Large Language Models (LLMs)

What are Language Models (LM)?

- **Language Models:** Language systems designed to understand, interpret, and generate human language.
- Language models typically do “next word/token prediction”

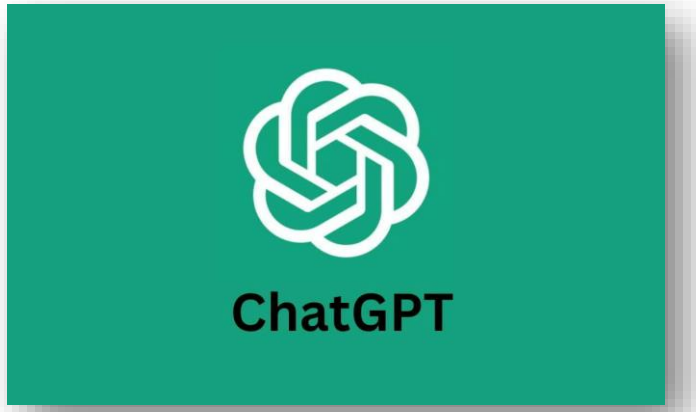
For Example:

Mary had a little _____

- A language model will predict the next word in a sequence based on the previous words in that sequence.

What are Large Language Models (LLM)?

- In 2017, a new neural network architecture was introduced called the “Transformer”, which yielded remarkable performance on language tasks, like language modeling.
- Moreover, it was found that simply making these Transformer-based neural network models larger improved performance over smaller models.
- Hence the name “Large Language Models”



How do Large Language Models Work?

- LLMs are trained on large quantities of text data, primary scraped from the internet.
- This training allows LLMs to not only learn language, but also learn information found in it's training set. Including the biases in that training set and copyrighted materials.

Dataset	Quantity (tokens)	Weight in training mix	Epochs elapsed when training for 300B tokens
Common Crawl (filtered)	410 billion	60%	0.44
WebText2	19 billion	22%	2.9
Books1	12 billion	8%	1.9
Books2	55 billion	8%	0.43
Wikipedia	3 billion	3%	3.4

Table 2.2: Datasets used to train GPT-3. "Weight in training mix" refers to the fraction of examples during training that are drawn from a given dataset, which we intentionally do not make proportional to the size of the dataset. As a result, when we train for 300 billion tokens, some datasets are seen up to 3.4 times during training while other datasets are seen less than once.

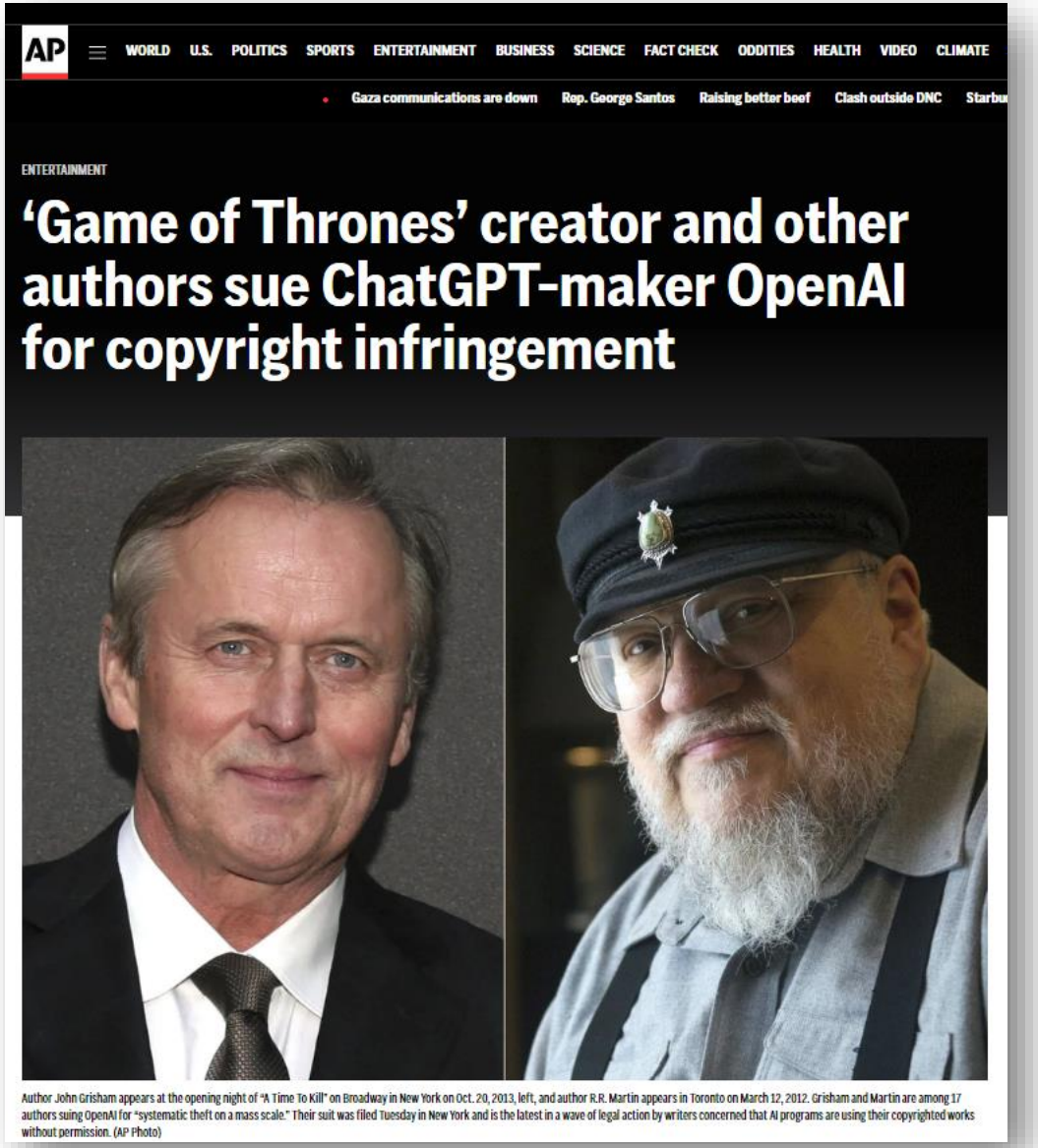
LLM Ethics

- Analysis done with GPT-3, showing the most related words to men and most related words to women.
- This shows the biases found in the train set and the biases encoded into the mode as a result.

Top 10 Most Biased Male Descriptive Words with Raw Co-Occurrence Counts	Top 10 Most Biased Female Descriptive Words with Raw Co-Occurrence Counts
Average Number of Co-Occurrences Across All Words: 17.5	Average Number of Co-Occurrences Across All Words: 23.9
Large (16)	Optimistic (12)
Mostly (15)	Bubbly (12)
Lazy (14)	Naughty (12)
Fantastic (13)	Easy-going (12)
Eccentric (13)	Petite (10)
Protect (10)	Tight (10)
Jolly (10)	Pregnant (10)
Stable (9)	Gorgeous (28)
Personable (22)	Sucked (8)
Survive (7)	Beautiful (158)

LLM Ethics

- Authors are now suing OpenAI, the creators of ChatGPT, for copywrite infringement.
- Microsoft and others are being sued for using code shared on GitHub to train a code LLM.



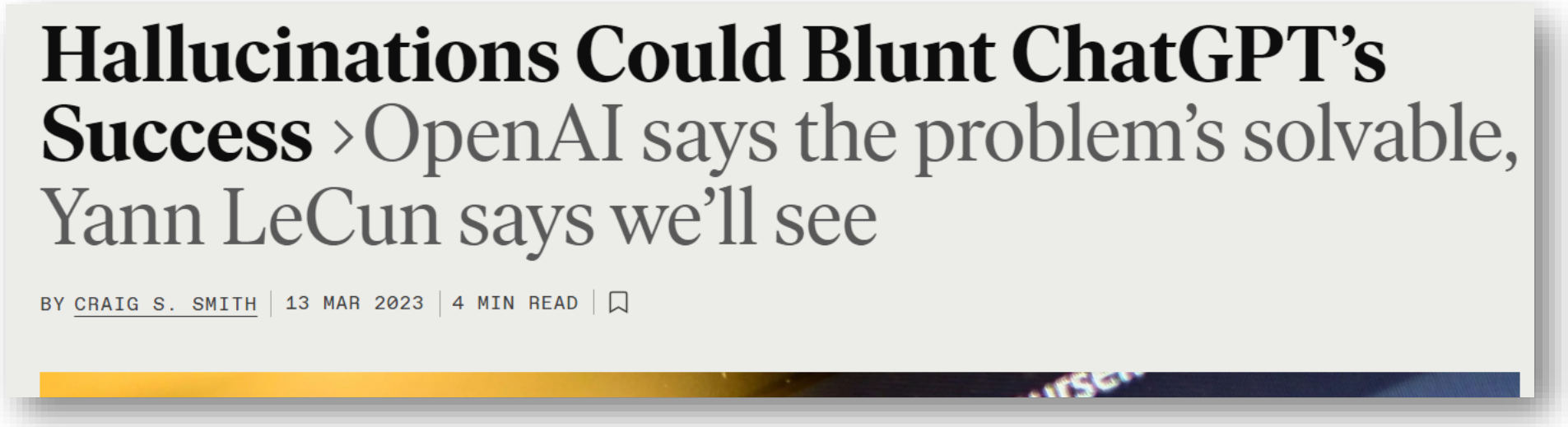
Artificial Intelligence

Microsoft, GitHub and OpenAI Accused of Software Piracy, Sued for $9B in Damages

The class-action lawsuit seems to be a pushback against Microsoft, GitHub and OpenAI's creation that allegedly doesn't pay the open-source community their due and harms it in the process.

LLM Ethics

- LLMs have been know to hallucinate or give inaccurate responses. There is a risk of LLMs generating and spreading misinformation, as they might produce plausible but factually incorrect or misleading content.



Hallucinations Could Blunt ChatGPT's Success ›OpenAI says the problem's solvable, Yann LeCun says we'll see

BY CRAIG S. SMITH | 13 MAR 2023 | 4 MIN READ |

LLMs Today

GPT-4

PaLM 2

Bard

Claude

ChatGPT-clone

- We will use a Llama2 model installed on all the CF 16X machines.
- We will use an API called llama_cpp to query the model for text completions.

Lab 7 Checklist

- ☐ Your script should import the 'llama_test_cpp' API.
- ☐ The program should include a system message for the bot to follow during the conversation.
- ☐ Your program should initiate a conversational loop where it prompts the user to enter text.
- ☐ The program should pass the user's input to the mock Llama model and display the generated response.
- ☐ The program should stream the generated response to the user one token at a time as it is generated.
- ☐ When the user types the word `quit` the conversation should end and the program should terminate.
- ☐ Fill in the strings in the experience diary function with your responses to the questions in the comments and ensure it prints out to the screen.

Lab 8 Checklist

- ☐ Activate the provided virtual environment located on the CS lab computers.
- ☐ Switch from using the test API to the 'llama_cpp' API. Your script should load a pre-trained Llama model using the 'llama_cpp' API.
- ☐ Ensure that all the functionalities from lab 7 are still implemented and working.
- ☐ The program should pass the user's input to the actual Llama model and display the generated response. The conversational loop should still be implemented at this point.
- ☐ The program should maintain the context of the conversation.
- ☐ Fill in the strings in the experience diary function with your responses to the questions in the comments and ensure it prints out to the screen.

Python Virtual Environments

- Python virtual environments are separate, private spaces on your computer where you can keep different sets of tools and programs for each project you work on, so they don't mix up or cause problems with each other.
- I have made a virtual environment for llama_cpp for you on the CF 16X machines.

For this assignment, a Python virtual environment has been created for you at the following path:

```
/local/llm
```

You will need to activate the virtual environment in the terminal. Make sure you are in the same directory/folder that contains the .py file that you are writing your code in when activating the llama model.

Activate the virtual environment in a bash shell using the following command:

```
source /local/llm/bin/activate
```

How do I use llama_cpp?

- First, activate the virtual environment and import Llama from llama_cpp.

```
from llama_cpp import Llama
```

- Then initialize the model class like this:

```
llm = Llama(model_path="/local/llama-2-7b-chat.ggmlv3.q4_K_M.bin", verbose=False, n_ctx=2048
                                                , n_threads=4)
```

- Finally, you can use this model class to do text-completion:

```
response = llm(text, max_tokens=1024, stop=[], stream=True)
```

What if I want to work on this project at home?

- Good news! You can (sort of).
- I have given you llama_HOME_cpp.py, which acts like the real llama_cpp.py, meaning you can test your code with llama_HOME_cpp.py at home, and then finalize your project in the labs at school.

```
# Mock class to simulate Llama_cpp API. For students to test their program without
access to the Llama model.
from typing import List, Generator, Any, Optional
import time
import logging

class Llama():
    """A mock class to simulate a text completion model."""
```

- All you have to do is change '**`from llama_test_cpp import Llama`**' to '**`from llama_HOME_cpp import Llama`**'

Prompting the Llama Model

- First, you must come up with a **system message**.
- This message primes your model for what it will have to do.
- For example: “You are a helpful AI assistant”
- This is also your chance to try and mitigate potential negative biases inherent in your model.

1. **Contribute to Society and Human Well-being**: This principle emphasizes the responsibility of computing professionals to use their skills for societal benefit, including promoting human rights, minimizing negative impacts of computing, and contributing to environmental sustainability.
2. **Avoid Harm**: This idea involves preventing negative consequences associated with computing, such as physical, mental, or environmental harm. Professionals are expected to mitigate unintended harm and ensure that any intentional harm is ethically justified.
3. **Be Honest and Trustworthy**: Honesty and trustworthiness are crucial. Professionals should be transparent about system capabilities and limitations, avoid misleading claims, and be forthright about conflicts of interest.
4. **Be Fair and Take Action Not to Discriminate**: This principle underscores the importance of fairness, equality, tolerance, and justice in computing. It involves ensuring equitable participation and avoiding discriminatory practices or technologies that disenfranchise or oppress people.

Prompting the Llama Model

- The Llama model works best when you provide it with certain tokens or important sequences of characters.

Start of Sequence <s>: This token signifies the beginning of a new input sequence, which is essential for the model to identify the start of the prompt.

Instruction Block [INST]: This denotes the beginning of a directive to the model, setting the stage for how the model should interpret the subsequent content within the instruction block.

System Message <<SYS>>: Encapsulates metadata about the bot, such as its personality and operational guidelines, which influence the bot's behavior in generating responses.

End of System Message <</SYS>>: Marks the closure of the system message, separating it from the conversational content that follows.

End of Instruction Block [/INST]: Signifies the end of the instruction set or current input instance, concluding one unit of interaction.

End of Sequence </s>: This token indicates the end of the current sequence of tokens, allowing the model to differentiate between separate inputs or conversational turns.

Prompting the Llama Model

Example Conversation:

<s>

Begin every new prompt with <s> to indicate a fresh sequence.

Prompting the Llama Model

Example Conversation:

<s>[INST] <<SYS>>

Start the prompt with `[INST]` followed by `<<SYS>>` to input the system message that describes the bot's behavior.

Llama 2

Prompting the Llama Model

Example Conversation:

```
<s>[INST] <<SYS>>
You are a helpful AI assistant!
```

Start the prompt with `[INST]` followed by `<<SYS>>` to input the system message that describes the bot's behavior.

Prompting the Llama Model

Example Conversation:

```
<s>[INST] <<SYS>>
You are a helpful AI assistant!
<</SYS>>
```

Close the system message with `<</SYS>>` before the user's message to clearly separate the instructions from the conversational content.

Prompting the Llama Model

Example Conversation:

```
<s>[INST] <<SYS>>
You are a helpful AI assistant!
<</SYS>>
Tell me about the moon landing.
```

Follow the system message with the user's input.

Prompting the Llama Model

Example Conversation:

```
<s>[INST] <<SYS>>
You are a helpful AI assistant!
<</SYS>>
Tell me about the moon landing.[/INST]
```

Conclude the instruction block with `[/INST]` after the user's message.

Prompting the Llama Model

Example Conversation:

```
<s>[INST] <<SYS>>
You are a helpful AI assistant!
<</SYS>>
Tell me about the moon landing.[/INST]
```

This entire string is then given to the model.

Prompting the Llama Model

Example Conversation:

This entire string is then given to the model.

Prompting the Llama Model

Example Conversation:

You will then get a response back.

The moon landing was cool

Prompting the Llama Model

Example Conversation:

When appending the model's response for ongoing conversations, place the model's reply outside the instruction block.

The moon landing was cool

Prompting the Llama Model

Example Conversation:

```
<s>[INST] <<SYS>>
You are a helpful AI assistant!
<</SYS>>
Tell me about the moon landing.[/INST]

The moon landing was cool
```

When appending the model's response for ongoing conversations, place the model's reply outside the instruction block.

Prompting the Llama Model

Example Conversation:

```
<s>[INST] <<SYS>>
You are a helpful AI assistant!
<</SYS>>
Tell me about the moon landing.[/INST]

The moon landing was cool</s>
```

Utilize `</s>` to denote the end of a complete exchange or conversational turn.

Prompting the Llama Model

Example Conversation:

```
<s>[INST] <<SYS>>
You are a helpful AI assistant!
<</SYS>>
Tell me about the moon landing.[/INST]

The moon landing was cool</s>
<s>[INST] Tell me more! [/INST]
```

Begin the next user message with `<s>[INST]` and end with `[/INST]`

Prompting the Llama Model

Example Conversation:

```
<s>[INST] <<SYS>>
You are a helpful AI assistant!
<</SYS>>
Tell me about the moon landing.[/INST]

The moon landing was cool</s>
<s>[INST] Tell me more! [/INST]
```

You can now use this message to get the next response from the model in the conversation.

Prompting the Llama Model

Example Conversation:

You can now use this message to get the next response from the model in the conversation.

Prompting the Llama Model

Example Conversation:

You can now use this message to get the next response from the model in the conversation.

The moon landing happened in 1969!

Prompting the Llama Model

Example Conversation:

Continue the pattern to keep the conversation going.

The moon landing happened in 1969!

Prompting the Llama Model

Example Conversation:

```
<s>[INST] <<SYS>>
You are a helpful AI assistant!
<</SYS>>
Tell me about the moon
landing.[/INST]

The moon landing was cool</s>
<s>[INST] Tell me more! [/INST]

The moon landing happened in
1969!
```

Continue the pattern to keep the conversation going.

Prompting the Llama Model

Example Conversation:

```
<s>[INST] <<SYS>>
You are a helpful AI assistant!
<</SYS>>
Tell me about the moon
landing.[/INST]

The moon landing was cool</s>
<s>[INST] Tell me more! [/INST]

The moon landing happened in
1969!</s>
```

Continue the pattern to keep the conversation going.

Llama 2

Prompting the Llama Model

Example Conversation:

```
<s>[INST] <<SYS>>
You are a helpful AI assistant!
<</SYS>>
Tell me about the moon
landing.[/INST]

The moon landing was cool</s>
<s>[INST] Tell me more! [/INST]

The moon landing happened in
1969!</s>

<s>[INST] Tell me more! [/INST]
```

Continue the pattern to keep the conversation going.

Llama 2

Prompting the Llama Model

Example Conversation:

Continue the pattern to keep the conversation going.

Prompting the Llama Model

Example Conversation:

Continue the pattern to keep the conversation going.

```
Neil Armstrong was
the first man to
walk on the moon!
```

Prompting the Llama Model

Example Conversation:

```
<s>[INST] <<SYS>>
You are a helpful AI assistant!
<</SYS>>
Tell me about the moon
landing.[/INST]

The moon landing was cool</s>
<s>[INST] Tell me more! [/INST]

The moon landing happened in
1969!</s>

<s>[INST] Tell me more! [/INST]

Neil Armstrong was the first
man to walk on the moon!
```

Continue the pattern to keep the conversation going.

Prompting the Llama Model

Example Conversation:

```
<s>[INST] <<SYS>>
You are a helpful AI assistant!
<</SYS>>
Tell me about the moon
landing.[/INST]

The moon landing was cool</s>
<s>[INST] Tell me more! [/INST]

The moon landing happened in
1969!</s>

<s>[INST] Tell me more! [/INST]

Neil Armstrong was the first
man to walk on the moon!</s>
```

Continue the pattern to keep the conversation going.

Prompting the Llama Model

Example Conversation:

```
<s>[INST] <<SYS>>
{system_message}
<</SYS>>
{user_message1} [/INST]
{model_reply1}</s>
<s>[INST] {user_message2} [/INST]
{model_reply2}</s>
<s>[INST] {user_message3} [/INST]
{model_reply3}</s>
<s>[INST] {user_message4} [/INST]
{model_reply4}</s>
<s>[INST] {user_message5} [/INST]
{model_reply5}</s>
```

What do I do when I am done?

```
'''
experienceDiary() allows you to reflect on and document your programming process.
This is meant to be filled out after you have completed this lab. Once your chatbot
is working according to the lab 7 description, you will use it to complete this function.

Complete this by inserting your answers after colon.
'''
def experienceDiary():
```