Introduction to LLMs Module

Introduction to LLMs

This module uncovers the core principles of Large Language Models (LLMs), zooming in on their foundational underpinnings. We provide a historical perspective, highlighting the emergence of Transformers and the difference between proprietary and open-source LLMs. Key attention is on recognizing and mitigating inherent issues like hallucinations and biases within these models.
The module is structured as follows, concisely describing each lesson.

- **What are Large Language Models?** This lesson dives into the core principles of LLMs, highlighting the capabilities of notable models such as GPT-3 and GPT-4. We introduce the concepts of tokens, few-shot learning, emergent abilities, and the significance of scaling laws. As we explore the functions and outputs of these models, we emphasize the potential challenges of hallucinations and biases. Additionally, we also discuss the context size limitation in LLMs.
- **The Evolution of LLMs and Transformers**: This lesson provides a chronological narrative of the progression in language modeling techniques. Starting with the foundational Bag of Words model from 1954, we navigate through significant milestones like TF-IDF, the groundbreaking Word2Vec with its semantic-rich word embeddings, and the sequence-processing capabilities of RNNs. Central to our exploration is the 2017 Transformer architecture, which set the stage for powerhouses like BERT, RoBERTa, and ELECTRA. This lesson only offers an overview without the deep technical intricacies, presenting a panoramic view of the model evolution in NLP.
- **A timeline of Large Language Models**: This lesson steers towards a comprehensive overview of the advancements in the Large Language Models landscape, spotlighting models that marked distinct milestones such as GPT-3, PaLM, and Galactica. Recognizing the significant role of techniques like scaling and alignment tuning in the unprecedented capabilities exhibited by LLMs, we untangle the principles that steer these giants. From exploring the enigmatic emergent abilities to decoding the scaling laws, this lesson explores the phenomena driving the potency and performance of LLMs.
- **Emergent Abilities in LLMs**: This lesson covers the unexpected skills that surface in Large Language Models as they grow beyond certain thresholds. As models expand, they exhibit unique capabilities influenced by factors like training compute. These emergent skills indicate performance leaps in LLMs as they scale, revealing unforeseen learning beyond what was initially anticipated.
- **Proprietary LLMs**: This lesson introduces prominent proprietary Large Language Models such as GPT-4, ChatGPT, and Cohere, among others. We'll weigh the advantages and drawbacks of proprietary models against open-source counterparts. Practical demonstrations will guide students in executing API calls for select models.
- **Open-Source LLMs**: This lesson offers insights into open-source Large Language Models, with a focus on LLaMA 2, Open Assistant, Dolly, and Falcon. We will explore their unique features, capabilities, and licensing details. Additionally, we'll discuss potential commercial uses and emphasize any restrictions within their licenses.
- **Understanding Hallucinations and Bias in LLMs**: This lesson focuses on the challenges posed by hallucinations and biases in Large Language Models. We'll define hallucinations, provide examples, and discuss their impact on LLM use cases. We'll also explore methods to minimize these issues, such as retriever architectures. The session also covers the concept of bias, its origin in LLMs, and potential mitigation strategies, including approaches like constitutional AI.
- **Applications and Use-Cases of LLMs:** This lesson highlights the leading applications and emerging trends of Large Language Models across industries. By referencing real-world news

and examples, we illustrate the transformative impact of LLMs across sectors. While emphasizing the vast potential benefits, the module also underscores the importance of recognizing LLMs' limitations and potential challenges.

This section provided a comprehensive overview of Large Language Models, highlighting their evolution and significant milestones. Topics ranged from understanding emergent abilities in LLMs to discerning between proprietary and open-source models. Critical challenges like hallucinations and biases were also addressed.

What are Large Language Models

Introduction

Welcome to our introductory module on Large Language Models or **LLMs**.

LLMs, or Large Language Models, are a specific category of neural network models characterized by having an exceptionally high number of parameters, often in the billions. These parameters are essentially the variables within the model that allow it to process and generate text. They are trained on vast quantities of textual data, which provides them with a broad understanding of language patterns and structures. The main goal of LLMs is to comprehend and produce text that closely resembles human-written language, enabling them to capture the subtle complexities of both syntax (the arrangement of words in a sentence) and semantics (the meaning conveyed by those words).

These models undergo training with a simple **objective: predicting the subsequent word** in a sentence. However, they develop a range of **emergent abilities** during this training process. For example, they can perform tasks such as **arithmetic calculations and word unscrambling** and even achieve remarkable feats like successfully passing professional-level exams such as the US Medical Licensing Exam.

They generate text in an **autoregressive manner**, generating the next tokens one by one based on the tokens they have previously generated.

The **attention mechanism** plays a key role in enabling these models to establish connections between words and produce coherent and contextually relevant text.

LLMs have significantly advanced the natural language processing (NLP) field, revolutionizing our approach to tasks like machine translation, natural language generation, part-of-speech tagging, parsing, information retrieval, and more.

As we dive further into this module, we will explore the capabilities of these models, their practical applications, and their exciting future possibilities.

Language Modeling

Language modeling is a fundamental task in Natural Language Processing (NLP). It involves explicitly learning the probability distribution of the words in a language. This is generally learned by **predicting the next token in a sequence**. This task is typically approached using statistical methods or deep learning techniques.

LLMs are trained to predict the next token (word, punctuation, etc.) based on the previous tokens in the text. The models achieve this by **learning the distribution of tokens in the training data**.

Tokenization

The first step in this process is tokenization, where the **input text** is **broken** down into **smaller units** called **tokens**. Tokens can be as small as individual characters or as large as whole words. The choice of token size can significantly affect the model's performance. Some models even use

subword tokenization, where words are broken down into smaller units that capture meaningful linguistic information.
For example, let’s consider the sentence "The child’s book.”
We could split the text whenever we find white space characters. The output would be:

```
["The", "child's", "book."]
```

As you can see, the punctuation is still attached to the words *"child’s"* and *"book."*
Otherwise, we could split the text according to white spaces and punctuation. The output would be:

```
["The", "child", "'", "s", "book", "."]
```

Importantly, tokenization is model-specific, meaning different models require different tokenization processes, which can complicate pre-processing and multi-modal modeling.

Model Architecture and Attention

The core of a language model is its architecture. Recurrent Neural Networks (**RNNs**) were traditionally used for this task, as they are capable of processing sequential data by maintaining an internal state that captures the information from previous tokens. However, they **struggle** with long sequences due to the **vanishing gradient problem**.
To overcome these limitations, transformer-based models have become the standard for language modeling tasks. These models use a mechanism called **attention**, which allows them to weigh the importance of different tokens when making predictions. This allows them to capture long-range dependencies between tokens and generate high-quality text.

Training

The model is trained on a large corpus of text to predict the next token of a sentence correctly. The goal is to adjust the model's parameters to maximize the probability of the observed data.
Typically a model is trained on a very large general dataset of texts from the Internet, such as The Pile or CommonCrawl. Sometimes also more specific datasets are used, such as the Stackoverflow Posts dataset.
The model learns to predict the next token in a sequence by adjusting its parameters to maximize the probability of outputting the correct next token from the training data.

Prediction

Once the model is trained, it can be used to generate text by predicting the next token in a sequence. This is done by feeding the sequence into the model, which outputs a probability distribution over the possible subsequent tokens. The next token is then chosen based on this distribution. This process can be repeated to generate sequences of arbitrary length.

Fine-Tuning

The model is often fine-tuned on a specific task **after pre-training**. This involves continuing the training process on a smaller, task-specific dataset. This allows the model to adapt its learned knowledge to the specific task (e.g. text translation) or specialized domain (e.g. biomedical, finance, etc), improving its performance.
This is a brief explanation, but the actual process can be much more complex, especially for state-of-the-art models like GPT-4. These models use advanced techniques and large amounts of data to achieve impressive results.

Context Size

The context size, or context window, in LLMs is the **maximum number** of **tokens** that the **model can handle in one go**. The context size is significant because it determines the length of the text that can be processed at once, which can impact the model's performance and the results it generates.

Different LLMs have different context sizes. For instance, the OpenAI "gpt-3.5-turbo-16k" model has a context window of 16,000 tokens. There is a natural limit to the number of tokens a model can produce. Smaller models can go up to 1k tokens, while larger models can go up to 32k tokens, like GPT-4.

Let's Generate Some Text

Let's try generating some text with LLMs. You must first generate an API key to use OpenAI's models in your Python environment. You can follow the below steps to generate the API key:

1. After creating an OpenAI account, log in.
2. After logging in, choose Personal from the top-right menu, then choose "View API keys."
3. The "Create new secret key" button is on the page containing API keys once step 2 has been finished. Clicking on that generates a secret key. Save this because it will be required in further lessons.

After that, you can save your key in a `.env` file like this:

```
OPENAI_API_KEY="<YOUR-OPENAI-API-KEY>"
```

Every time you start a Python script with the following lines, your key will be loaded into an environment variable called `OPENAI_API_KEY`Copy. This environment variable will then be used by the `openai`Copy library whenever you want to generate text.

```
from dotenv import load_dotenv

load_dotenv()
```

We are now ready to generate some text! Here's an example of it.

```
from dotenv import load_dotenv

load_dotenv()

import os

import openai

# English text to translate

english_text = "Hello, how are you?"


response = openai.ChatCompletion.create(

  model="gpt-3.5-turbo",

  messages=[

    {"role": "system", "content": "You are a helpful assistant."},

    {"role": "user", "content": f'Translate the following English text to French: "{english_text}"'}

  ],)
```

```
print(response['choices'][0]['message']['content'])
```

Code to run

```
Bonjour, comment ça va?
```

The output

By using `dotenv`, you can safely store sensitive information, such as API keys, in a separate file and avoid accidentally exposing it in your code. This is particularly important when working with open-source projects or sharing your code with others, as it ensures that the sensitive information remains secure.

Few-Shot Learning

Few-shot learning in the context of LLMs refers to providing the model with a few examples before making predictions. These examples "teach" the model how to reason and act as "filters" to help the model search for relevant patterns in the dataset.

The idea of few-shot learning is fascinating as it suggests that the model can be quickly reprogrammed for new tasks. While LLMs like GPT3 excel at language modeling tasks like machine translation, they may struggle with more complex reasoning tasks.

The few-shot examples are helping the model search for relevant patterns in the dataset. The dataset, which is effectively compressed into the model's weights, can be searched for patterns that strongly respond to these provided examples. These patterns are then used to generate the model's output. The more examples provided, the more precise the output becomes.

Here’s an example of few-shot learning:

```
from dotenv import load_dotenv

load_dotenv()

import os

import openai


# Prompt for summarization

prompt = """

Describe the following movie using emojis.


{movie}: """


examples = [

        { "input": "Titanic", "output": "🚢🌊♡⍰🎶🔥🛥💔👫💑" },

        { "input": "The Matrix", "output": "🕶💊💥👾🔮🌃👨💻🔄🔓💪" }

]
```

```python
movie = "Toy Story"

response = openai.ChatCompletion.create(

  model="gpt-3.5-turbo",

  messages=[

        {"role": "system", "content": "You are a helpful assistant."},

        {"role": "user", "content": prompt.format(movie=examples[0]["input"])},

        {"role": "assistant", "content": examples[0]["output"]},

        {"role": "user", "content": prompt.format(movie=examples[1]["input"])},

        {"role": "assistant", "content": examples[1]["output"]},

        {"role": "user", "content": prompt.format(movie=movie)},

  ]

)


print(response['choices'][0]['message']['content'])
```

Code to run

```
⍰🤠👤⍰🎢🌈🌟👫🚁👽🐶🚀
```

The output

Scaling Laws

Scaling laws refer to **the relationship between the model's performance and factors** such as the number of parameters, the size of the training dataset, the compute budget, and the network architecture. They were discovered after a lot of experiments and are described in the Chinchilla paper. These laws provide insights into how to allocate resources when training these models optimally.

The main elements characterizing a language model are:

1. The number of parameters (N) reflects the model's capacity to learn from data**. More parameters** allow the model to **capture complex patterns** in the data.
2. The size of the training dataset (D) is measured in **the number of tokens** (small pieces of text ranging from a few words to a single character).
3. **FLOPs** (floating point operations per second) measure the **compute budget** used for training.

The researchers trained the Chinchilla model, which has 70B parameters, on 1.4 trillion tokens. This aligns with the rule of thumb proposed in the paper**: for a model with X parameters, it is optimal to train it on approximately X * 20 tokens.** For example, in the context of this rule, a model with 100 billion parameters would be optimally trained on approximately 2 trillion tokens. Applying this rule, the Chinchilla model, though smaller, performed better than other LLMs. It showed gains in language modeling and task performance and needed less memory and computing power. You can read more about Chinchilla in its paper “Training Compute-Optimal Large Language Models”.

Emergent Abilities in LLMs

Emergent abilities in LLMs refer to the sudden appearance of new capabilities as the size of the model increases. These abilities, which include performing arithmetic, answering questions, summarizing passages, and more, are not explicitly trained in the model. Instead, they seem to arise spontaneously as the model scales, hence the term "emergent."
LLMs are probabilistic models that learn patterns in natural language. When these models are scaled up, they not only improve quantitatively in their ability to learn patterns, but they also exhibit qualitative changes in their behavior.

Traditionally, the models require task-specific fine-tuning and architectural modifications to perform specific tasks. However, when scaled, these models can perform these tasks without any architectural modifications or task-specific training. They can do this simply by phrasing the tasks in terms of natural language. This capability of LLMs to perform tasks without fine-tuning is remarkable in itself.
What's even more intriguing is how these abilities appear. As LLMs grow, they rapidly and unpredictably transition from near-zero to sometimes state-of-the-art performance. This phenomenon suggests that these abilities are emergent properties of the model's scale rather than being explicitly programmed into the model.
This concept of emergent abilities in LLMs has significant implications for the field of AI, as it suggests that scaling up models can lead to the spontaneous development of new capabilities.

Prompts

The text containing the instructions that we pass to LLMs is commonly known as prompts.
Prompts are **instructions given to AI** systems like OpenAI's GPT-3 and GPT-4, providing context to generate human-like text. The more detailed the prompt, the better the model's output.

Shorter, concise, and descriptive prompts tend to yield better results as they leave room for the LLM's creativity. Specific words or phrases can help narrow down potential outcomes and ensure relevant content generation.
Writing effective prompts requires a clear goal, simplicity, strategic use of keywords, and actionability. Testing the prompts before publishing ensures the output is relevant and error-free.
Here are some prompting tips:

1. Use **precise language** when crafting a prompt – this will help ensure accuracy in the generated output:
    - Less Precise Prompt: "Write about dogs."
    - More Precise Prompt: "Write a 500-word informative article about the dietary needs of adult Golden Retrievers."
2. Provide **enough context** around each prompt – this will give a better understanding of what kind of output should be produced:
    - Less Contextual Prompt: "Write a story."
    - More Contextual Prompt: "Write a short story set in Victorian England featuring a young detective solving his first major case."
3. Test different **variations** of each prompt – this allows you to experiment with different approaches until you find one that works best:
    - Initial Prompt: "Write a blog post about the benefits of yoga."
    - Variation 1: "Compose a 1000-word blog post detailing the physical and mental benefits of regular yoga practice."
    - Variation 2: "Create an engaging blog post that highlights the top 10 benefits of incorporating yoga into daily routine."

4. **Review** generated outputs before publishing them – while most automated systems produce accurate results, occasionally mistakes occur so it's always wise to double-check everything before releasing any content into production environments:
    - Before Review: "Yoga is a great way to improve your flexibility and strength. It can also help reduce stress and improve mental clarity. However, it's important to remember that all yoga poses are suitable for everyone."
    - After Review (correcting inaccurate information): "Yoga is a great way to improve your flexibility and strength. It can also help reduce stress and improve mental clarity. However, it's important to remember that not all yoga poses are suitable for everyone. Always consult with a healthcare professional before starting any new exercise regimen."

Hallucinations and Biases in LLMs

The term **hallucinations** refers to instances where AI systems generate outputs, such as text or images, that **don't align with real-world facts** or inputs. For example, ChatGPT might generate a plausible-sounding answer to an entirely incorrect factual question.

Hallucinations in LLMs refer to instances where the model generates outputs that do not align with real-world facts or context. This can lead to the propagation of **misinformation**, especially in critical sectors like healthcare and education where the accuracy of information is of utmost importance. Similarly, bias in LLMs can result in outputs that favor certain perspectives over others, potentially leading to the reinforcement of harmful stereotypes and discrimination.

Consider an interaction where a user asks, "Who won the World Series in 2025?" If the LLM responds with, "The New York Yankees won the World Series in 2025," it's a clear case of hallucination. As of now (July 2023), the 2025 World Series hasn't taken place, so any claim about its outcome is a fabrication.

Bias in AI and LLMs is another significant issue. It refers to these models' inclination to **favor** specific outputs or decisions based on their training data. If the training data is predominantly from a specific region, the model might show a bias toward that region's language, culture, or perspectives. If the training data contains inherent biases, such as gender or racial bias, the AI system might produce skewed or discriminatory outputs.

For example, if a user asks an LLM, "Who is a nurse?" and it responds with, "She is a healthcare professional who cares for patients in a hospital," it shows a gender bias. The model automatically associates nursing with women, which doesn't accurately reflect the reality where both men and women can be nurses.

Mitigating hallucinations and bias in AI systems involves refining model training, using verification techniques, and ensuring the training data is diverse and representative. Finding a balance between maximizing the model's potential and avoiding these issues remains challenging.
Interestingly, in creative domains like media and fiction writing, these "hallucinations" can be beneficial, enabling the generation of unique and innovative content.

The ultimate goal is to develop LLMs that are not only powerful and efficient but also reliable, fair, and trustworthy. By doing so, we can maximize the potential of LLMs while minimizing their risks, ensuring that the benefits of this technology are accessible to all.

Conclusion

In this introductory module, we explored the fascinating world of LLMs. These powerful models, trained on vast amounts of text data, can understand and generate human-like text. They're built on transformer architectures, allowing them to capture long-range dependencies in language and generate text in an autoregressive manner.

We covered the capabilities of LLMs, discussing their impact on the field of NLP. We've learned about few-shot learning, scaling laws, and the emergent abilities of these models.

We also acknowledged the challenges that come with these models, including hallucinations and biases, emphasizing the importance of mitigating these issues.

In the next lesson, we'll see a timeline of machine learning models used for language modeling up to the beginning of Large Language Models.

The Evolution of Language Modeling up to LLMs

Introduction

In this lesson, we'll see the most popular models used for language modeling, starting from statistical ones up to the first Large Language Models (LLMs). This lesson is meant to be more like a narrative on the evolution of the models rather than a technical explanation. Therefore, don't worry if you can't understand every model in detail.

The Evolution of Language Modeling

The evolution of NLP models has been a remarkable journey marked by continuous innovation and improvement. It began with the Bag of Words model in 1954, which simply counted word occurrences in documents. This was followed by TF-IDF in 1972, which adjusted these scores based on the rarity or commonality of words.

The advent of Word2Vec in 2013 marked a significant leap forward, introducing the concept of word embeddings that captured semantic relationships between words.

This was then further enhanced by Recurrent Neural Networks (RNNs), which could learn sequence patterns and handle documents of any length.

The introduction of the Transformer architecture in 2017 revolutionized the field, with its attention mechanism allowing the model to focus on the most relevant parts of the input when generating output. This was the foundation for BERT in 2018, which used bidirectional Transformers to achieve impressive results in traditional NLP tasks.

The subsequent years saw a flurry of advancements, with models like RoBERTa, XLM, ALBERT, and ELECTRA each introducing their own improvements and optimizations.

Model's Timeline

- **[1954]** Bag of Words (BOW)

  BOW is a simple model that counts word occurrences in documents, using these counts as features. It was a basic yet effective way to analyze text. However, it did not account for word order or context.
- **[1972]** TF-IDF

  TF-IDF enhanced BOW by giving more weight to rare words and less to common ones. This improved the model's ability to discern document relevance. However, it still did not account for word context.

- **[2013]** Word2Vec
  Word2Vec introduced word embeddings, high-dimensional vectors that capture semantic relationships. These embeddings were learned by a neural network trained on a large corpus of text. This model marked a significant advancement in capturing semantic meaning in text.
- **[2014]** RNNs in Encoder-Decoder architectures
  RNNs (Recurrent Neural Networks) compute document embeddings, leveraging word context in sentences, which was not possible with word embeddings alone. Later evolved with **LSTM** [1997] to capture long-term dependencies and to **Bidirectional RNN** [1997] to capture left-to-right and right-to-left dependencies. Eventually, **Encoder-Decoder RNNs** [2014] emerged, where an RNN creates a document embedding (i.e., the encoder), and another RNN decodes it into text (i.e., the decoder).
- **[2017]** Transformer
  The Transformer is an encoder-decoder model that leverages attention mechanisms to compute better embeddings and to align output better to input. This model marked a significant advancement in NLP tasks.
- **[2018]** BERT
  BERT is a bidirectional Transformer pre-trained using a combination of Masked Language Modeling and Next Sentence Prediction objectives. It uses global attention.
- **[2018]** GPT
  GPT is the first autoregressive model based on the Transformer architecture. Later, it evolved into **GPT-2** [2019], a bigger & optimized version of GPT pre-trained on WebText, & **GPT-3** [2020], a further bigger and optimized version of GPT-2, pre-trained on Common Crawl.
- **[2019]** CTRL
  CTRL, similar to GPT, introduced control codes for conditional text generation. This allowed for more control over the generated text.
- **[2019]** Transformer-XL
  Transformer-XL reused previously computed hidden states to attend to a longer context. This allowed the model to handle longer sequences of text.
- **[2019]** ALBERT
  ALBERT is a lighter version of BERT where (1) Next Sentence Prediction is replaced by Sentence Order Prediction, and (2) parameter-reduction techniques are used for lower memory consumption and faster training.
- **[2019]** RoBERTa
  RoBERTa is a better version of BERT, where (1) the Masked Language Modeling objective is dynamic, (2) the Next Sentence Prediction objective is dropped, (3) the BPE tokenizer is employed, and (4) better hyperparameters are used.
- **[2019]** XLM
  XLM, a multilingual Transformer, was pre-trained using objectives like Causal Language Modeling, Masked Language Modeling, and Translation Language Modeling.
- **[2019]** XLNet
  It's a Transformer-XL with a generalized autoregressive pre-training method that enables learning bidirectional dependencies.
- **[2019]** PEGASUS
  PEGASUS, a bidirectional encoder and left-to-right decoder, was pre-trained with Masked Language Modeling and Gap Sentence Generation objectives.
- **[2019]** DistilBERT
  It is the same as BERT but smaller and faster while preserving over 95% of BERT's performances. Trained by distillation of the pre-trained BERT model.

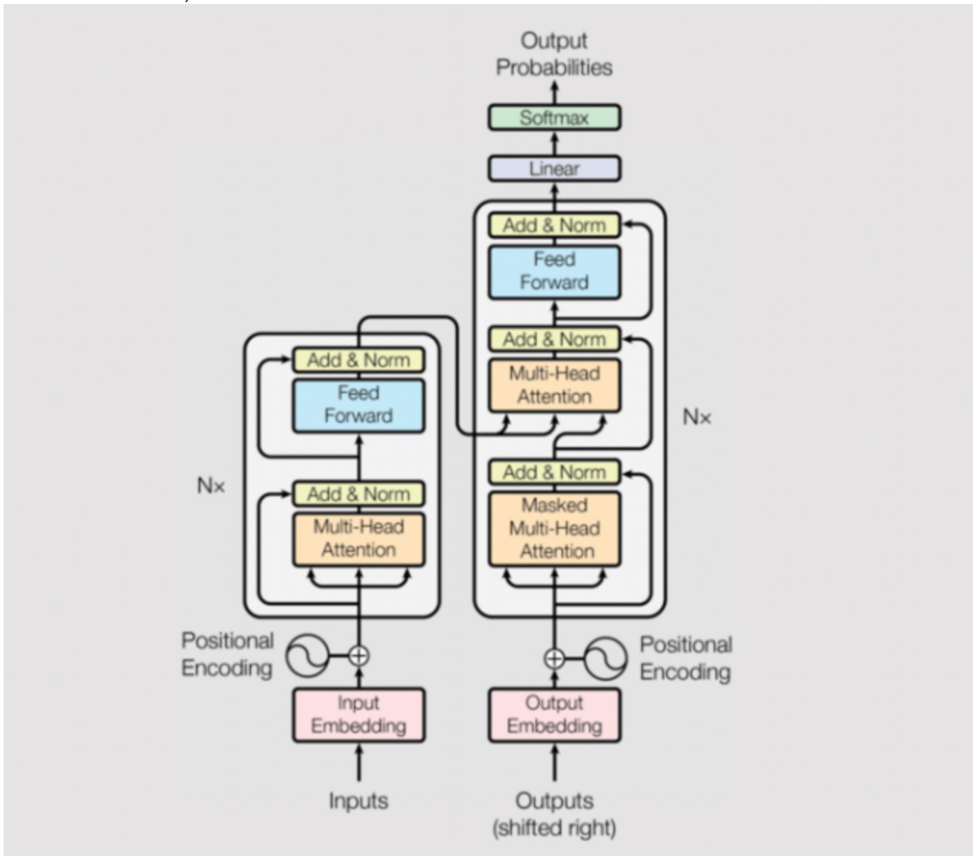
- **[2019]** XLM-RoBERTa
  XLM-RoBERTa is a multilingual version of RoBERTa, trained on a multilanguage corpus with the Masked Language Modeling objective.
- **[2019]** BART
  BART, a bidirectional encoder and left-to-right decoder, was trained by corrupting text with an arbitrary noising function and learning a model to reconstruct the original text.
- **[2019]** ConvBERT
  ConvBERT replaced self-attention blocks with new ones that leveraged convolutions to better model global and local contexts.
- **[2020]** Funnel Transformer
  It's a type of Transformer that gradually compresses the sequence of hidden states to a shorter one, reducing the computation cost.
- **[2020]** Reformer
  Reformer is a more efficient Transformer thanks to local-sensitive hashing attention, axial position encoding, and other optimizations.
- **[2020]** T5
  T5, a bidirectional encoder and left-to-right decoder, was pre-trained on a mix of unsupervised and supervised tasks.
- **[2020]** Longformer
  Longformer replaced the attention matrices with sparse matrices for higher training efficiency. This made the model faster and more memory-efficient.
- **[2020]** ProphetNet
  ProphetNet was trained with the Future N-gram Prediction objective and with a novel self-attention mechanism.
- **[2020]** ELECTRA
  Lighter and better than BERT, ELECTRA was trained with the Replaced Token Detection objective. This made the model more efficient and improved its performance on NLP tasks.
- **[2021]** Switch Transformers
  Switch Transformers introduced a sparsely-activated expert Transformer model, aiming to simplify and improve over Mixture of Experts. This allowed the model to handle a wider range of tasks.
  The years 2020 and 2021 are the ones where Large Language Models truly arose. Up to 2020, most language models were able to generate good-looking texts. After this date, the best language models could follow instructions and solve various tasks aside from simple text generation.

The Transformer

The most crucial model of the previous pipeline is, without doubt, the Transformer, introduced in the very popular paper "Attention Is All You Need." The Transformer is a type of neural network that is used today by all of the best Large Language Models like GPT-4, Claude, and LLaMA.
Central to Transformers is the encoder-decoder structure, which excels at modeling long-range dependencies and capturing contextual information.

The Encoder processes the input text, identifying key elements and creating word embeddings based on their relevance to other words in the sentence. In the original Transformer architecture, designed for text translation, the attention mechanism was employed in two distinct ways: encoding the source language and decoding the encoded embedding back into the target language.

On the other hand, **the Decoder** takes the encoder's output, an embedding, and transforms it back into text. Some models may opt to use only the decoder, bypassing the encoder entirely. The decoder's attention mechanism differs slightly from the encoder's, functioning more like a conventional language model by focusing on previous words during text processing. This approach is particularly useful for tasks like **language generation**, which is why models like GPT, primarily designed for text generation in response to an input text sequence, utilize the decoder part of the Transformer.
Later in the course, we'll learn more about the Transformer architecture.



Scaling Transformers: What Lead to Large Language Models

The effectiveness of the Transformer models was further improved by **scaling**, i.e., increasing the number of parameters and training on more data. This scaling led to models with more than 100B parameters that could perform tasks using few-shot or zero-shot approaches, eliminating the need for fine-tuning specific tasks.
The increase in the size of these models and the datasets used for training them (and thus the associated costs) led to the large language models that we see today, like Cohere Command, GPT-4, and LLaMA.

Conclusion

In this lesson, we navigated through the rich history of Natural Language Processing, tracing the path from the rudimentary Bag of Words model to the advanced Transformer family. This timeline underscored the continuous innovation in NLP, spotlighting the progression of models in sophistication and proficiency. In the next lesson, we'll continue the timeline of popular models from 2020 (with GPT-3) up to today.

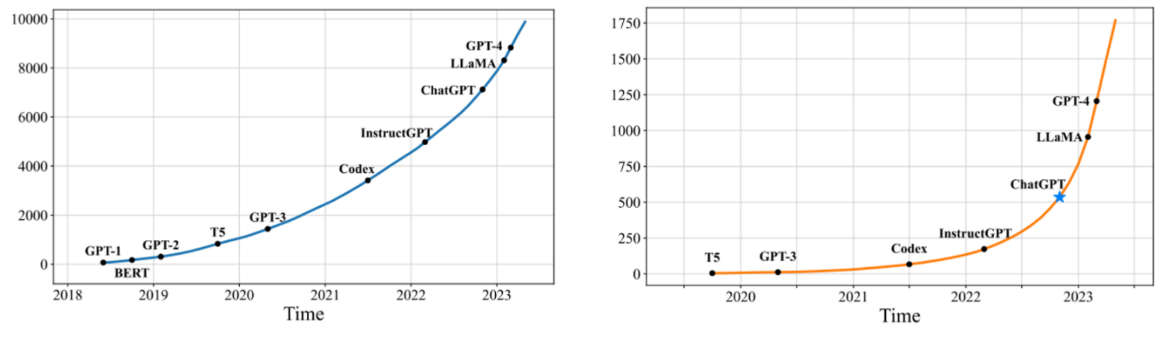
A Timeline of Large Language Models

Introduction

In this lesson, we'll explore the transformative shifts that have reshaped AI, exploring the key features that set LLMs apart from their predecessors. We'll see how scaling laws, emergent abilities, and innovative architectures have propelled LLMs to tackle complex tasks and define the current landscape of popular LLMs.

From Language Models to Large Language Models

The evolution of language models has undergone a transformative shift from pre-trained language models (LMs) to the emergence of large language models (LLMs).
LMs, like ELMo and BERT, initially captured context-aware word representations through pre-training and fine-tuning for specific tasks. However, the introduction of LLMs, exemplified by GPT-3 and PaLM, demonstrated that scaling model size and data can unlock emergent abilities, exceeding the capabilities of their smaller counterparts. These LLMs can tackle **more complex tasks through in-context learning**.
The following image shows the trends of the cumulative numbers of arXiv papers containing the keyphrases "language model" and "large language model," emphasizing the growing interest in them in the last years.



(a) Query="Language Model" (b) Query="Large Language Model"

Fig. 1: The trends of the cumulative numbers of arXiv papers that contain the keyphrases "*language model*" (since June 2018) and "*large language model*" (since October 2019), respectively. The statistics are calculated using exact match by querying the keyphrases in title or abstract by months. We set different x-axis ranges for the two keyphrases, because "language models" have been explored at an earlier time. We label the points corresponding to important landmarks in the research progress of LLMs. A sharp increase occurs after the release of ChatGPT: the average number of published arXiv papers that contain "*large language model*" in title or abstract goes from 0.40 per day to 8.58 per day (Figure 1(b)).

From "A Survey of Large Language Models" paper

Key Characterizing Features of LLMs

Here are the main characteristics that differentiate LLMs from the previous models:

1. **Scaling Laws for Enhanced Capacity:** Scaling laws play a crucial role in LLM development, indicating a relationship between model performance, model size, dataset size, and training compute. The **KM scaling laws** emphasize the impact of these factors, revealing distinct formulas for their influence on cross-entropy loss. The **Chinchilla scaling laws** provide an alternate approach, optimizing compute allocation between model and data size.
2. **Emergent Abilities:** LLMs possess emergent abilities, defined as capabilities that manifest in large models but are absent in smaller counterparts. One prominent emergent ability is *in-context*

learning (ICL), showcased by models like GPT-3. ICL allows LLMs to generate unexpected outputs based on natural language instructions, eliminating the need for further training.

3. **Instruction following**: LLMs can be finetuned to follow text instructions, which further enhances generalization for new tasks.
4. **Step-by-Step Reasoning:** LLMs can perform *step-by-step reasoning* using the chain-of-thought (CoT) prompting strategy. This mechanism enables them to solve complex tasks by breaking them down into intermediate reasoning steps, which is particularly beneficial for tasks involving multiple steps like mathematical word problems.

A Timeline of the Most Popular LLMs

Here's an overview of the timeline of the most popular LLMs in the last years.

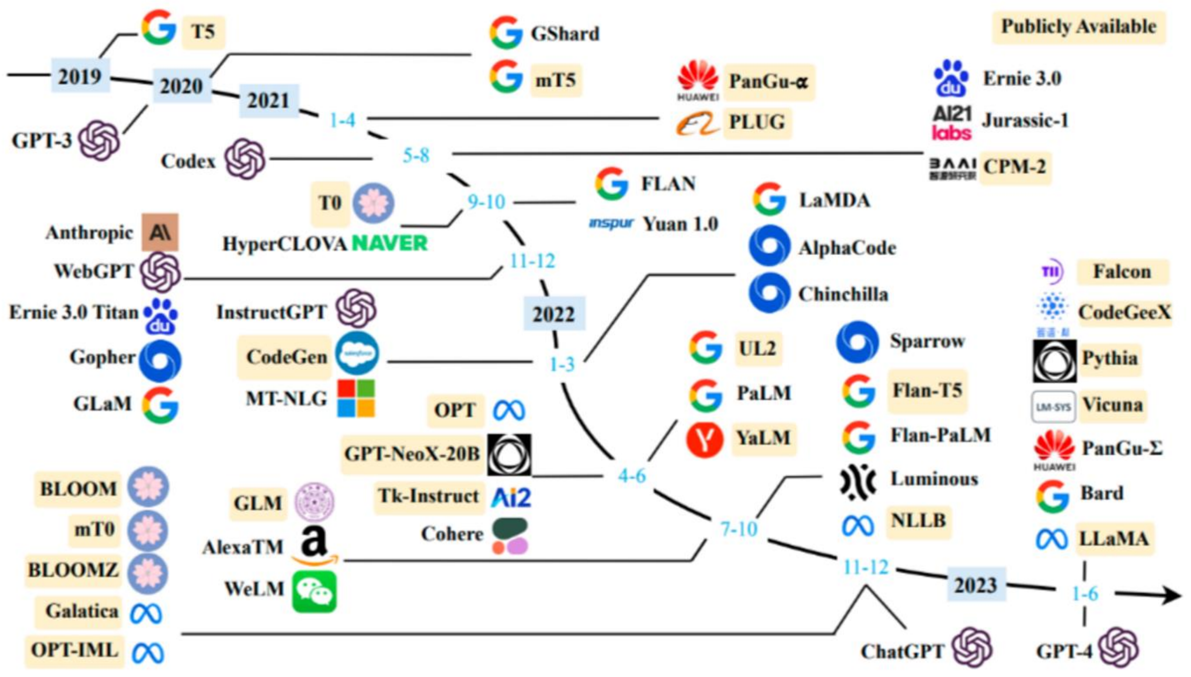


Fig. 2: A timeline of existing large language models (having a size larger than 10B) in recent years. The timeline was established mainly according to the release date (*e.g.*, the submission date to arXiv) of the technical paper for a model. If there was not a corresponding paper, we set the date of a model as the earliest time of its public release or announcement. We mark the LLMs with publicly available model checkpoints in yellow color. Due to the space limit of the figure, we only include the LLMs with publicly reported evaluation results.

From "A Survey of Large Language Models" paper

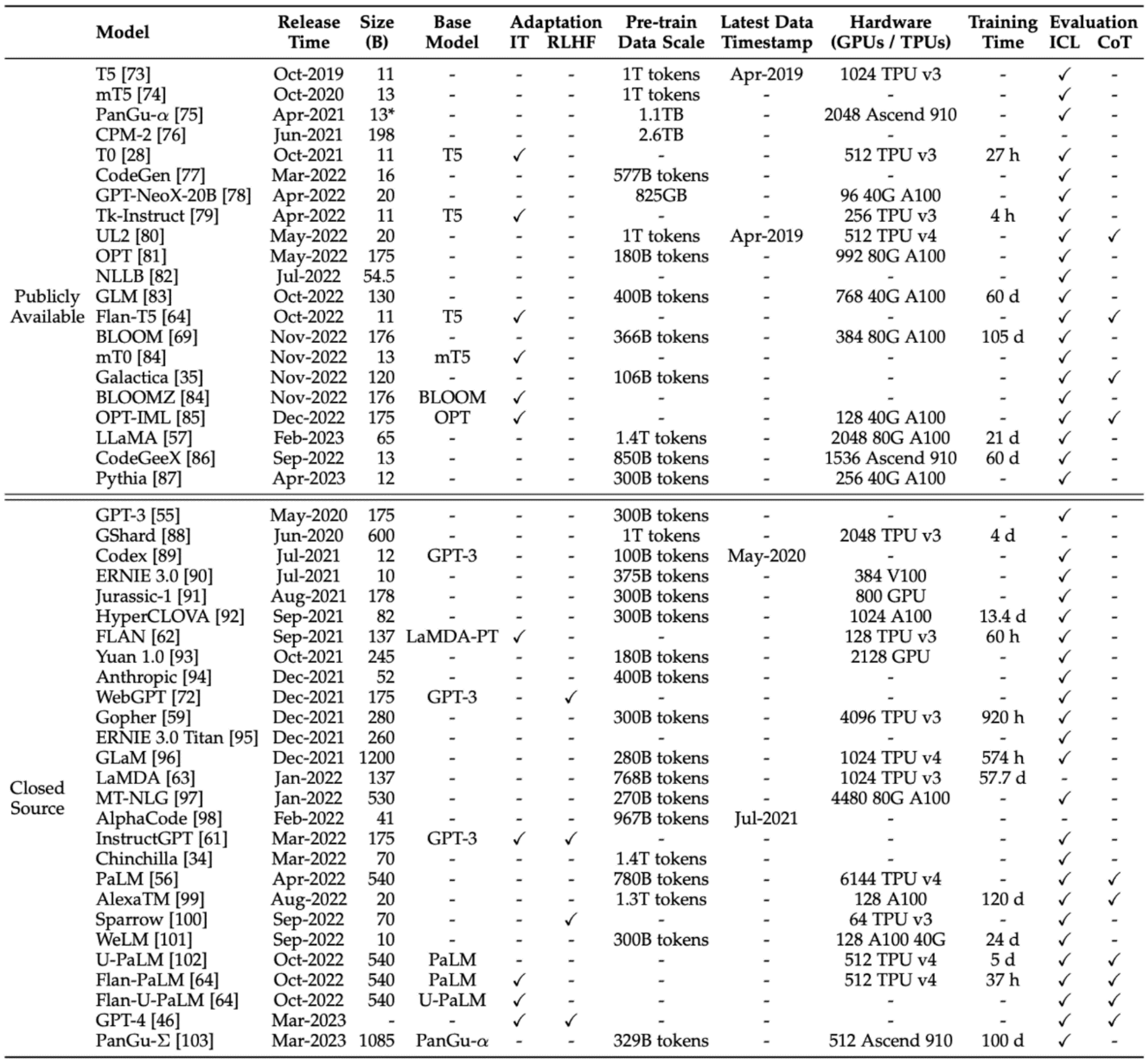
Here is a brief description of some of them.

- **[2018]** GPT-1
  GPT-1 (Generative Pre-Training 1) was introduced by OpenAI in 2018. It laid the foundation for the GPT-series models by employing a **generative, decoder-only** Transformer architecture. It combined **unsupervised pretraining and supervised fine-tuning** to predict the next word in natural language text.
- **[2019]** GPT-2
  Building upon the architecture of GPT-1, GPT-2 was released in 2019 with an increased parameter scale of 1.5 billion. This model demonstrated potential for solving a variety of tasks using language text as a unified format for input, output, and task information.

- **[2020]** GPT-3
  Released in 2020, GPT-3 marked a significant capacity leap by scaling the model to 175 billion parameters. It introduced the concept of **in-context learning (ICL),** enabling LLMs to understand tasks through **few-shot or zero-shot learning**. GPT-3 showcased excellent performance in numerous NLP tasks, including reasoning and domain adaptation, highlighting the potential of scaling up model size.
- **[2021]** Codex
  Codex was introduced by OpenAI in July 2021 as a **fine-tuned version of GPT-3** specifically trained on a large **corpus of GitHub code**. It demonstrated enhanced ability in solving **programming and mathematical problems**, showcasing the potential of training LLMs on specialized data.
- **[2021]** LaMDA
  LaMDA (Language Models for Dialog Applications) was introduced by researchers from DeepMind. LaMDA focuses on enhancing **dialog applications and dialog generation** tasks. It has a significant number of parameters, with the largest model consisting of 137 billion parameters, making it slightly smaller than GPT-3.
- **[2021]** Gopher
  In 2021, DeepMind introduced Gopher, a language model with an impressive parameter scale of 280 billion. Notably, Gopher demonstrated a remarkable capability to approach human expert performance on the **Massive Multitask Language Understanding (MMLU)** benchmark. However, like its predecessors, Gopher exhibited certain limitations, including tendencies for repetition, biases, and propagation of incorrect information.
- **[2022]** InstructGPT
  In 2022, InstructGPT was proposed as an enhancement to GPT-3 for human alignment. It utilized **reinforcement learning from human feedback (RLHF)** to improve the model's instruction-following capacity and mitigate issues like generating harmful content. This approach proved valuable for training LLMs to align with human preferences.
- **[2022]** Chinchilla
  Chinchilla, introduced in 2022 by DeepMind, is a family of large language models that build upon the discovered scaling laws of LLMs. With a focus on efficient utilization of compute resources, Chinchilla boasts 70 billion parameters and achieves a remarkable 67.5% accuracy on the MMLU benchmark—a 7% improvement over Gopher.
- **[2022]** PaLM
  Pathways Language Model (PaLM) was introduced by Google Research in 2022, showcasing a leap in model scale with a whopping 540 billion parameters. Leveraging the proprietary Pathways system for distributed computation, PaLM exhibited great few-shot performance across an array of language understanding, reasoning, and code-related tasks.
- **[2022]** ChatGPT
  In November 2022, OpenAI released ChatGPT, a **conversation model based** on GPT-3.5 and GPT-4. Specially **optimized** for **dialogue**, ChatGPT exhibited great abilities in communicating with humans, reasoning, and aligning with human values.
- **[2023]** LLaMA
  LLaMA (Large Language Model Meta AI) emerged in February 2023 from Meta AI. It introduced a family of large language models available in varying sizes from **7 billion to 65 billion** parameters. LLaMA's release marked a departure from the limited access trend, as its model weights were made available to the research community under a noncommercial license. Subsequent developments, including Llama 2 and other chat models, further emphasized accessibility, this time with a license for commercial use.
- **[2023]** GPT-4

In March 2023, GPT-4 was released, extending **text input to multimodal signals**. With stronger capacities than GPT-3.5, GPT-4 demonstrated significant performance improvements on various tasks.

If you want to dive deeper into these models, I suggest reading the paper "A Survey of Large Language Models." Here's a table summarizing the architectural and training details of all the mentioned models (and others).



	Model	Release Time	Size (B)	Base Model	Adaptation IT	RLHF	Pre-train Data Scale	Latest Data Timestamp	Hardware (GPUs / TPUs)	Training Time	Evaluation ICL	CoT
Publicly Available	T5 [73]	Oct-2019	11	-	-	-	1T tokens	Apr-2019	1024 TPU v3	-	✓	-
	mT5 [74]	Oct-2020	13	-	-	-	1T tokens	-	-	-	✓	-
	PanGu-α [75]	Apr-2021	13*	-	-	-	1.1TB	-	2048 Ascend 910	-	✓	-
	CPM-2 [76]	Jun-2021	198	-	-	-	2.6TB	-	-	-	-	-
	T0 [28]	Oct-2021	11	T5	✓	-	-	-	512 TPU v3	27 h	✓	-
	CodeGen [77]	Mar-2022	16	-	-	-	577B tokens	-	-	-	✓	-
	GPT-NeoX-20B [78]	Apr-2022	20	-	-	-	825GB	-	96 40G A100	-	✓	-
	Tk-Instruct [79]	Apr-2022	11	T5	✓	-	-	-	256 TPU v3	4 h	✓	-
	UL2 [80]	May-2022	20	-	-	-	1T tokens	Apr-2019	512 TPU v4	-	✓	✓
	OPT [81]	May-2022	175	-	-	-	180B tokens	-	992 80G A100	-	✓	-
	NLLB [82]	Jul-2022	54.5	-	-	-	-	-	-	-	✓	-
	GLM [83]	Oct-2022	130	-	-	-	400B tokens	-	768 40G A100	60 d	✓	-
	Flan-T5 [64]	Oct-2022	11	T5	✓	-	-	-	-	-	✓	✓
	BLOOM [69]	Nov-2022	176	-	-	-	366B tokens	-	384 80G A100	105 d	✓	-
	mT0 [84]	Nov-2022	13	mT5	✓	-	-	-	-	-	✓	-
	Galactica [35]	Nov-2022	120	-	-	-	106B tokens	-	-	-	✓	✓
	BLOOMZ [84]	Nov-2022	176	BLOOM	✓	-	-	-	-	-	✓	-
	OPT-IML [85]	Dec-2022	175	OPT	✓	-	-	-	128 40G A100	-	✓	✓
	LLaMA [57]	Feb-2023	65	-	-	-	1.4T tokens	-	2048 80G A100	21 d	✓	-
	CodeGeeX [86]	Sep-2022	13	-	-	-	850B tokens	-	1536 Ascend 910	60 d	✓	-
	Pythia [87]	Apr-2023	12	-	-	-	300B tokens	-	256 40G A100	-	✓	-
Closed Source	GPT-3 [55]	May-2020	175	-	-	-	300B tokens	-	-	-	✓	-
	GShard [88]	Jun-2020	600	-	-	-	1T tokens	-	2048 TPU v3	4 d	-	-
	Codex [89]	Jul-2021	12	GPT-3	-	-	100B tokens	May-2020	-	-	✓	-
	ERNIE 3.0 [90]	Jul-2021	10	-	-	-	375B tokens	-	384 V100	-	✓	-
	Jurassic-1 [91]	Aug-2021	178	-	-	-	300B tokens	-	800 GPU	-	✓	-
	HyperCLOVA [92]	Sep-2021	82	-	-	-	300B tokens	-	1024 A100	13.4 d	✓	-
	FLAN [62]	Sep-2021	137	LaMDA-PT	✓	-	-	-	128 TPU v3	60 h	✓	-
	Yuan 1.0 [93]	Oct-2021	245	-	-	-	180B tokens	-	2128 GPU	-	✓	-
	Anthropic [94]	Dec-2021	52	-	-	-	400B tokens	-	-	-	✓	-
	WebGPT [72]	Dec-2021	175	GPT-3	-	✓	-	-	-	-	✓	-
	Gopher [59]	Dec-2021	280	-	-	-	300B tokens	-	4096 TPU v3	920 h	✓	-
	ERNIE 3.0 Titan [95]	Dec-2021	260	-	-	-	-	-	-	-	✓	-
	GLaM [96]	Dec-2021	1200	-	-	-	280B tokens	-	1024 TPU v4	574 h	✓	-
	LaMDA [63]	Jan-2022	137	-	-	-	768B tokens	-	1024 TPU v3	57.7 d	-	-
	MT-NLG [97]	Jan-2022	530	-	-	-	270B tokens	-	4480 80G A100	-	✓	-
	AlphaCode [98]	Feb-2022	41	-	-	-	967B tokens	Jul-2021	-	-	-	-
	InstructGPT [61]	Mar-2022	175	GPT-3	✓	✓	-	-	-	-	✓	-
	Chinchilla [34]	Mar-2022	70	-	-	-	1.4T tokens	-	-	-	✓	-
	PaLM [56]	Apr-2022	540	-	-	-	780B tokens	-	6144 TPU v4	-	✓	✓
	AlexaTM [99]	Aug-2022	20	-	-	-	1.3T tokens	-	128 A100	120 d	✓	✓
	Sparrow [100]	Sep-2022	70	-	-	✓	-	-	64 TPU v3	-	✓	-
	WeLM [101]	Sep-2022	10	-	-	-	300B tokens	-	128 A100 40G	24 d	✓	-
	U-PaLM [102]	Oct-2022	540	PaLM	-	-	-	-	512 TPU v4	5 d	✓	✓
	Flan-PaLM [64]	Oct-2022	540	PaLM	✓	-	-	-	512 TPU v4	37 h	✓	✓
	Flan-U-PaLM [64]	Oct-2022	540	U-PaLM	✓	-	-	-	-	-	✓	✓
	GPT-4 [46]	Mar-2023	-	-	✓	✓	-	-	-	-	✓	✓
	PanGu-Σ [103]	Mar-2023	1085	PanGu-α	-	-	329B tokens	-	512 Ascend 910	100 d	✓	-

From "A Survey of Large Language Models" paper

Moreover, here's an image showing the evolution of the LLaMA models into other fine-tuned models made by online communities, highlighting the great interest it sparked.

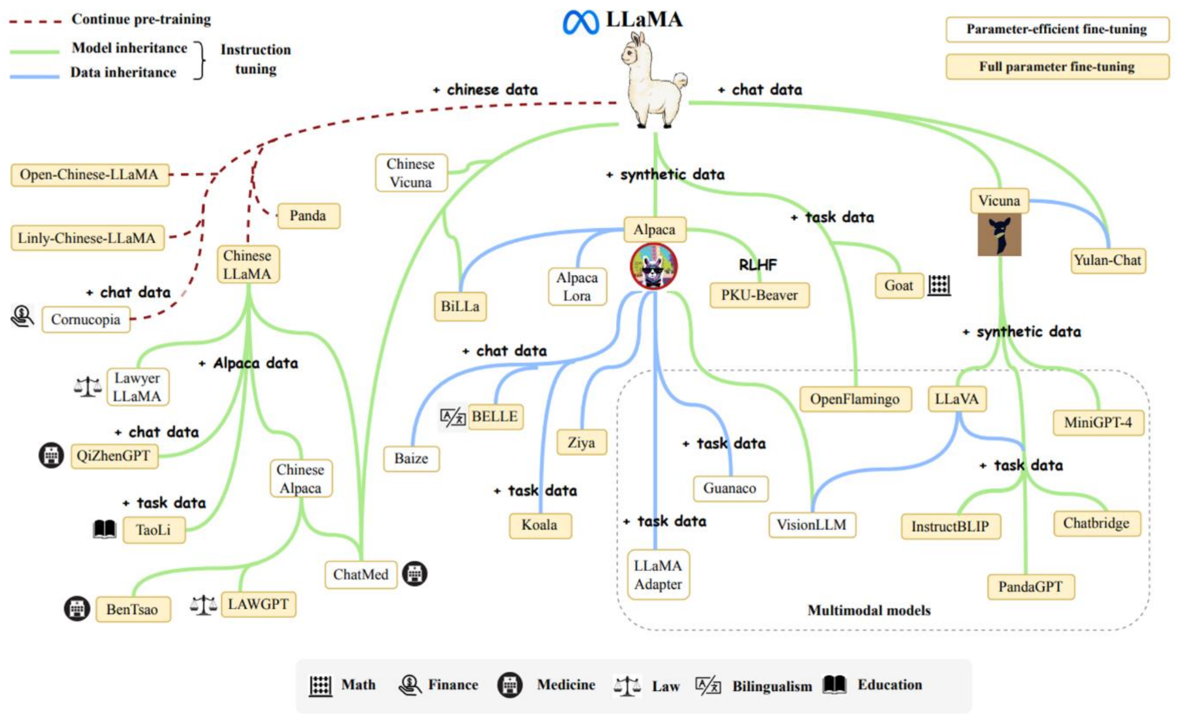


Fig. 4: An evolutionary graph of the research work conducted on LLaMA. Due to the huge number, we cannot include all the LLaMA variants in this figure, even much excellent work. To support incremental update, we share the source file of this figure, and welcome the readers to include the desired models by submitting the pull requests on our GitHub page.

From “A Survey of Large Language Models” paper

Conclusion

In this lesson, we learned more about the transition from pre-trained language models (LMs) to the emergence of large language models (LLMs). We explored the key differentiating features of LLMs, including the influence of scaling laws and the manifestation of emergent abilities like in-context learning, step-by-step reasoning strategies, and instruction following.
We also saw a brief timeline of the most popular LLMs: from the foundational GPT-1 to the revolutionary GPT-3, the specialized Codex, LaMDA, Gopher, and Chinchilla, to PaLM, ChatGPT, and LLaMA.

Emergent Abilities in LLMs

An ability is considered as emergent when larger models exhibit it, but it's absent in smaller models - a key factor contributing to the success of Large Language Models.

Introduction

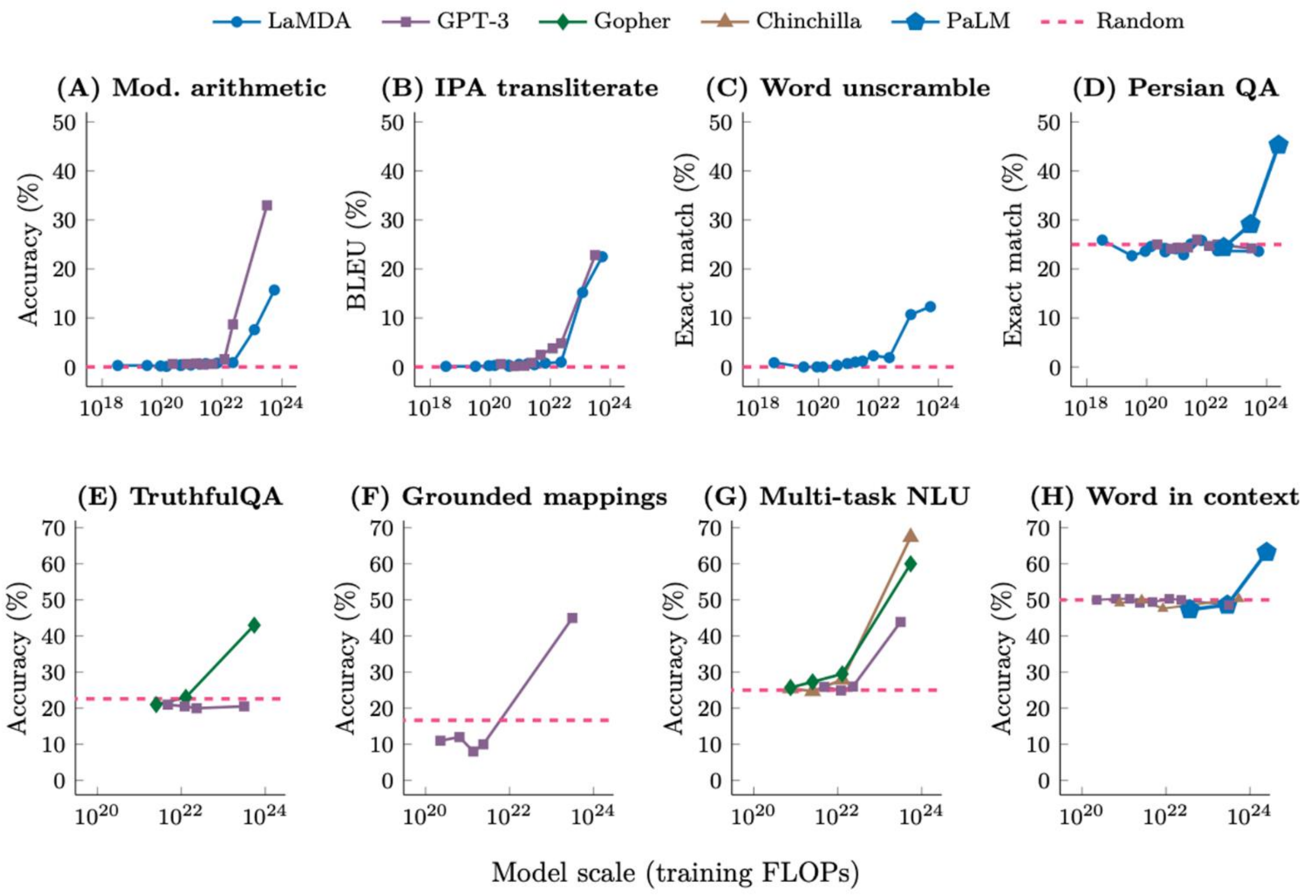
In this lesson, we’ll dive more into the concept of **emergent abilities**, the empirical phenomenon of the new abilities that language models get when their size increases over specific thresholds.
Emergent abilities become apparent as we scale up the models and are influenced by factors such as training compute and model parameters.
We'll also explore various instances of these emergent abilities, focusing on scenarios like few-shots and augmented prompting, and examine the reasons behind the emergence of these abilities and whether further scaling could reveal more of them.

What Are Emergent Abilities

Emergent abilities in LLMs are defined as significant improvements in task performance that become apparent as the model size or scale increases. These abilities, which are not present or noticeable in smaller or less complex models, become evident in larger or more complex models. This suggests that the model is learning and generalizing from its pre-training in ways that were not explicitly programmed or expected.

When visualized on a scaling curve, emergent abilities show a pattern where performance is almost random until a certain scale threshold, after which performance increases significantly. This is known as a **phase transition**, a dramatic change in behavior that could not have been predicted by examining smaller-scale systems.

In the following image, taken from the paper "Emergent Abilities of Large Language Models," we see several charts showing the emergence of abilities of LLMs (whose performance is shown on the y-axis) with respect to the model scale (shown on the x-axis).



From the paper "Emergent Abilities of Large Language Models"

Language models have been scaled primarily along computation amount, model parameters, and training dataset size. The emergence of abilities may occur with less training computation or fewer model parameters for models trained on higher-quality data. It also depends on factors such as the amount of data, its quality, and the number of parameters in the model.

Emergent abilities in LLMs appear as the models scale up and cannot be predicted by simply extrapolating from smaller models.

Evaluation Benchmarks for Emergent Abilities

Several benchmarks are used to evaluate the emergent abilities of language models. These include the BIG-Bench suite, TruthfulQA, the Massive Multi-task Language Understanding (MMLU) benchmark, and the Word in Context (WiC) benchmark.

1. The first of these is the **BIG-Bench suite**, a comprehensive set of over 200 benchmarks that test a model's capabilities across a variety of tasks. These tasks include **arithmetic operations** where the model is expected to perform the four basic operations (example: "Q: What is 132 plus 762? A: 894), transliteration from the International Phonetic Alphabet (IPA) to measure if the model is able to manipulate and use rare words (example: "English: The 1931 Malay census was an alarm bell. IPA: ðə 1931 ˈmeɪleɪ ˈsɛnsəs wɑz ən əˈlɑrm bɛl."), word unscrambling that analyzes the model's ability to work with alphabets. A large number of benchmarks can be found within the Github repository where you can delve into their specific details. The performance of models like GPT-3 and LaMDA on these tasks starts near zero but jumps to significantly above random at a certain scale, demonstrating emergent abilities.
2. Another benchmark is **TruthfulQA**, which measures a model's capacity to provide **truthful responses** when addressing questions. The evaluation consists of two tasks: 1) Generation: The model will be asked to answer a question with 1 or 2 sentences. 2) Multiple-choices: The second task involves multiple-choice questions, where the model must choose the correct answer from either 4 options or True/False statements. When the Gopher model is scaled up to its largest size, its performance jumps to more than 20% above random, indicating the emergence of this ability.
3. **The Massive Multi-task Language Understanding (MMLU)** is another key benchmark. The primary objective of this benchmark is to evaluate models for their ability to demonstrate a broad range of **world knowledge and problem-solving** skills. The test encompasses 57 tasks, spanning areas such as elementary mathematics, US history, computer science, law, and more. GPTs, Gopher, and Chinchilla models of a specific scale do not perform better than guessing on average of all the topics, but scaling up to a larger size enables performance to surpass random, indicating the emergence of this ability.
4. Finally, the **Word in Context (WiC)** is a **semantic understanding** benchmark. WiC is a binary classification task for context-sensitive word embeddings. It involves target words (verbs or nouns) with two provided contexts, aiming to determine if they share the same meaning. Chinchilla fails to achieve the one-shot performance of better than random, even when scaled to its largest model size. Above-random performance eventually emerged when PaLM was scaled to a much larger size, suggesting the emergence of this ability at a larger scale.

Other Factors That Could Give Rise To Emergent Abilities

- Multi-step reasoning is a strategy where a model is guided to produce a sequence of intermediate steps before giving the final answer. This strategy, known as **chain-of-thought prompting**, only surpasses standard prompting when applied to a sufficiently large model.
- **Instruction following** is another strategy that involves fine-tuning a model on a mixture of tasks phrased as instructions. This strategy only improves performance when applied to a model of a specific size.

Risks With Emergent Abilities

As we scale up language models, we also need to be aware of the emergent risks that come with it. These risks could be societal issues related to **truthfulness, bias, and toxicity**. These risks can be avoided by applying strategies, such as giving model prompts that encourage them to be "**helpful, harmless, and honest**."

The **WinoGender** benchmark, which measures gender bias in occupations, has shown that scaling can improve performance but also increase bias in ambiguous contexts. Larger models were found to be more likely to memorize training data, although deduplication methods can reduce this risk.
Emergent risks also include phenomena that might only exist in future language models or that have not yet been characterized in current models. These could include backdoor vulnerabilities or harmful content synthesis.

A Shift Towards General-Purpose Models

The emergence of abilities has led to sociological changes in how the community views and uses these models. Historically, NLP focused on task-specific models. Scaling models has led to an explosion in research on "general purpose" models that aim to perform a range of tasks not explicitly encoded in the training data.
This shift towards general-purpose models is evident when scaling enables a few-shot prompted general-purpose model to outperform prior state-of-the-art held by fine-tuned task-specific models. For example, GPT-3 achieved a new state-of-the-art on the TriviaQA and PiQA question-answering benchmarks; PaLM achieved a new state-of-the-art on three arithmetic reasoning benchmarks; and the **multimodal Flamingo** model achieved a new state of the art on six visual question answering benchmarks.
The ability of general-purpose models to perform unseen tasks, given only a few examples, has also led to many new applications of language models outside the NLP research community. For instance, language models have been used by prompting to translate natural language instructions into actions that are executable by robots, interact with users, and facilitate multi-modal reasoning.

Conclusion

Emergent abilities in LLMs are capabilities that appear as the models scale up and are a key factor in their success. These abilities, unpredictable from smaller models, become evident after reaching a certain scale threshold. They have been observed in various contexts, such as in a few-shot prompting and augmented prompting strategies. Scaling up LLMs also introduces emergent risks like increased bias and toxicity, which can be avoided with appropriate strategies. The emergence of these abilities has led to a shift towards general-purpose models and opened up new applications outside the traditional NLP research community.
In the next lesson, we'll dive into today's most popular proprietary LLMs and describe the tradeoffs between proprietary and open-source LLMs.

The Most Popular Proprietary LLMs

Introduction

In this lesson, we'll dive into the most popular proprietary LLMs, such as GPT-4, Claude, and Cohere LLMs. The debate between open-source or proprietary models is multifaceted when discussing aspects like customizations, development speed, control, regulation, and quality.

Proprietary LLMs

Proprietary models like GPT-4 and PaLM are developed and controlled by specific organizations, in contrast to open-source LLMs such as BigScience's Bloom and various community-driven projects, which are freely available for developers to use, modify, and distribute. As a result, these

models may offer advanced features and customization options tailored to specific use cases. Organizations can fine-tune these models to meet their exact requirements, providing a competitive edge in the market.
Since proprietary models are developed and controlled by specific organizations, they have complete control over the model's development, deployment, and updates. This level of control allows organizations to protect their intellectual property and maintain a competitive advantage.
Organizations offering proprietary models often provide commercial support and service level agreements (SLAs) to ensure reliability and performance guarantees. This level of professional support can be crucial for some use cases.
Using **proprietary LLMs** can be cost-effective in most use cases. LLMs are large and need several GPUs to operate at low latency (and the right competencies); therefore, they benefit from economies of scale.
Let’s see now a list of popular proprietary models (as of July 2023).

Cohere LLMs

Cohere’s models are categorized into two main types: **Generative and Embeddings**. The Generative models, also known as **command** models, are trained on a large corpus of data from the Internet. They are continually developed and updated, with improvements released weekly.
You can register for a Cohere account and get a free trial API key. There is no credit or time limit associated with a trial key; calls are rate-limited to 100 calls per minute, which is typically enough for an experimental project.
Save your key in your `.env` file as follows.

```
COHERE_API_KEY="<YOUR-COHERE-API-KEY>"
```

Then, install the `cohere` package with this command.

```
pip install cohere
```

You can now generate text with Cohere as follows.

```
from dotenv import load_dotenv

load_dotenv()

import cohere

import os

co = cohere.Client(os.environ["COHERE_API_KEY"])

response = co.generate(

    prompt='Please briefly explain to me how Deep Learning works using at most 100 words.',

    max_tokens=200)

print(response.generations[0].text)
```

Code to execute

```
Deep Learning is a subfield of artificial intelligence and machine learning that is
based on artificial neural networks with many layers, inspired by the structure and
function of the human brain. These networks are trained on large amounts of data and
algorithms to identify patterns and learn from experience, enabling them to perform
complex tasks such as image and speech recognition, language translation, and
decision-making. The key components of Deep Learning are neural networks with many
layers, large amounts of data, and algorithms for training and optimization. Some of
the applications of Deep Learning include autonomous vehicles, natural language
processing, and speech recognition.
```

The output

On the other hand, the embedding models are **multilingual** and can support over 109 languages. These models are designed for large enterprises whose end users are spread worldwide. Developers can also build classifiers on top of Cohere's language models to automate language-based tasks and workflows. The Cohere service provides a variety of models such as ***Command*** (`command`) for dialogue-like interactions, ***Generation*** *(*`base`) for generative tasks, ***Summarize*** *(*`summarize-xlarge`) for generating summaries, and more.

OpenAI's GPT-3.5

GPT-3.5 is a language model developed by OpenAI. Its turbo version, **GPT-3.5-turbo** (recommended by OpenAI over other variants), offers a more affordable option for generating human-like text through an API accessible via OpenAI endpoints. The model is optimized for **chat applications** while remaining powerful on other generative tasks and can process 96 languages. GPT-3.5-turbo comes in two variants: one with a **4k tokens** context length and the other with **16k** tokens.

The Azure Chat Solution Accelerator, powered by Azure Open AI Service, offers enterprises a robust platform to host OpenAI chat models with enhanced moderation and safety features. This solution enables organizations to establish a dedicated chat environment within their Azure Subscription, ensuring a secure and tailored user experience. One of the key advantages is its **privacy aspect**, as it's deployed within **your Azure tenancy**, allowing for complete isolation and control over your chat services.

In the first lesson, "What are Large Language Models," we saw how to use GPT-3.5-turbo via API, so refer to that lesson for a code snippet on how to use it with Python and get an API key. Additionally, we have recently introduced the "**LangChain & Vector Databases in Production**" free course, aimed at assisting you in getting the most out of LLMs and enhancing their functionality. The course encompasses fundamental topics such as initiating prompts and addressing hallucination, as well as delving into advanced areas like using LangChain to give memory to LLMs and developing agents for interaction with the real world.

OpenAI's GPT-4

OpenAI's **GPT-4** is a **multimodal** model with an undisclosed number of parameters or training procedures. It is the latest and most powerful model published by OpenAI, and the multi-modality enables the model to process both **text and image as input**. It can be accessed by submitting your early access request through the OpenAI platform (as of July 2023). The two variants of the model are `gpt-4` and `gpt-4-32k` with different context lengths, 8k and 32k tokens, respectively.

Anthropic’s Claude

Anthropic, an AI safety and research company, is a significant player in the AI landscape. It has secured substantial funding and partnered with Google for cloud computing access, mirroring OpenAI's trajectory in recent years.

Anthropic's flagship product, **Claude 2,** is an LLM with a **context size of 100k** tokens. Anthropic has ambitious growth plans and aims to compete with top players like OpenAI and Deepmind, working with similarly advanced models.

Claude 2 is trained to be a helpful assistant in a conversational tone, similar to ChatGPT. Its beta, unfortunately, is open only to people in the US or UK (as of July 2023).

If you're in the US or UK, you can sign up for free on Anthropic's website. Just click "Talk to Claude," and you'll be prompted to provide an email address. You'll be ready to go after you confirm the email address.

The API is made available via the web Console. First, read here, Getting Access to Claude, for how to apply for access.

Once you have access to the Console, you can generate API keys via your Account Settings.

Google’s PaLM

Google's Pathways Language Model, or PaLM, is a next-generation artificial intelligence model optimized for various developer use cases, particularly in the realm of NLP. Its primary applications include the development of chatbots, text summarization, question-answering systems, and document search through its text embedding service.

PaLM 2, the upgraded version of the model, is renowned for its ease of use and precision in following instructions. It features variants that are specifically trained for **text and chat generation**, as well as **text embeddings**, allowing for a broad range of use cases.

Access to PaLM is exclusively through the PaLM API. Read the Setup process, and please note that as of July 2023, the PaLM API is only available after being selected from a waitlist.

Google's PaLM 2 showcases significant advancements, including **multilingual training** for superior foreign language performance, enhanced logical reasoning, and the ability to generate and debug code. It integrates seamlessly into services like Gmail. PaLM 2 can be fine-tuned for specific domains such as cybersecurity vulnerability detection (**Sec-PaLM**) and medical query responses (**Med-PaLM**).

Conclusion

The choice between proprietary and open-source AI models depends on the specific needs and resources of the user or organization, and the decision should be based on a careful evaluation of all factors.

The next lesson will cover the most popular open-source LLMs.

Open-Source LLMs

Introduction

In this lesson, we will discuss several open-source LLMs and their features, capabilities, and licenses. This overview will cover LLaMA 2, Open Assistant, Dolly (by Databricks), and Falcon as the most used LLMs. We will also explore the licenses and potential commercial usage of these models. Additionally, we will discuss limitations or restrictions that may be present in their licenses.

LLaMA 2

LLaMA 2 is a cutting-edge large language model developed by Meta, released on July 18, 2023, with an open license for both research and commercial use.

The architecture of LLaMA 2 is described in great detail in the 77-page paper, making it easier for data scientists **to recreate and fine-tune** the models for their specific needs. The model's training data comprises an impressive **2 trillion tokens**. It has been trained on a massive scale, outperforming all open-source benchmarks and demonstrating performance comparable to GPT3.5 in terms of human evaluation.

LLaMA 2 is available in three parameter variations: **7B, 13B, and 70B**, and there are also instruction-tuned versions known as **LLaMA-Chat**.

The fine-tuning process is done through **Supervised Fine-Tuning (SFT**) and Reinforcement Learning with Human Feedback (**RLHF**), using a novel approach to segment data based on helpfulness and safety prompts.

The reward models are crucial to LLaMA 2's performance, allowing it to balance safety and helpfulness effectively. The safety reward model and helpfulness reward model are trained to evaluate the quality of generated responses.

The impact of LLaMA 2 in Generative AI is substantial, outperforming other open innovation models like Falcon or Vicuna.

You can find the LLaMA 2 models on the Hugging Face Hub here. Here, we test the `meta-llama/Llama-2-7b-chat-hf` model. For this, you'll first have to request access to the model on this page.

First, let's download the model. It takes some time as the model weighs about 14GB.

```
from transformers import AutoModelForCausalLM, AutoTokenizer

import torch

# download model

model_id = "meta-llama/Llama-2-7b-chat-hf"

tokenizer = AutoTokenizer.from_pretrained(model_id)

model = AutoModelForCausalLM.from_pretrained(

        model_id,

        trust_remote_code=True,

        torch_dtype=torch.bfloat16)
```

Then, we generate a completion with it. This step will take a lot of time if you're generating text using CPUs instead of GPUs!

```
# generate answer

prompt = "Translate English to French: Configuration files are easy to use!"

inputs = tokenizer(prompt, return_tensors="pt", return_token_type_ids=False)

outputs = model.generate(**inputs, max_new_tokens=100)

# print answer

print(tokenizer.batch_decode(outputs, skip_special_tokens=True)[0])Copy
```

Falcon

The Falcon models, developed and trained by the Technology Innovation Institute (TII) of Abu Dhabi, have gained significant attention since their release in May 2023. These models are causal large language models (LLM), similar to GPT, and are also known as "**decoder-only**" models. They excel in predicting the next token in a sequence of tokens with their attention focused solely on the left context during training, while the right context remains masked.
The Falcon models are distributed under the Apache 2.0 License, allowing even commercial use. The largest of these models**, Falcon-40B**, has shown great performance, outperforming other causal LLMs like LLaMa-65B and MPT-7B. **Falcon-7B**, a slightly smaller version, was designed to be fine-tuned on consumer hardware and has half the number of layers and embedding dimensions compared to Falcon-40B.
The training data for Falcon models primarily comes from the **"Falcon RefinedWeb dataset**," which is meticulously curated and multimodal-friendly, preserving links and alt texts of images. This dataset and curated corpora make up 75% of the pre-training data for the Falcon models. While it primarily covers English, additional versions like **"RefinedWeb-Europe**" have been prepared to include several European languages.
The instruct versions of Falcon-40B and Falcon-7B perform even better, with fine-tuning done on a mixture of chat/instruct datasets sourced from various places, including GPT4all and GPTeacher.
You can find the Falcon models on the Hugging Face Hub here. Here, we test the `tiiuae/falcon-7b-instruct` model. You can use the same code previously used for the LLaMA example by changing the `model_id`.

```
model_id = "tiiuae/falcon-7b-instruct"
```

Dolly

Dolly is an open-source LLM introduced by Databricks. It was first unveiled as Dolly 1.0, a language model that showcased ChatGPT-like human interactivity. The team has now released **Dolly 2.0**, a better instruction-following LLM.
One of the critical features of Dolly 2.0 is that it is built on a new, high-quality human-generated instruction **dataset** called "**databricks-dolly-15k**". This dataset consists of 15,000 prompt/response pairs designed explicitly for instruction tuning large language models. Unlike many instruction-following models, Dolly 2.0's dataset is open-source and licensed under the Creative Commons Attribution-ShareAlike 3.0 Unported License. This means that anyone can use, modify, or extend the dataset for any purpose, including commercial applications.
The Dolly 2.0 model is based on the **EleutherAI Pythia-12 b** architecture, comprising 12 billion parameters, which makes it capable of high-quality instruction following behavior. Despite being smaller than some other models, such as Alpaca, Dolly 2.0 has demonstrated great performance due to its reliance on real-world, human-generated training records rather than synthesized data.
You can find the Databricks models on the Hugging Face Hub here. Here, we test the `databricks/dolly-v2-3b` model. You can use the same code previously used for the LLaMA example by changing the `model_id`.

```
model_id = "databricks/dolly-v2-3b"
```

Open Assistant

The Open Assistant project is an initiative aiming to make high-quality large language models accessible to everyone through an open-source and collaborative approach. Unlike some other ChatGPT open-source alternatives with restricted licenses, Open Assistant seeks to provide a

versatile chat-based language model comparable to ChatGPT and GPT-4 that can be used for commercial purposes.
The heart of the project lies in its commitment to openness and inclusivity. They have collected a substantial dataset from over 13,000 volunteers, comprising more than 600,000 interactions, 150,000 messages, and 10,000 fully annotated conversation trees on various topics and in multiple languages. This dataset serves as the foundation for training various models hosted on platforms like Hugging Face.
Users can explore the potential of Open Assistant by interacting with the model through the Hugging Face demo or the official chat interface, both designed to solicit user feedback to help improve the chatbot's responses. The project encourages community involvement and contributions, allowing users to participate in data collection and ranking tasks to enhance the capabilities of the language model.
As with most open-source large language models, Open Assistant does have some **limitations**, particularly in answering **math and coding** questions, as they are trained on fewer interactions in these domains. However, the model is generally adept at generating interesting and human-like responses, though occasional inaccuracies may occur.

Mistral

Mistral, in September 2023, has released their language model Mistral 7B under the Apache 2.0 license. This model, with **7.3 billion** parameters, has shown superior performance compared to the Llama 2 13B and Llama 1 34B models on all and many benchmarks respectively. It also approaches the performance of CodeLlama 7B on code while maintaining proficiency in English tasks.
Mistral 7B uses **Grouped-query attention (GQA)** for faster inference and **Sliding Window Attention (SWA)** to handle longer sequences more cost-effectively. This, along with modifications to **FlashAttention and xFormers**, has led to a **2x speed** improvement for sequence lengths of 16k with a window of 4k.
The model can be downloaded and used anywhere, including locally, with the team's reference implementation. It can also be deployed on any cloud (AWS/GCP/Azure) using the vLLM inference server, or used on HuggingFace.
Mistral 7B is easily fine-tuned for any task. As a demonstration, the team has provided a model fine-tuned for chat, which outperforms the Llama 2 13B chat model. The fine-tuned model, Mistral 7B Instruct, outperforms all 7B models on MT-Bench and is comparable to 13B chat models.

The Hugging Face Open LLM Leaderboard

Hugging Face hosts an **LLM leaderboard**. This leaderboard is created by evaluating community-submitted models on text generation benchmarks on Hugging Face’s clusters. It’s an excellent resource for checking the new best-performant open-source LLMs.
If you can’t find the language or domain you’re looking for, you can filter them out and find the one that meets your specific requirements.

Conclusion

In this lesson, we explored several open-source LLMs) and their features, capabilities, and licenses. We discussed LLaMA 2, Falcon, Dolly, and Open Assistant as some of the most prominent open-source LLMs available.

- **LLaMA 2**, developed by Meta, is a cutting-edge language model with impressive performance and is available in various parameter variations. It has been trained on a massive scale and demonstrates remarkable performance comparable to GPT3.5.

- Falcon models, developed and trained by the Technology Innovation Institute (TII) of Abu Dhabi, have gained attention for their decoder-only approach and have shown great performance, especially the **Falcon-40B** model.
- **Dolly**, introduced by Databricks, is an open-source LLM with a focus on instruction following. It has a high-quality human-generated instruction dataset and is licensed under Creative Commons, allowing for versatile use, including commercial applications.
- **Open Assistant** is an ambitious project aiming to make high-quality LLMs accessible to everyone through openness and inclusivity. It encourages community involvement and contributions to enhance the capabilities of the language model.

It is essential to acknowledge the importance of open-source LLMs in advancing the field of natural language processing and enabling wider access to state-of-the-art language models for research and commercial purposes.

In the next lesson, we will explore an equally important aspect of LLMs - **hallucinations and bias**. Hallucinations refer to the generation of fake or incorrect information by LLMs, while bias entails the perpetuation of prejudiced or discriminatory content. Understanding and addressing these challenges are crucial to ensuring the responsible and ethical use of large language models in various applications.

How to reduce Hallucination & Biases in LLMs

1. Trustworthy Data source
2. Tweak the inference parameter –
    - Temperature
    - Top-p
    - Frequency-penalty
    - Presence-penalty
    - Top-k
3. Prompt engineering
4. RAG & Deep Memory
5. Fine tuning of LLM & Embedding model
6. RLHF (Reinforcement Learning Human Feedback) / RLAIF (Reinforcement Learning AI Feedback)

Understanding Hallucinations and Bias

Introduction

In this lesson, we'll cover the concept of **hallucinations** in LLMs, highlighting their influence on AI applications and demonstrating how to mitigate them using techniques like the retriever's architectures. We'll also explore **bias** within LLMs with examples.

Hallucinations in LLMs

In Large Language Models, hallucinations refer to cases where the model produces text that's **incorrect and not based on reality**. An AI hallucination is a confident response by the model that cannot be grounded in any of its training data.

There are several possible reasons for these types of hallucinations:

- An LLM could be trained on a dataset that doesn't have the knowledge required to answer a question.
- An LLM does not have a reliable way to check the factual accuracy of its responses. It just outputs a response that may be wholly incorrect yet convincing.
- The training dataset used to train the LLM may include fictional content and subjective content, like opinions and beliefs.
- LLMs are not generally optimized to say "I don't know" when they don't have enough information. Therefore, when the LLM has no answer, it generates whatever is the most probable sequence of text as a response. But note that "most probable" does not necessarily mean "true." Hallucination is the result of this misalignment. Note that the goal of language modeling is learning the underlying distribution of the words in a language, not to distinguish what is true and what isn't.

Let's see a clear example of a hallucination.

On April 12th, 2023, Databricks announced their "Dolly 2.0" model, an open-source 12B parameter language model fine-tuned on a human-generated instruction-following dataset. GPT-3.5 has been trained on a dataset with a cutoff year of 2021. Therefore, it can't know about Dolly 2.

When GPT-3.5 is asked: "Does Databricks Dolly provide value to the open source community?" it responds with something like this:

GPT 3.5: *Databricks Dolly is a tool developed by Databricks, which is designed to help users migrate their data from one platform to another, particularly from on-premise data centers to cloud-based platforms such as AWS or Azure. While Databricks Dolly is not an open source tool, it can still provide value to the open source community in a few ways.[…]*

Except for "Databricks Dolly is a tool developed by DataBricks," the rest of the text is entirely incorrect! Dolly is not a tool to migrate data; it is open-source, contrary to the response. Still, GPT-3.5 knows about Databricks as a company, and therefore, it generates a plausible description of Dolly 2 as a generic product of Databricks.

While this example shows OpenAI GPT-3.5 hallucinating, it's important to note that this phenomenon applies to other similar LLMs like Bard or LLama.

Strategies to mitigate hallucinations include tuning the text generation parameters, cleaning up the training data, precisely defining prompts (prompt engineering), and using retriever architectures to ground responses in specific retrieved documents.

Misinformation Spreading

One significant risk associated with hallucinations in LLMs is their potential to generate content that, while appearing credible, is factually incorrect. Due to their limited capacity to understand the context and verify facts, LLMs can unintentionally spread misinformation.

There's the potential for individuals with malicious intent to exploit LLMs to spread disinformation deliberately, creating and promoting false narratives. A study by Blackberry found that nearly half of the respondents (49%) believed that GPT-4 could be used to spread misinformation. The unrestricted spread of such false information via LLMs can lead to widespread negative impacts across societal, cultural, economic, and political landscapes. It's crucial to **address these issues** related to LLM hallucinations to ensure the ethical use of these models.

Tuning the Text Generation Parameters

The generated output of LLMs is greatly influenced by various model parameters, including **temperature, frequency penalty, presence penalty, and top-p**. We'll learn more about them in a later lesson in the course.

Higher temperature values promote randomness and creativity, while lower values make the output more deterministic. Increasing the frequency penalty value encourages the model to use repeated tokens more conservatively. Similarly, a higher presence penalty value increases the likelihood of generating tokens not yet included in the generated text. The "top-p" parameter controls response diversity by setting a cumulative probability threshold for word selection.

Leveraging External Documents with Retrievers Architectures

Response accuracy can be improved by providing **domain-specific knowledge** to the LLM in the form of external documents. Augmenting the **knowledge base** with domain-specific information allows the model to ground its responses in the knowledge base. After a question from a user, we could retrieve documents relevant to the questions (leveraging a module called "**retriever**") and use them in a prompt to produce the answer. This type of process is implemented into architectures typically called "**retrievers architectures**".

In these architectures:

1. When a user poses a question, the system computes an embedding representation of it.
2. The embedding of the question is then used for executing a **semantic search** in the database of documents (by comparing their embeddings and computing similarity scores).
3. The top-ranked documents are used by the LLM as context to give the final answer. Usually, the LLM is asked to extract the answer from those context passages precisely and not to write anything that can't be inferred from them.

Retrieval-augmented generation (RAG) is a technique that enhances language model capabilities by sourcing data from external resources and integrating it with the context provided in the model's prompt.

Providing access to external data sources during the prediction process enriches the model's knowledge and grounding. By leveraging external knowledge, the model can generate more accurate, contextually appropriate responses and be less prone to hallucination.

Bias in LLMs

Large language models like GPT-3.5 and GPT-4 have raised serious privacy and ethical concerns. Research has shown that these models are prone to inherent bias, leading to the generation of prejudiced or hateful language, intensifying the concerns regarding their use and governance.

Biases in LLMs arise from various sources: the data, the annotation process, the input representations, the models, and the research design.

For instance, training data that don't represent the diversity of language can lead to demographic biases, resulting in a model's inability to understand and accurately represent certain user groups. Misrepresentation can vary from mild inconveniences to more covert, gradual declines in performance, which can unfairly impact certain demographic groups.

LLMs can unintentionally intensify harmful biases through their hallucinations, creating prejudiced and offensive content.
The data used to train LLMs frequently includes stereotypes, which the models may unknowingly reinforce. This imbalance can lead the models to generate prejudiced content that discriminates against underrepresented groups, potentially targeting them based on factors like race, gender, religion, and ethnicity.
This can be exemplified when an LLM produces content that presents women as inferior or portrays certain ethnicities as intrinsically violent or unreliable. Also, if a model is trained on data biased towards a younger, technologically savvy demographic, it may generate outputs that overlook older individuals or those from less technologically equipped regions. If the model is steeped in data from sources promoting hate speech or toxic content, it might produce damaging and prejudiced outputs, amplifying the diffusion of harmful stereotypes and biases.
These examples underscore the urgent need for constant monitoring and ethical management in the use of these models.

Constitutional AI

Constitutional AI' is a conceptual framework crafted by researchers at **Anthropic**. It aims to align AI systems with human values, ensuring that they become beneficial, safe, and trustworthy.
In the beginning, the model is trained to self-review and modify its responses based on a set of predetermined principles and a small set of process examples. The next phase involves reinforcement learning training. At this point, the model leans on AI-generated feedback, grounded in the given principles, as opposed to human feedback, to choose the least harmful response.
Constitutional AI employs methodologies like **self-supervision training**. These techniques allow the AI to learn to conform to its constitution, without the need for explicit human labeling or supervision.

The approach also includes developing constrained optimization techniques. These ensure that the AI pursues helpfulness within the boundaries set by its constitution rather than pursuing unbounded optimization, potentially forgetting helpful knowledge.

Conclusion

The risks of hallucinations and biases in LLMs present significant issues in producing reliable and accurate outputs. The presence of biases can further damage the accuracy and fairness of the outputs, resulting in the ongoing progression of harmful stereotypes and misinformation.
It's imperative to formulate strategies to mitigate these risks. Such strategies should incorporate pre-processing and input control measures, model configuration adjustments, improvement mechanisms, and context and knowledge enhancement techniques.
Integrating the ethical guidelines is essential to ensure that the models generate fair and trustworthy outputs, ultimately achieving responsible use of these powerful technologies.

Applications and Use-Cases of LLMs

Introduction

In this lesson, we will explore the diverse applications and use cases of LLMs and generative AI across various industries.

We dive into how LLMs are revolutionizing healthcare and medical research by improving diagnosis, drug discovery, and patient care. Additionally, we will uncover their impact on finance, copywriting, education, programming, and the legal industry.

While LLMs offer immense potential, we will also address the risks and ethical considerations associated with their deployment in real-world scenarios, emphasizing the importance of responsible AI implementation and human oversight.

Healthcare and Medical Research

Generative AI offers promising applications that can enhance patient care, drug discovery, and operational efficiency in the industry.

Generative AI is being utilized for diagnosis, patient monitoring, and resource optimization. By incorporating large language models into digital pathology, accuracy for detecting diseases such as cancer has improved significantly. Furthermore, the technology aids in automating administrative tasks, which streamlines workflows and allows clinical staff to focus on more critical aspects of patient care.

In the pharmaceutical industry, generative AI has become a game-changer in **drug discovery**. It accelerates the process and improves precision in medicine therapies, leading to shorter drug development timelines and reduced costs. This advancement paves the way for more personalized treatments and targeted therapies, ultimately benefiting patients.

Medtech companies are exploring the potential of generative AI to create personalized devices for patient-centered care. Integrating generative AI into the design process optimizes medical devices for specific patient needs, improving treatment outcomes and increasing patient satisfaction.

For example**, Med-PaLM** is a large language model designed by Google to provide high quality answers to medical questions. It functions as a multimodal generative model capable of processing diverse biomedical data such as clinical text, medical images, and genomics, all using the same set of model parameters. Another example is **BioMedLM**, a domain-specific LLM for biomedical text, made by the Stanford Center for Research on Foundation Models (CRFM) and MosaicML.

Finance

LLMs like GPT have proven to be powerful tools for analyzing and processing financial data, revolutionizing how financial institutions interact with their clients and manage risks.

One of the key applications of LLMs in finance is in customer interactions with digital platforms, where models can be utilized to enhance user experience through **chatbots** or AI-based apps. These applications enable seamless and efficient customer support, providing real-time responses to queries and concerns.

The analysis of **financial time-series data** is another area where LLMs and generative AI have proven worthy. By leveraging large datasets of stock exchange information, these models can offer valuable insights for macroeconomic analysis and stock exchange prediction. Predicting market trends and identifying potential investment opportunities are crucial for making informed financial decisions. LLMs play a significant role in this aspect.

For example, Bloomberg trained an LLM on a mix of general purpose and domain specific documents, calling it **BloombergGPT**. BloombergGPT outperforms similarly-sized open models on financial NLP tasks, without sacrificing performance on general LLM benchmarks.

Copywriting

Large Language Models and generative AI are influencing the field of copywriting by providing powerful tools for creating content.

The applications of generative AI in copywriting are diverse. It can be utilized to **speed up** the **writing process**, **overcome writer's block**, and **reduce costs** by improving overall productivity. Additionally, generative AI helps maintain a consistent brand image by learning a company's language patterns and style, ensuring cohesive marketing activities.

Some prominent use cases include generating website content and blog posts, crafting social media posts, creating product descriptions, and optimizing content for SEO. Generative AI can also contribute to developing content for mobile apps, tailoring it to suit different platforms and user experiences.

A popular copywriting tool that uses LLMs is **Jasper**, which makes it easy to generate diverse kinds of content using generative AI.

Education

LLMs can help a lot in **online learning** and **personalized tutoring**. By analyzing individual learning progress, LLMs offer personal feedback, adaptive testing, and personalized interventions.

These models can address the challenges of teacher shortages by providing scalable solutions such as virtual teachers or supporting para-teachers with advanced tools. They empower educators to become mentors and guides, offering personalized support and interactive learning experiences.

AI can analyze individual student performance and personalize the learning experience.

For example, an application of LLMs for education is **Khanmigo** of Khan Academy. LLMs serve as virtual tutors, offering explanations and examples for better subject understanding. LLMs aid language learning, generating sentences for grammar and vocabulary practice.

Programming

LLMs and generative AI can significantly help in coding by providing powerful tools for developers.

LLMs like GPT-4 and its predecessors can **generate code snippets** based on natural language prompts, significantly enhancing programmers' efficiency. These models are trained on vast corpora of code samples and can understand context, enabling them to generate more relevant and accurate code over time.

The applications of LLMs for coding are diverse. They can assist in code completion by suggesting code snippets as developers type, saving time and **reducing errors**. Additionally, LLMs are employed for **unit test** generation, automating the creation of **test cases**. This not only enhances code quality but also assists in software maintenance.

However, the use of generative AI in coding also presents challenges. While it can boost productivity, developers must exercise caution and review the generated code, as it may contain errors or security vulnerabilities. Furthermore, the potential for model biases and "hallucinations" (fabricating incorrect information) necessitates careful scrutiny.

A popular product using LLMs for programming is **GitHub Copilot**, which is trained on billions of lines of code. Copilot can turn natural language prompts into coding suggestions across dozens of languages.

Legal Industry

LLMs and generative AI have emerged as powerful tools for the legal industry, offering a range of applications and use cases. These models can be designed to handle the complexities of legal language, interpretations, and the dynamic nature of law. LLMs have the potential to assist legal practitioners in various tasks, such as providing **legal advice**, understanding complex **legal documents**, and analyzing **court case texts**.

One key application is reducing hallucinations, a common challenge with early legal LLMs. These models can produce more accurate and reliable results by integrating domain-specific knowledge through reference modules and reliable data from knowledge bases. They can also identify legal feature words within users' input and quickly analyze legal situations.

Risks and Ethical Considerations of Using LLMs in the Real World

As we learned in previous lessons, deploying Large Language Models (LLMs) in real-world applications poses certain risks and ethical concerns.

One significant risk is "**hallucinations**," where the LLM generates false but plausible-sounding information. This could lead to serious consequences, particularly in critical domains like healthcare, finance, and law.

Another concern is "**bias,**" as LLMs can inadvertently perpetuate societal biases present in their training data. This could result in unfair treatment in areas such as healthcare and finance. Addressing bias requires rigorous data evaluation, inclusivity efforts, and continuous improvement in fairness.

Data privacy and security are crucial as LLMs might memorize sensitive information, potentially leading to privacy breaches. Organizations must implement measures like data anonymization and strict access controls.

Additionally, the impact on employment should be considered, balancing automation and human involvement to preserve human expertise. Dependence on LLMs without human judgment can be risky, necessitating a responsible approach that combines AI benefits with human oversight.

Conclusion

This lesson explored the wide-ranging applications and use cases of LLMs and generative AI across diverse industries. From healthcare and medical research to finance, copywriting, education, programming, and the legal industry, LLMs are powerful tools with immense potential. However, alongside their benefits, we must be mindful of the risks and ethical considerations associated with their deployment. Addressing issues like hallucinations, bias, data privacy, and the impact on human employment is crucial for responsible AI implementation.

In the next module, we will study Transformer architectures, the backbone of LLMs, to gain insights into their working principles and understand how these models process and generate language with great accuracy and efficiency.