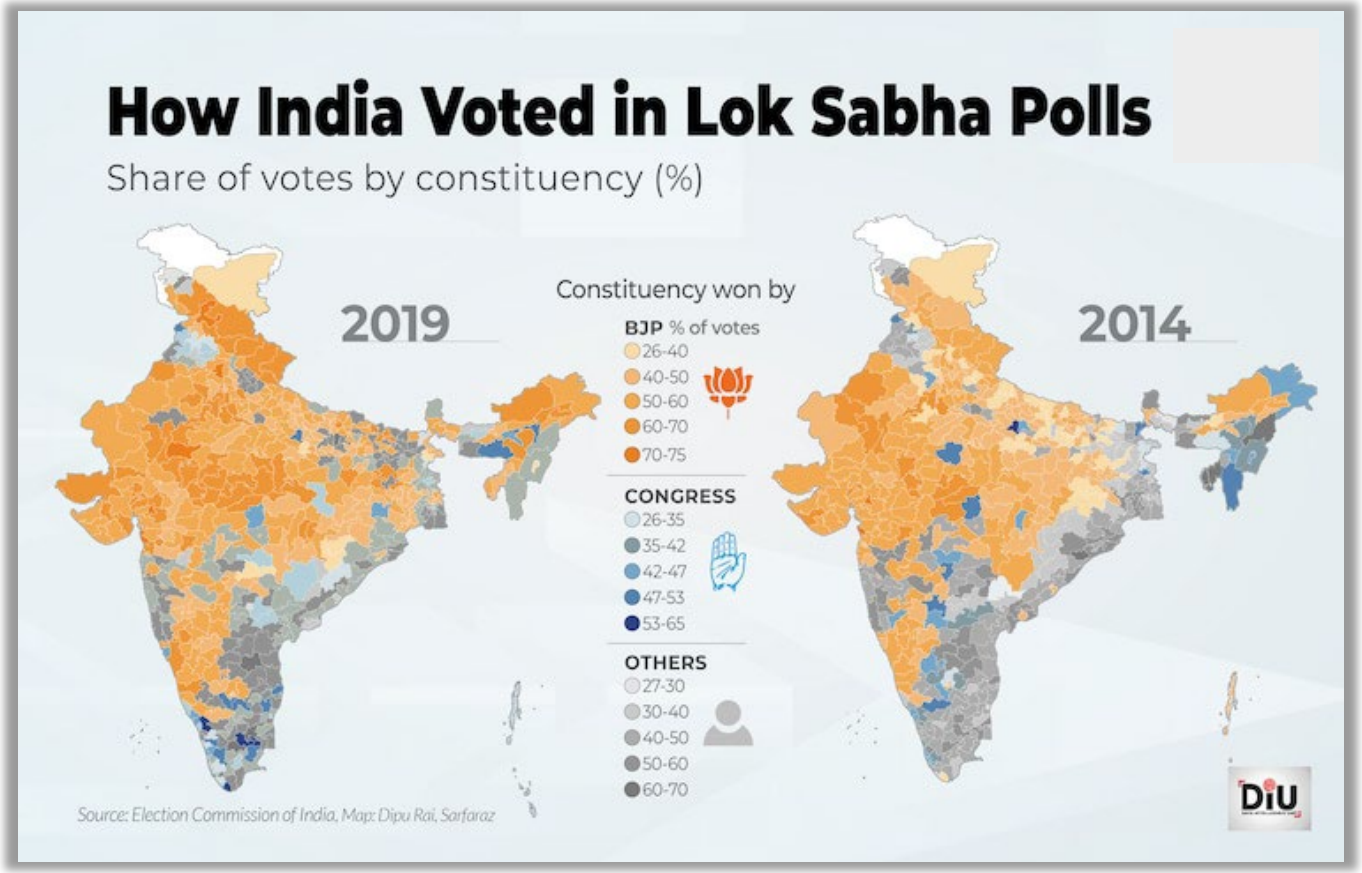
- We will build this project step by step
- For each sub system, we will walk through a sample code and let you code
- You can do this in a team of 4.
- Please complete the dataset cleaning, database creation, basic LLM prompting to generate SQL today. Please create a demo video and place it in the shared folder.
- You can submit the remaining by next session.

Project: Maha Neta

Build a GenAI based product to analyze data from past general elections and Maharashtra assembly elections to help political strategists for the upcoming Maharashtra state elections.

We use the data from 2019, 2024 general elections and 2019 Maharashtra results for this project.

Data source: ECI



Problem - Example

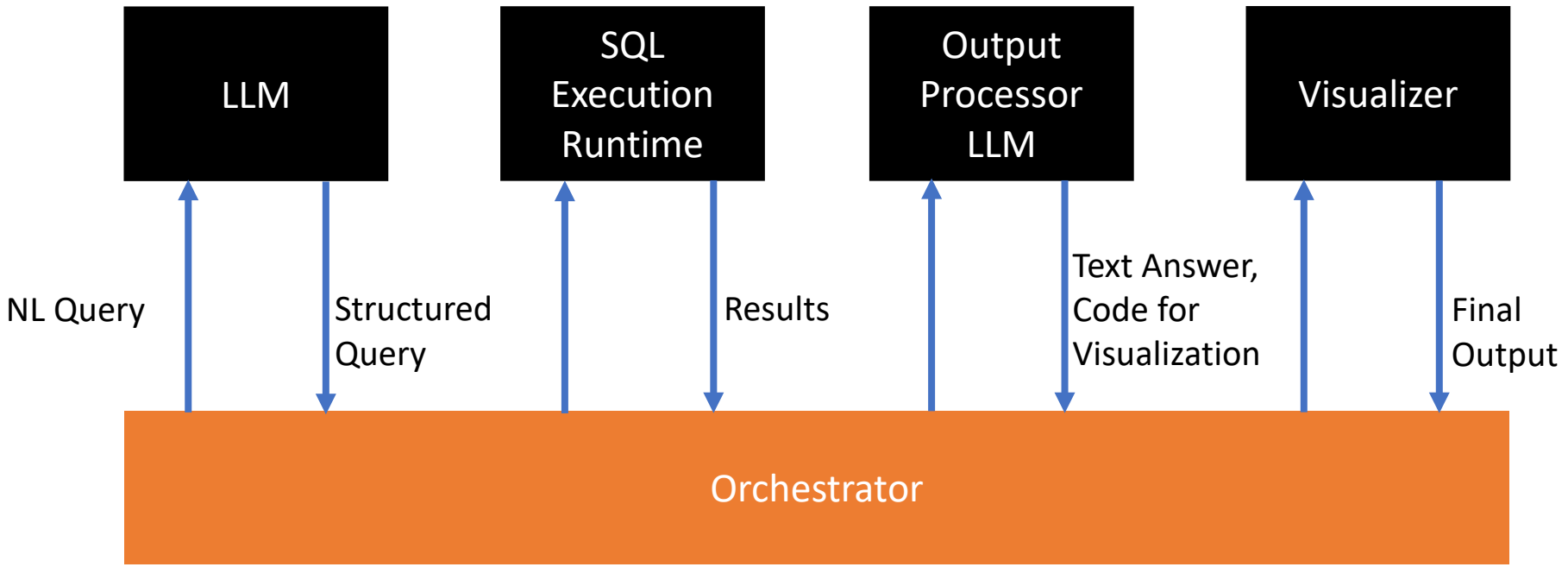
- Persona: You are the key strategist of party X, where X could be the ruling party. You want to list all the constituencies you won in 2019, order them in terms of votes you got (descending order), put them in to 3 categories: Large wins, medium wins and low margin wins.

- You want to analyze those constituencies that are low margin wins more in depth by looking at assembly level granules, gender ratio and/or any other variable and identify them.

- Analyze all lost constituencies, determine if any of them are winnable this time. List them.

- Use visualizations (such as bar graphs, etc) to perform your analysis

General Architecture (Non Agentic, Chain)



Can we use ReAct prompting here?

Steps

- Data downloading (Uploaded in shared drive)
- Data Preparation
- Write the framework code
- Develop Prompts
- Evaluate

Data Preparation

- The files are in .xls format, open them in MS Excel, save them as .xlsx. This will enable us to avoid installing xlrd package and we can directly use pandas to read this file.

- For the excel no 34, remove unwanted rows, ensure that the excel is a plain table.

- Rename the column names as: state_name, constituency_number, constituency_name, assembly_constituency_number, assembly_ constituency_name, total_voters, total_votes_in_state, nota_votes, candidate_name, party_name, secured_votes
  - The purpose of renaming are 2 fold: (a) It is easy to write SQL with this schema (b) LLM can interpret these names easier and can produce the SQL

- You will find that some fields are empty in the votes_secured. Fill them with 0.

- Save this file as .csv

Write Code: DB creation

- Write a function to save the csv data as a sqlite3 db. Name the db as “elections” and table name as “elections_2019”

Write Code: LLM Client

- Run the LLM server using LMStudio as discussed during the earlier hands on
- Develop the client code get_completion(prompt) that takes a prompt as input and returns the output returned by the server.
- Test the code by sending some test prompts and checking the results.

Build a Chatbot using streamlit

- Streamlit is a library to create UI on browser using Python.
- Using streamlit components, it is easy to implement a chatbot in a few lines of code
- Integrate the LLM with streamlit front end using get_completion() function.
- Streamlit has necessary functions for charting and visualization so that one can build LLM driven dashboards quickly
- Review the front end code: my_chatbot.py, you can add necessary code for rich visualization like bar charts etc.

Ref: https://github.com/streamlit/llm-examples/blob/main/Chatbot.py
Ref: https://docs.streamlit.io/develop/tutorials/llms/build-conversational-apps

Write Code: Build DB execution runtime

- Write a module that takes the query as input, execute the query on the given database. You can chose SQL or MongoDB.

- Make sure that the LLM generated code doesn’t cause any harmful side effects, such as deleting or corrupting any database record

Write Code: Prompting for SQL

- Review the questions and pick those that can be answered from the database
- Write prompts that take NL Query and Return SQL from the LLM
- Input should be through Chat GUI and SQL should be displayed in the GUI

OPTIONAL: Use ReAct framework

- ReAct is about using external tool to perform actions

- Can you build a tool that can automatically execute SQL code, get the results, run it again through LLM?

Write Code: Develop the orchestrator

- Now that all modules are coded and tested separately, build the orchestrator that runs the workflow through all these modules.

Generate NL questions

- Using an LLM, auto generate about 25 questions
- These will be used as test cases.
- These questions should be turned in to suitable prompts using the prompt templates

Integrate and Test

- Complete the end to end workflow: starting from questions, generating prompts to the LLM, getting SQL code, running it, getting results from database, post processing and visualizing the results

- You are required to develop and modify your prompts such that you get accurate SQL code out of the LLM

- Evaluate against 25 test cases and report the results.

- Upload your work in the shared folder