Introduction to Prompt Engineering

- Motivation
- Prompt Engineering Project
- GPT-3 Playground
- Examples
- Strategies

Introduction to Prompt Engineering

What is Prompt Engineering about?

- Using existing pre-trained language models to solve a broad range of tasks.
  Might be classification tasks or generative tasks
  (where the task is to generate an output sequence).

- Justification: Language models generalize well and can be applied to tasks which they were not trained on.
  Just need to ask the right way!

- Might allow for tasks that were not possible before

- Benefits: No (re-)training, no finetuning

- Behaviour of the model is almost completely controlled by the model input (i.e. the prompt).
  Must optimize the input prompt to receive optimal results (prompt engineering).

Introduction to Prompt Engineering

Disclaimer

- May be seen as orthogonal to the approaches that you are used to

  Doesn't replace the classic DL approaches that we studied before

- Many aspects stay the same
  - Careful problem analysis
  - Data aquisition and exploration
  - Rigorous and metrics-based evaluation

- May seem like alchemy

  But didn't Feature Engineering and DL Model Engineering feel like alchemy at first, too?

Introduction to Prompt Engineering

Disclaimer

- New problems arise
  - Extremely hard to detect train-test leakage

    Using canary strings in benchmark tasks
  - Non-rigorous evaluation leads to wrong beliefs about performance
  - Model availability and computation cost
  - Blackbox models (even more extreme than in classic DL!)
  - Undetected bias

Introduction to Prompt Engineering

Relevance to term projects

- Provides a way to quickly demonstrate language-based tasks
  Without actually designing and training a model

- Forces you to think about model inputs and outputs

- Helps to visualize the text data in the single processing steps
  Actually see what is happening!

- Might actually be used in the term projects for some pipeline steps
  E.g. for refining the data

- Next week, we will talk about making prompt engineering accessible to software through APIs

Introduction to Prompt Engineering

Large Language Models (LLM)

- Transformer-based language models
- Typically billions of parameters
- Built to generate an output sequence based on an input sequence
- Many publicly available models
  e.g. Open Pre-trained Transformers (OPT)
- Some only through APIs
  e.g. Generative Pre-trained Transformer 3 (GPT-3)

Prompt Engineering Project

Workflow

One mini project for each student.

1. Find an interesting prompt task

   i.e. a question that should be solved using prompt engineering

   Examples follow. (Please do not choose exactly the tasks from the lab.)

2. To make sure that you do engineering and not just running some arbitrary prompt:

   ❑ Choose an interesting aspect to investigate further

      e.g. How well do different prompt engineering strategies work?

   ❑ If the quality of the results can be measured objectively, a plot would be useful

Prompt Engineering Project

Deliverable: Slides

To present the results to your fellow students, please prepare exactly 3 slides:

1. Short title of the project + short problem description in max 2 sentences
2. Example prompt that aims to solve the problem + output
   also specify the used model
3. Plot or chart or … to document your engineering investigation

Please send the slides as a PDF to niklas.deckers@uni-leipzig.de until 12.06.2022, 22:00,

and prepare for a 2-minute presentation in class.

Prompt Engineering Examples

GPT-3 Playground

`https://beta.openai.com/playground/`

- Basic usage
- Engine selection
- Stop sequences

Prompt Engineering Examples

Basic examples

- Q&A
- Magic Spells
- Summarize for a 2nd grader
- Analogy extraction
- Group Name Generation

Prompt Engineering Examples

More examples

- https://beta.openai.com/examples
- https://github.com/google/BIG-bench/tree/main/bigbench/benchmark_tasks#readme

Prompt Engineering Examples

More ideas related to web data

- Argument generation
- Generating replies to reddit posts from a specific subreddit
  r/explainlikeimfive, r/AmItheAsshole, r/changemyview, r/WrongAnswersOnly
  . . .
- Generating/understanding/explaining humour
- Your own ideas?

Prompt Engineering Strategies

General advice

- Do not think that prompts should “command the machine” (imperatives)
- Do not think that “the machine is talking to you”
- Rather think about the output as the “most probable completion”
  Depends on the data the LLM had been trained on, beware of internet language!
- This might also lead to mere reproduction of training data
- Classification tasks are also possible
  Computing which of the given options is more probable to be the completion

Prompt Engineering Strategies

Defining context/identity/intent

- Defining context allows the LLM to adjust to a specific task
- Specifying the intent allows to e.g. prevent insulting language
- May significantly improve result quality

Prompt Engineering Strategies
Few-shot learning

- Idea: The LLM has been trained on a too broad scope, so provide it with successful examples
- Also useful to enforce a certain output format
- Cover a broad range of samples for your task
- Samples must have high quality
- Remember: Computation cost increases linearly

Prompt Engineering Strategies

Instructions

- Originally, LLMs did not perform well with instructions

  This aligns with our expectations from the concept of the “most probable completion”

- However, modern LLMs are often optimized to follow specified instructions

- Might help to specify/clarify the task explicitly

Prompt Engineering Strategies

Intermediate steps

- Splitting the task up into multiple intermediate steps
- May be represented by multiple execution steps
- Should also be reflected in the few-shot learning samples

Prompt Engineering Strategies

Allowing and expecting edge cases

- Allow for special values as a result (e.g. exception statements)

  Can be specified in the context/instruction and included in few-shot learning samples

- For classification-like tasks, cover a broad scope of classes (e.g. neutral)

- Expect outputs that don’t match your desired pattern