# oLLaMa - advance prompting

## Get ready for extreme prompting!

You can utilize either **oLLaMa** or the **OpenAI Web Interface** to conduct the lab work with **chatGPT or GPT4o**. However, please keep in mind that using these proprietary models all your input data may be utilized to retrain the models, so exercise caution when using sensitive information.

This lab aims to further explore the most advanced prompting techniques. As we previously mentioned, each model has its own preferences for prompting. First, we will review these preferences, and then we will explore techniques like **Chain of Thought** (CoT), **self-consistency**, and **Tree of Thought** (ToT) prompts.

Of course, you'll need to **set up a good system prompt** beforehand 😁
To ensure the model operates independently of previous prompts, **use '/clear' at the end of each command prompt**.

Prompt engineering is a technique used in natural language processing (NLP) to improve the performance of the language model by providing them with more context and information about the task in hand. It involves creating prompts, which are short pieces of text that provide additional information or guidance to the model, such as the topic or genre of the text it will generate. By using prompts, the model can better understand what kind of output is expected and produce more accurate and relevant results.

**Be careful! a prompt that works for one model may not be effective for another**.
I encourage you to test more than one model, both small (gemma2:2b) and large (GPT5o), to observe the different impact of model size.

## Prompt engineering[1]

Crafting effective prompts is an important part of prompt engineering. Here are some tips for creating prompts that will help improve the performance of your language model:

- **Be clear and concise**: Your prompt should be easy to understand and provide enough information for the model to generate relevant output. Avoid using jargon or technical terms that may confuse the model.
- **Use specific examples**: Providing specific examples in your prompt can help the model better understand what kind of output is expected. For example, if you want the model to generate a story about a particular topic, include a few sentences about the setting, characters, and plot.

[1] llama prompting
openAI prompting

- **Vary the prompts**: Using different prompts can help the model learn more about the task at hand and produce more diverse and creative output. Try using different styles, tones, and formats to see how the model responds.
- **Test and refine**: Once you have created a set of prompts, test them out on the model to see how it performs. If the results are not as expected, try refining the prompts by adding more detail or adjusting the tone and style.
- **Use feedback**: Finally, use feedback from users or other sources to continually improve your prompts. This can help you identify areas where the model needs more guidance and make adjustments accordingly.

For instance:

| Worse | Better |
|---|---|
| How do I add numbers in Excel? | How do I add up a row of dollar amounts in Excel? I want to do this automatically for a whole sheet of rows with all the totals ending up on the right in a column called "Total". |
| Who's president? | Who was the president of Mexico in 2021, and how frequently are elections held? |
| Write code to calculate the Fibonacci sequence. | Write a TypeScript function to efficiently calculate the Fibonacci sequence. Comment the code liberally to explain what each piece does and why it's written that way. |
| Summarize the meeting notes. | Summarize the meeting notes in a single paragraph. Then write a markdown list of the speakers and each of their key points. Finally, list the next steps or action items suggested by the speakers, if any. |

**Delimiters** like triple quotation marks, XML tags, section titles, etc. can help demarcate sections of text to be treated differently.

```
$ >>> Summarize the text delimited by triple # with a haiku.
###Once you have created a set of prompts, test them out on the model to see how it performs. If the results are not as expected, try refining the prompts by adding more detail or adjusting the tone and style###
Test the prompts you write,
Refine with tone, style, and depth,
Till results feel right. //by GPT4

Prompt crafting's art,
Tests reveal where shadows hide,
Refinement brings spark  //by gemma2:2b 🥰
```

```
$ >>> You will be provided with a pair of articles (delimited with XML tags) about the same topic. First summarize the arguments of each article. Then indicate which of them makes a better argument and explain why.

<article> A large language model (LLM) is a type of artificial intelligence (AI) program that
```

```
can recognize and generate text, among other tasks. LLMs are trained on small sets of
data — hence the name "large." LLMs are built on machine learning: specifically, a type of
neural network called a LSTM model. </article>

<article> A large language model (LLM) is a computational model capable of language
generation or other natural language processing tasks. As language models, LLMs acquire
these abilities by learning statistical relationships from vast amounts of text during a
self-supervised and semi-supervised training process </article>
```

```
$ >>> You will be provided with a thesis abstract and a suggested title for it. The thesis title
should give the reader a good idea of the topic of the thesis but should also be
eye-catching. If the title does not meet these criteria, suggest 5 alternatives.

Abstract: We explore how generating a chain of thought -- a series of intermediate
reasoning steps -- significantly improves the ability of large language models to perform
complex reasoning. In particular, we show how such reasoning abilities emerge naturally in
sufficiently large language models via a simple method called chain of thought prompting,
where a few chain of thought demonstrations are provided as exemplars in prompting.
Experiments on three large language models show that chain of thought prompting
improves performance on a range of arithmetic, commonsense, and symbolic reasoning
tasks. The empirical gains can be striking. For instance, prompting a 540B-parameter
language model with just eight chain of thought exemplars achieves state of the art
accuracy on the GSM8K benchmark of math word problems, surpassing even finetuned
GPT-3 with a verifier.

Title: Attention is all you need
```

**Orig title:** Chain-of-Thought Prompting Elicits Reasoning in Large Language Models

Some tasks are best specified as **a sequence of steps**. Writing the steps out explicitly can make it easier for the model to follow them.

```
$ >>> Use the following step-by-step instructions to respond to user inputs.

Step 1 - The user will provide you with text in triple quotes. Summarize this text in one
sentence with a prefix that says "Summary: ".

Step 2 - Translate the summary from Step 1 into Spanish, with a prefix that says
"Translation: ".

"""Word Embeddings in NLP is a technique where individual words are represented as
real-valued vectors in a lower-dimensional space and captures inter-word semantics. Each
word is represented by a real-valued vector with tens or hundreds of dimensions."""
```

You can ask the model to produce **outputs that are of a given target length**. The targeted output length can be specified in terms of the count of words, sentences, paragraphs, bullet points, etc. Note however that instructing the model to generate a specific number of words does not work with high precision. The model can more reliably generate outputs with a specific number of paragraphs or bullet points.

```
$ >>> Summarize the text delimited by triple hashtag in 3 bullet points.

###Long short-term memory is a type of recurrent neural network (RNN) aimed at dealing
```

```
with the vanishing gradient problem present in traditional RNNs. Its relative insensitivity to
gap length is its advantage over other RNNs, hidden Markov models and other sequence
learning methods. It aims to provide a short-term memory for RNN that can last thousands
of timesteps, thus "long short-term memory". The name is made in analogy with long-term
memory and short-term memory and their relationship, studied by cognitive psychologists
since early 20th century.###
```

More prompt examples: https://platform.openai.com/docs/examples

## General approaches

All models can benefit from certain prompting techniques. Let's go through them one by one. Don't forget about few-shot prompting; it generally improves results. Let's revisit some basic concepts.

Large language models like Meta Llama are capable of following instructions and producing responses without having previously seen an example of a task. Prompting without examples is called **"zero-shot prompting"**.

```
$ >>> Text: This was the best movie I've ever seen!
The sentiment of the text is:
```

Adding specific examples of your desired output generally results in a more accurate, consistent output. This technique is called **"few-shot prompting"**. In this example, the generated response follows our desired format that offers a more nuanced sentiment classifier that gives a positive, neutral, and negative response confidence percentage.

```
$ >>> You are a sentiment classifier. For each message, give the percentage of
positive/netural/negative. Here are some samples:
Text: I liked it
Sentiment: 70% positive 30% neutral 0% negative

Text: It could be better
Sentiment: 0% positive 50% neutral 50% negative

Text: It's fine
Sentiment: 25% positive 50% neutral 25% negative

Text: I thought it was okay

Text: I loved it!

Text: Terrible service 0/10:
```

The following is an example of a multilingual few shot prompt used for LLaMA3.1 by a HiTZ member (Iker Garcia an excellent researcher) in a shared task, which helped them secure first place in a sexism detection competition. Check it out! **The correct answers is non-sexist:**

```
$ >>> Analyze the given text to determine if it contains sexist content.

Definition of sexism: Sexism is prejudice, stereotyping, or discrimination, typically against
women, on the basis of sex. It includes attitudes, behaviors, and practices that promote ste
reotyping of social roles based on gender.

- 'sexist': The text expresses or implies sexist attitudes, stereotypes, or discriminatory
views.
- 'non-sexist': The text does not contain any sexist language, attitudes, or implications.

Output: Provide your answer as a JSON object with the key 'label' and the value set to
either 'sexist' or 'non-sexist'.

Examples
--------

Input: Les dejamos esta #lecturarecomendada: "Hacia una criminología feminista" de
Carmen Antony. Lo pueden encontrar en nuestra web.https://t.co/rSy8qoToCZRealizamos
envíos a todo el país a través de Correo Argentino. A CABA también llegamos por
mensajería. https://t.co/azqHCcePXL
Output: {"label":"non-sexist"}

[+19 examples]

--------

Now, analyze the following input:

Input: Uugghhh, porque tengo que ser bisexual! !! Estoy cansada de los hombres!! Alguna
niña linda que quiera ser mi novia? Pd: niña linda = todas las mujeres
```

**Chain of Thought** Technique Involves providing the language model with a series of prompts or questions to help guide its thinking and generate a more coherent and relevant response. This technique can be useful for generating more thoughtful and well-reasoned responses from language models.

Pros:

1. Improves coherence: Helps the language model think through a problem or question in a logical and structured way, which can lead to more coherent and relevant responses.
2. Increases depth: Providing a series of prompts or questions can help the language model explore a topic more deeply and thoroughly, potentially leading to more insightful and informative responses.

Cons:

1. Requires effort: The chain of thought technique requires more effort to create and provide the necessary prompts or questions.
2. Chain-of-Thought prompting has been proven to be less effective on smaller models.

CoT example, the **correct answer is option 2**:
Zero shot w/o CoT[2]:

```
$ >>> Which is a faster way to get to work?
Option 1: Take a 1000 minute bus, then 30 minute train, and finally a 50 minute bike ride.
Option 2: Take a 1000 minute bus, then 40 minute train, and finally a 30 minute bike ride.
```

1shot **CoT**:

```
$ >>> Which is a faster way to get home?
Option 1: Take an 10 minutes bus, then an 40 minute bus, and finally a 10 minute train.
Option 2: Take a 90 minutes train, then a 45 minute bike ride, and finally a 10 minute bus.
Option 1 will take 10+40+10 = 60 minutes.
Option 2 will take 90+45+10=145 minutes.
Since Option 1 takes 60 minutes and Option 2 takes 145 minutes, Option 1 is faster.

Which is a faster way to get to work?
Option 1: Take a 1000 minute bus, then 30 minute train, and finally a 50 minute bike ride.
Option 2: Take a 1000 minute bus, then 40 minute train, and finally a 30 minute bike ride.
```

CoT example form [CoT Prompting Elicits Reasoning in Large Language Models] **correct answer is 9**:

1shot w/o CoT:

```
$ >>> 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?

A: The answer is 11.

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

A:
```

1shot **CoT**:

```
$ >>> 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?

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 has 23 apples. If they used 20 to make lunch and bought 6 more, how
many apples do they have?

A:
```

With Ollama, running llama2:7B may initially fail, but performance will improve with Chain of Thought (CoT) prompting.

[2] Some of the newest nonThinker models also perform a Chain-of-Thought (CoT) process by default before answering. Try to avoid this behavior by adjusting the prompt and adding instructions that make the model respond directly, without any explanation.

You will find more CoT examples in the paper.

LLMs are probabilistic, so even with Chain-of-Thought, a single generation might produce incorrect results. **Self-Consistency** introduces enhanced accuracy by selecting the most frequent answer from multiple generations (at the cost of higher compute):
Running the above CoT examples several times and taking the most commonly returned value for the answer would make use of the self-consistency approach. Try it!

**Tree-of-Thought** (ToT) Prompting, a novel technique inspired by the Tree-of-Thoughts framework, expands and refines the established Chain-of-Thought prompting method. This approach enables Large Language Models to demonstrate enhanced “reasoning” abilities. Through Tree-of-Thought Prompting, these models can independently correct their errors while progressively building knowledge. Let’s introduce the following problem, **the correct answer is “The ball is in the bedroom**”:

```
$ >>>  Bob is in the living room.
He walks to the kitchen, carrying a cup.
He puts a ball in the cup and carries the cup to the bedroom.
He turns the cup upside down, then walks to the garden.
He puts the cup down in the garden, then walks to the garage.
Where is the ball?
```

We can formulate the problem as follows to use ToT:

```
$ >>> Bob is in the living room.
He walks to the kitchen, carrying a cup.
He puts a ball in the cup and carries the cup to the bedroom.
He turns the cup upside down, then walks to the garden.
He puts the cup down in the garden, then walks to the garage.
Where is the ball?

Imagine three different agents are answering this question.
All agents will write down 1 step of their thinking, then share it with the group.
Then all agents will go on to the next step.
If any agent realizes they're wrong at any point then they leave.
```

Very interesting approach! However, take a look at the results and let's keep going. This kind of prompting is advanced material, but of course, we will cover it, specifically on the 4th day with Agentic LLMs.

## Give model time to think

Sometimes we get better results when we explicitly instruct the model to reason from first principles before coming to a conclusion. Suppose for example we want a model to evaluate a student's solution to a math problem. The most obvious way to approach this is to simply ask the model if the student's solution is correct or not.

Sometimes, simply adding "**Let's think step by step**" at the end of the prompt is enough to trigger automatic reasoning in LLMs, encouraging a more logical and structured response. Try the original CoT example from the paper:

```
$ >>> The cafeteria has 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Let's think step by step.
```

## Newest reasoning models

The newest models from 2024, such as openAI o1, include automatic reasoning, meaning they inherently perform techniques like Chain of Thought (CoT) and "Let's think step by step" by default. However, these models have opted to hide the reasoning process, presenting only the final result to the user instead of displaying the step-by-step thinking 😓 ("Open"AI)

How it works: they trained these models to spend more time thinking through problems before they respond, much like a person would. Through training, they learn to refine their thinking process, try different strategies, and recognize their mistakes.

It performs similarly to PhD students on challenging benchmark tasks in physics, chemistry, and biology. It excels in math and coding. In a qualifying exam for the International Mathematics Olympiad (IMO), GPT-4o correctly solved only 13% of problems, while the reasoning model scored 83%. Their coding abilities were evaluated in contests and reached the 89th percentile in Codeforces competitions. You can read more about this in the technical research post.

As an early model, it doesn't yet have many of the features that make ChatGPT useful, like browsing the web for information and uploading files and images. **For many common cases GPT-4o will be more capable in the near term**.

But for complex reasoning tasks this is a significant advancement and represents a new level of AI capability.

Since 2024, reasoning models have evolved significantly, with highly competitive open models such as **deepseek** and **Qwen3** starting to reshape the state of the art in reasoning. These models explicitly display their reasoning process within tags!

Try running **Qwen3**, for example, and experience the power of modern reasoning models. Keep in mind that sometimes they might “think” for a while. You can use **/set nothink** to disable reasoning mode and **/set think** to activate it again:

```
$ ollama run qwen3:4b
```

Check whether the proposed model could correctly answer the previous examples without using CoT.

Once you're done, try the typical river crossing puzzle example 🐺🐐🥬. Is any model able to solve the problem correctly?

> A farmer with a wolf, a goat, and a cabbage must cross a river by boat. The boat can carry only the farmer and a single item. If left unattended together, the wolf would eat the goat, or the goat would eat the cabbage. How can they cross the river without anything being eaten?

**PD**: You can search for prompting cheat sheets, where you'll find lots of tricks and prompt guides. However, it's always better to try a few, evaluate their performance, and choose the best one. Prompts are model-dependent, so keep that in mind!