Introduction and Fundamentals

Module 1.2: Understanding Large Language Models (LLMs)

Objectives:

- Define and identify the capabilities of Large Language Models (LLMs).
- Being able to differentiate between LLMs and traditional Machine Learning (ML) / Natural Language Processing (NLP) models.
- Recognize the historical evolution and technological advancements that have led to the rise of LLMs.
- Identify the applications of LLMs across various industries, including chatbots, translation, code generation, and sentiment analysis.
- Gain an understanding about popular LLMs: GPT-3, GPT-4, BERT, etc.
- Compare proprietary, open, and open-source LLMs, and understand their respective use cases and limitations.

Introduction

Building on our understanding of AI transformers from Module 1.1, we now delve into large language models (LLMs), which are advanced implementations of transformer architectures.

An LLM is a type of AI designed to process and generate human language at an unprecedented scale. These models are trained on vast datasets using deep learning architectures, such as transformers, to learn the patterns, structures, and nuances of human language. This enables them to generate natural language responses to a wide range of user inputs, such as answering questions, generating text, writing code, and more. LLMs have become critical in fields like NLP, machine translation, code generation, and other AI-driven applications.

LLMs typically consist of billions of parameters and are trained on diverse datasets, allowing them to predict and generate coherent text based on the input they receive. This extensive training enables LLMs to engage in complex conversations, summarize large bodies of text, or even complete tasks like writing functional code snippets. Their ability to handle multiple tasks through fine-tuning makes them incredibly versatile.

While this module focuses on LLMs, they are part of a larger generative AI landscape, which includes other implementations such as art generation from text, audio generation, video synthesis, and more (as shown in **Figure 1**). These broader applications of generative AI are evolving rapidly and are expected to expand into new domains.

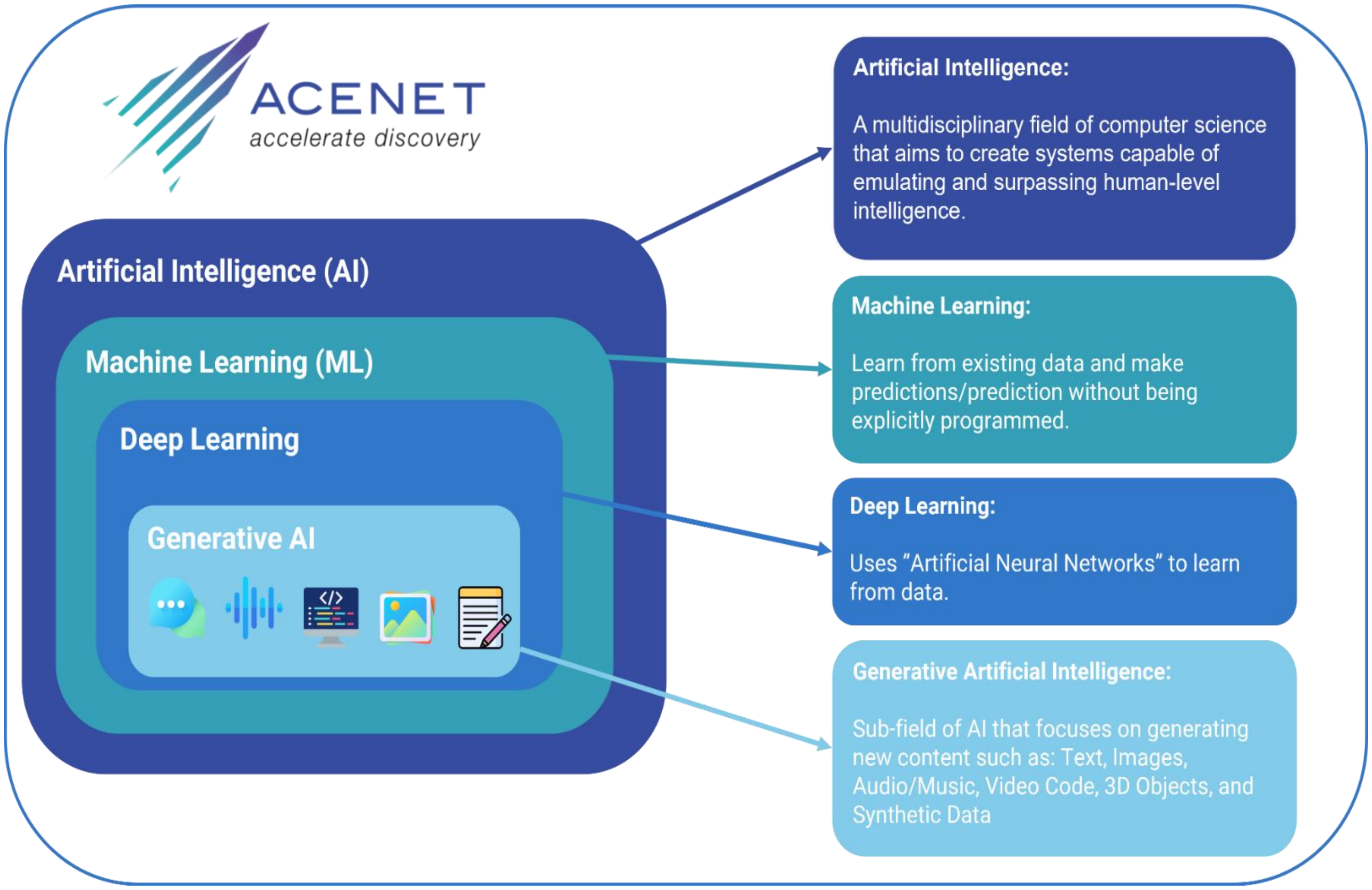
ACENET
accelerate discovery
Artificial Intelligence (AI)
Machine Learning (ML)
Deep Learning
Generative AI
Artificial Intelligence:
A multidisciplinary field of computer science that aims to create systems capable of emulating and surpassing human-level intelligence.
Machine Learning:
Learn from existing data and make predictions/prediction without being explicitly programmed.
Deep Learning:
Uses "Artificial Neural Networks" to learn from data.
Generative Artificial Intelligence:
Sub-field of AI that focuses on generating new content such as: Text, Images, Audio/Music, Video Code, 3D Objects, and Synthetic Data

Figure 1: Sub-domains of AI

1. LLMs Vs GenAI

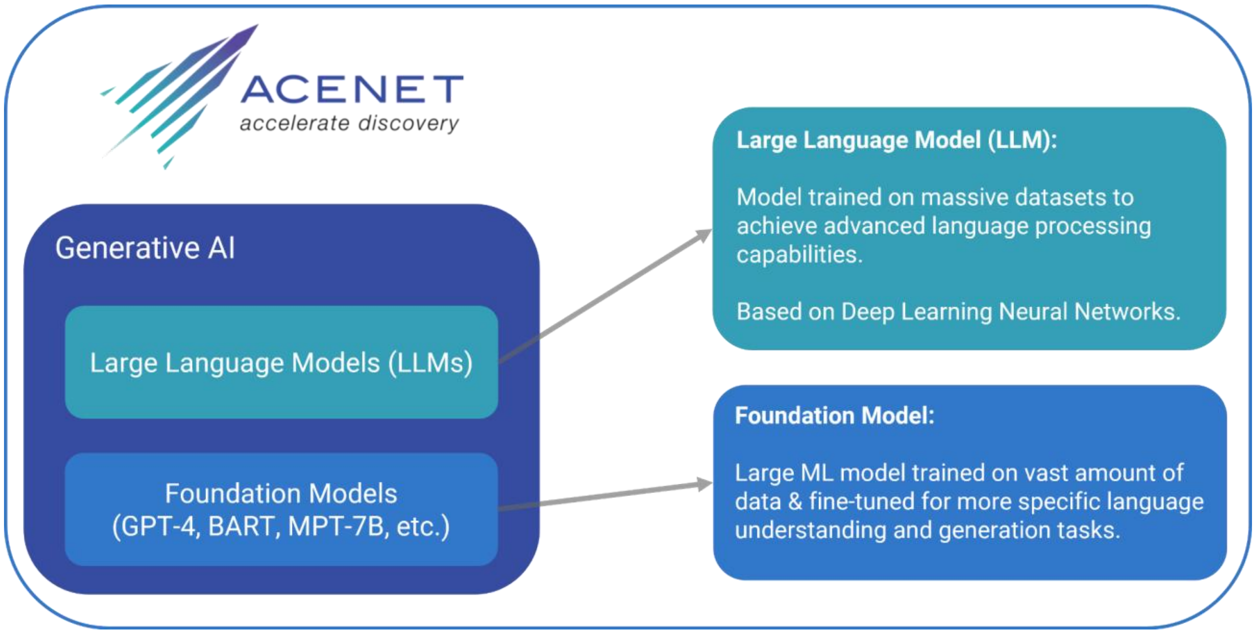
ACENET
accelerate discovery
Generative AI
Large Language Models (LLMs)
Foundation Models
(GPT-4, BART, MPT-7B, etc.)
Large Language Model (LLM):
Model trained on massive datasets to achieve advanced language processing capabilities.
Based on Deep Learning Neural Networks.
Foundation Model:
Large ML model trained on vast amount of data & fine-tuned for more specific language understanding and generation tasks.

Figure 2: LLMS Vs GenAI

The key difference between **LLMs** and **Generative AI (GenAI)** lies in their scope and application. While **LLMs** focus specifically on processing and generating human language, **GenAI** is a broader concept that encompasses any AI technology capable of generating content, which can include text, images, music, code, and more. In other words, LLMs are a subset of GenAI, specializing in language tasks, whereas GenAI covers a wider range of creative and generative outputs. LLMs like **GPT-4**, **BERT**, and **Llama** are foundational models within the GenAI space, specifically designed for tasks related to language understanding and generation, but they power various GenAI applications such as chatbots, code generation, and content creation **Figure 2.**

2. Historical Background and Development of LLMS

The development of LLMs has been shaped by decades of advancements in language processing and machine learning. Starting in the 1950s-1990s, early efforts relied on rule-based systems to map linguistic patterns, achieving limited success in structured tasks like translation. By the 1990s, language models began evolving into statistical frameworks, but their growth was constrained by computational power. The 2000s saw significant progress with the adoption of machine learning techniques and the explosion of training data, fueled by the rise of the internet. In 2012, breakthroughs in deep learning and larger datasets enabled the creation of models like GPT (Generative Pre-trained Transformer), marking a turning point in language modeling.

The introduction of BERT in 2018 revolutionized the field with its transformer-based architecture, paving the way for more powerful LLMs. OpenAI's GPT-3 in 2020 set new benchmarks with 175 billion parameters, making large-scale LLMs more impactful than ever. The public release of ChatGPT in 2022 brought generative AI to mainstream attention, demonstrating the potential of LLMs in everyday applications. By 2023, open-source models like LLaMA and Alpaca emerged, delivering high-quality results and expanding access to advanced AI capabilities. These milestones underscore the rapid evolution of LLMs and their growing importance in transforming industries (**Figure 3**).

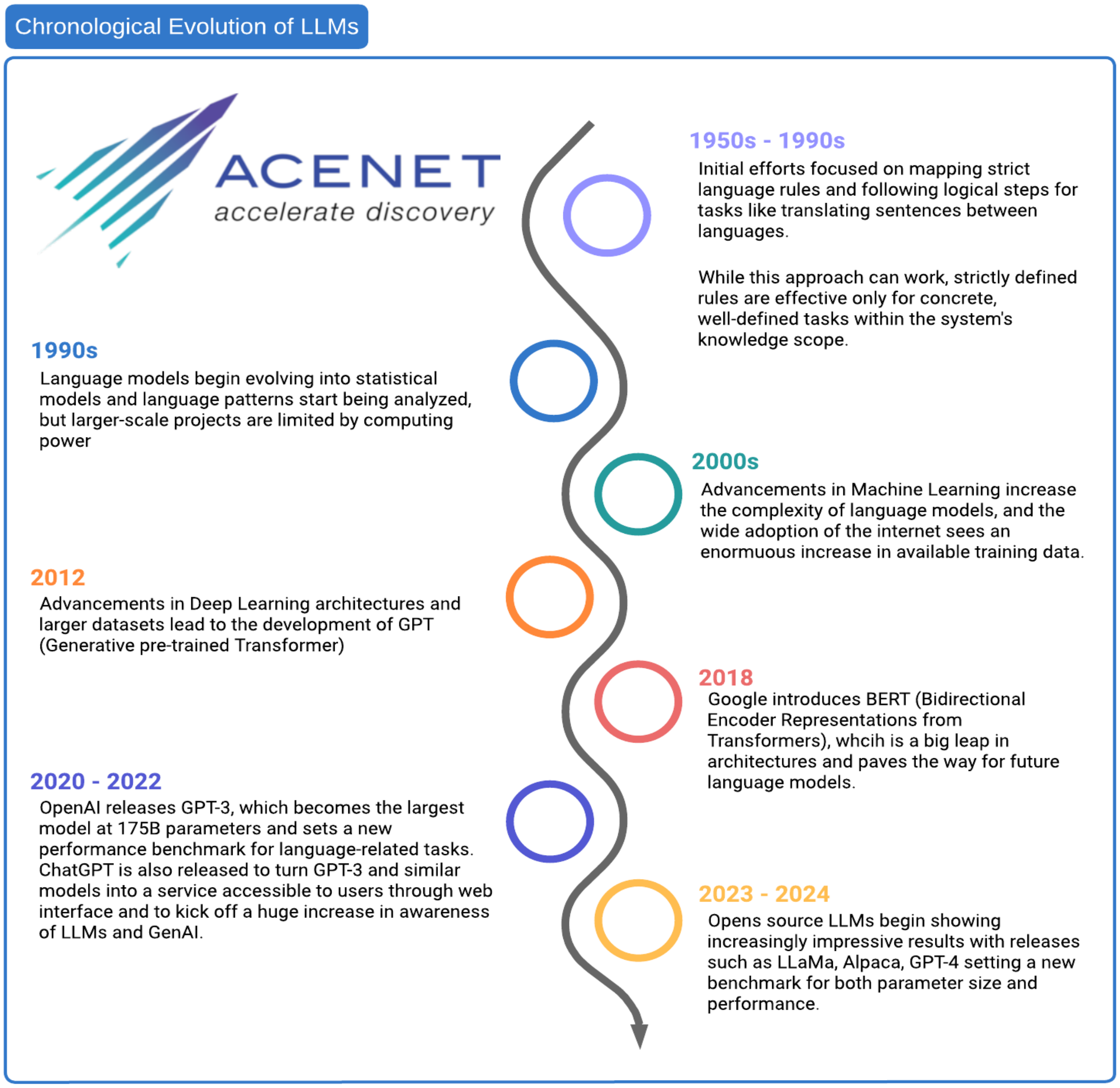
Chronological Evolution of LLMs
ACENET
accelerate discovery
1950s - 1990s
Initial efforts focused on mapping strict language rules and following logical steps for tasks like translating sentences between languages.
While this approach can work, strictly defined rules are effective only for concrete, well-defined tasks within the system's knowledge scope.
1990s
Language models begin evolving into statistical models and language patterns start being analyzed, but larger-scale projects are limited by computing power
2000s
Advancements in Machine Learning increase the complexity of language models, and the wide adoption of the internet sees an enormuous increase in available training data.
2012
Advancements in Deep Learning architectures and larger datasets lead to the development of GPT (Generative pre-trained Transformer)
2018
Google introduces BERT (Bidirectional Encoder Representations from Transformers), whcih is a big leap in architectures and paves the way for future language models.
2020 - 2022
OpenAI releases GPT-3, which becomes the largest model at 175B parameters and sets a new performance benchmark for language-related tasks. ChatGPT is also released to turn GPT-3 and similar models into a service accessible to users through web interface and to kick off a huge increase in awareness of LLMs and GenAI.
2023 - 2024
Opens source LLMs begin showing increasingly impressive results with releases such as LLaMa, Alpaca, GPT-4 setting a new benchmark for both parameter size and performance.

Figure 3: Chronological Evolution of LLMs

More details are explained in a different document called ([Chronological Evolution of LLMs](#)).

3. LLMs Vs Traditional Language Models

Traditional ML and NLP models are more task-specific, trained on smaller datasets and often require manual feature engineering and tuning. For example, models used for text classification, sentiment analysis, or named entity recognition typically need specific adjustments to perform well on each task. In contrast, LLMs leverage transformer-based architectures, as seen in the previous module, which allow them to learn from massive datasets without the need for explicit feature extraction. Explicit feature extraction refers to the manual process of identifying and engineering specific features from raw data that are relevant to a machine learning task. For example, in traditional NLP workflows, explicit feature extraction might involve creating features such as word counts, term frequency-inverse document frequency (TF-IDF) scores, part-of-speech tags, or syntactic dependencies. These handcrafted features rely heavily on human intuition and domain expertise, and they often fail to capture complex patterns and relationships present in the data.

LLMs bypass this requirement by automatically extracting and learning features directly from the data during training. Using attention mechanisms and transformer layers, LLMs can dynamically identify meaningful relationships between words, phrases, and contexts across vast amounts of unstructured text. This eliminates the need for predefined features and enables models to discover nuanced patterns, such as semantic similarities, contextual dependencies, and linguistic structures, that might be difficult or impossible for humans to define explicitly.

By removing the dependency on explicit feature extraction, transformer-based LLMs not only simplify the machine learning pipeline but also significantly improve performance on diverse NLP tasks. This capability enables LLMs to generalize across a wide array of tasks, from text generation to answering complex questions, without needing extensive retraining for each new task. Additionally, fine-tuning allows LLMs to be easily adapted for specialized tasks, whereas traditional models would need to be retrained from scratch to perform new tasks.

Some of the leading LLMs include **GPT**, **Llama**, **LaMDA**, **PaLM**, **BERT**, and **ERNIE**. These models power applications ranging from customer service chatbots to content generation and software development. While developing a custom LLM requires significant investment in resources, expertise, and infrastructure, pre-trained LLMs can be integrated into existing systems with relative ease, allowing organizations to leverage their power without the need to build models from the ground up.

3.1. Traditional Language Models Explained

Traditional language models are statistical models designed to predict the likelihood of a sequence of words in a given text. They work by estimating the probability distribution of word sequences, which allows them to generate or evaluate sentences based on how frequently certain word combinations appear in a corpus. These models rely on the assumption that the probability of a word depends on a fixed number of preceding words (its context), and they are foundational in early NLP tasks. Here are some examples of traditional language models and how they work:

3.1.1. n-Gram Models

An n-gram model is one of the simplest traditional language models. It predicts the next word in a sequence based on the preceding n-1 words. The idea is that the probability of a word depends on the context of the last few words, with n representing the number of words considered.

- **Unigram**: Only the current word is considered (no context).
- **Bigram**: The model considers one previous word.
- **Trigram**: The model considers two previous words, and so on.

Example

- In a bigram model, given the sentence **"The cat is"**, the model would predict the next word by looking at the probability of each word following "is" based on its training data.

The sentence's probability is calculated by multiplying the conditional probabilities of each word given the previous one:

```
P(The cat is cute) = P(The) x P(cat|The) x P(is|cat) x P(cute|is)
```

Interactive example 1: Generating and counting n-grams - Kaggle notebook

Interactive example 2: Next word prediction using n-grams - Kaggle notebook

Challenges:

- **Limited context:** n-gram models only capture local dependencies and cannot handle long-range dependencies between words.
- **Data sparsity**: If certain n-grams don't appear in the training data, the model can't make predictions for them, requiring smoothing techniques to deal with unseen word combinations.

3.1.2. Bag-of-Words Model (BoW)

The Bag-of-Words (BoW) model represents text as a collection of words, without considering the order in which they appear. It breaks down a text into individual words, and each word is assigned a count or weight, treating each word as an independent entity.

- **Representation**: Each document is represented as a vector where each entry corresponds to the presence or frequency of a word in that document.
- **Applications**: BoW is often used for document classification, where the task is to classify text based on word occurrences, without caring about the word sequence.

Example:

For the sentence **"The cat sleeps on the mat"**, the BoW model would treat it as a collection of individual words: {"The": 2, "cat": 1, "sleeps": 1, "on": 1, "mat": 1}. The word "The" occurs twice, while the other words occur once. The order of words is irrelevant.

Interactive example 3: Using BoW model for text classification - Kaggle notebook

Challenges:

- **Loss of context**: BoW models ignore word order and syntax, which can lead to poor performance in tasks where word sequence matters.
- **Large vocabulary size**: As the vocabulary grows, the dimensionality of the model increases, making it less efficient.

3.1.3. Term Frequency-Inverse Document Frequency (TF-IDF)

The TF-IDF algorithm is a statistical technique widely used in NLP to evaluate the importance of a word in a document relative to a collection of documents (corpus). Unlike BoW, which focuses on word counts, TF-IDF accounts for the frequency of a word in a document (Term Frequency, TF) and adjusts its importance based on how frequently it appears across all documents in the corpus (Inverse Document Frequency, IDF).

Representation: Each document is represented as a vector where each entry corresponds to a word's TF-IDF score. A high TF-IDF score indicates that the word is important in a specific document but rare across the entire corpus.

Applications: TF-IDF is commonly used in text mining and NLP tasks such as text classification, document clustering, information retrieval, and keyword extraction.

Example:
Consider the following corpus:

1. Document 1: "The cat sleeps on the mat."
2. Document 2: "The dog sleeps under the table."

For the word "sleeps":

- **TF**: Appears once in each document divided by the total number of words in the document.
- **IDF**: Appears in both documents, so its importance is lower compared to a word that appears in only one document.

Interactive example 4: Using TF-IDF model to find relative frequency of words - Kaggle notebook

3.1.4. Recurrent Neural Networks (RNNs)

RNNs were one of the first neural models designed to handle sequential data like text. Unlike n-grams, which use a fixed context window, RNNs maintain a hidden state that captures information from all previous words in the sequence, making them capable of learning long-term dependencies.

- **Key Feature**: The hidden state of an RNN is updated at each step in the sequence, allowing the model to theoretically remember information from earlier words.

Example:
In a sentence like **"The cat sleeps on the mat"**, an RNN would process one word at a time, updating its hidden state to capture the sequence context and predict the next word based on the entire sentence.

Interactive example 5: Using RNN to predict words - Kaggle notebook

Challenges:

- **Vanishing Gradient**: RNNs struggle with learning long-term dependencies due to the vanishing gradient problem, where the gradients shrink over time, making it difficult to propagate information from earlier words.
- **Limited Context:** Standard RNNs struggle to capture dependencies in very long sequences, such as remembering the subject of a sentence introduced far earlier.
- **Sequential Processing**: Inefficient as inputs are processed step-by-step, slowing down training.
- **No Parallelism**: Sequential nature hinders efficient use of modern hardware for parallel computation.

Traditional language models, such as n-grams, HMMs, and BoW models, were pivotal in the early stages of NLP. These models enabled tasks like word sequence prediction and part-of-speech tagging by relying on simple statistical or rule-based techniques. However, they struggled with key limitations, such as their inability to capture long-range dependencies, understand deeper context, or effectively handle large vocabularies.

RNNs later replaced these models, introducing a hidden state to retain information across sequential inputs, allowing for a more dynamic representation of context. RNNs marked a significant improvement, enabling better performance on tasks requiring sequential understanding. However, even RNNs faced challenges with scalability and processing very long sequences.

The advent of transformers and LLMs ultimately replaced RNNs, offering a superior ability to capture context over long sequences and process complex language structures. By leveraging large-scale training on diverse datasets, LLMs have set a new standard in NLP, enabling a broader range of tasks to be performed with far greater sophistication.

Model Type	Parameter Size	Context Length	Capabilities
n-gram	Small (fixed n)	Limited (n words)	Basic sequence prediction
HMM	Varies, often small	Limited to recent state	Basic sequence tagging
Bag-of-Words	Small	No context	Word frequency analysis
RNN	Moderate	Dependent on hidden state, limited by vanishing gradients	Sequential data processing, limited long-range dependencies
GPT-3	175 billion	Up to 2048 tokens	Long-range context, text generation
BERT	110 million	Bidirectional (entire sequence)	Contextual understanding, classification

Table 1: Comparison of Traditional and Modern Language Models by Key Characteristics

4. The Rise of LLMs in the Spotlight

A few recent advancements have really brought the spotlight to generative AI and large language models:

4.1. Advancements in Techniques

In recent years, advancements in training techniques have led to remarkable improvements in model performance. A major breakthrough has been the integration of human feedback directly into the training process, driving some of the most substantial gains in model capabilities.

For example, models like **ChatGPT** have been enhanced using **Reinforcement Learning from Human Feedback (RLHF)**, where human evaluators rate model responses. This feedback is then used to fine-tune the model, making its answers more accurate, helpful, and aligned with human expectations.

4.2. Increased Accessibility

The release of ChatGPT made it possible for anyone with internet access to engage directly with one of the most advanced large language models through an easy-to-use web interface. This launch highlighted the remarkable progress in LLM technology, which was previously accessible only to researchers and organizations with extensive resources and specialized expertise.

For example, while models like GPT-3 had been available via API, ChatGPT's web-based interface allowed the public to experience and utilize advanced AI capabilities without needing technical knowledge or large-scale computing resources.

4.3. Growing Computational Power

Advances in powerful computing resources, particularly graphics processing units (GPUs) and, more recently, tensor processing units (TPUs), alongside improved data processing techniques, have revolutionized the training of LLMs. GPUs, originally designed for rendering images and graphics, excel at parallel processing, making them highly effective for general-purpose machine learning tasks. Their versatility and widespread support in frameworks like TensorFlow and PyTorch have made GPUs the backbone of most AI research and development. TPUs, on the other hand, are specialized hardware accelerators developed by Google specifically for deep learning tasks. Unlike GPUs, TPUs are optimized for matrix operations, such as those used in training and inference for neural networks, offering higher performance and energy efficiency in specific use cases.

The choice between GPUs and TPUs depends on the task at hand:

- GPUs are ideal for flexibility and compatibility across diverse machine learning applications, including computer vision, NLP, and fine-tuning pre-trained models.
- TPUs are better suited for large-scale training of deep learning models, particularly in Google Cloud environments where they can leverage their tailored infrastructure for distributed training.

While GPUs dominate general AI tasks due to their adaptability and widespread availability, TPUs provide unparalleled performance in highly specific scenarios, enabling researchers to train significantly larger models with reduced computational costs. Together, these advances have allowed LLMs to achieve greater accuracy and nuanced understanding of language, making them indispensable in modern AI development.

4.4. Improved Training Data

The quality and variety of training data have been pivotal in driving the recent success of LLMs. As methods for collecting and processing vast, high-quality datasets improve, these models can achieve far more accurate and contextually relevant responses. High-quality data enables LLMs to better capture language nuances, understand diverse topics, and respond accurately across a wide range of applications. This shift toward refined data has shown that model performance doesn't only depend on size but also on the richness and relevance of its training data, which has been a major factor in the increasing impact and applicability of LLMs.

Interactive example 6: Using LLMs for sentence generation - Kaggle notebook

5. LLM Applications in Organizations

LLMs have transformed how organizations leverage AI by enabling a wide range of applications across industries. From automating customer interactions to generating content and analyzing data, LLMs provide powerful tools to enhance productivity, decision-making, and user experiences. Below are some of the most common and impactful use cases for LLMs.

5.1. Chatbots and Virtual Assistants

LLMs power chatbots and virtual assistants to handle complex queries and provide personalized responses in real-time.

Example:
Amazon Alexa, Google Assistant, and Microsoft Cortana use LLMs to interact with users via voice or text, answering questions, setting reminders, and controlling smart devices.

5.2. Code Generation and Debugging

LLMs assist developers by generating code snippets, completing partially written code, and identifying errors.

Example:
GitHub Copilot, powered by OpenAI's **Codex**, helps developers write and debug code by providing real-time suggestions based on natural language descriptions.

5.3. Sentiment Analysis

LLMs analyze the emotional tone in text data to monitor customer feedback and brand sentiment.

Example:
Tools like **Hootsuite** and **Brandwatch** use LLMs to analyze social media posts and customer reviews, offering insights into public sentiment for brands.

5.4. Text Classification and Clustering

LLMs categorize and organize large volumes of text into predefined categories, streamlining document and email management.

Example:
Salesforce uses LLMs to automatically classify support tickets, ensuring faster issue resolution by routing them to the correct department.

5.5. Language Translation

LLMs are used for machine translation, enabling businesses to communicate effectively across languages.

Example:
Google Translate and **DeepL** offer real-time language translation services, helping global companies localize their content and communicate with international customers.

5.6. Summarization and Paraphrasing

LLMs summarize long texts or paraphrase content, making it easier to process large volumes of information.

Example:
Reuters and **Bloomberg** use LLMs to condense long reports and articles into concise summaries for quick consumption by readers.

5.7. Content Generation

LLMs generate human-like text for blogs, marketing copy, and product descriptions, helping companies scale content production.

Example:
Jasper AI and **Copy.ai** assist businesses in creating marketing copy, social media content, and blog posts at scale based on prompts.

6. Types of LLMs: Proprietary, open, and open source

When applying LLMs, there are several important considerations to keep in mind regarding the type of model. There are three major categories of LLM: proprietary, open, and open source.

- **Proprietary models (services)** like **GPT-4** (developed by OpenAI) and **Claude 3.5** (from Anthropic) are some of the most popular and powerful models available, but they're developed and operated by private companies. The source code, training strategies, model weights, and even details like the number of parameters they have are all kept secret. The only ways to access these models are through a chatbot or app built with them, or through an API. You can't just run GPT-4o on your own server.

- **Open and open-source models** are more freely available. For example, **you** can download Llama 3 and Gemma 2 from Hugging Face and run them on your own devices. These models can be fine-tuned with your own data, enabling developers to build custom applications. Open-source models also offer full transparency, allowing developers to explore their architecture and model weights.
  Now, what's the difference between open and open source? While companies like Meta or Google label models like Llama 3 as "open"**,** this is distinct from being truly open source.

  Open-source licenses are very permissive, typically allowing for free use, modification, and redistribution. Popular licenses, like MIT or Apache 2.0, even allow for the development of commercial applications without requiring the resulting products to remain open source. However, some licenses (like GPL) do require that derivative works be made open-source. This freedom means that developers can build a wide range of applications, from commercial products to ethically questionable projects, without violating any software licenses—though legal issues may still arise.

Open licenses, like those for Llama 3 or Gemma 2, provide more restrictions than open-source licenses. For example, Llama 3's license allows commercial use up to 700 million monthly users but restricts certain use cases, such as prohibiting large tech companies from using it. Gemma 2’s license blocks certain types of applications, like those that promote illegal activities. These limitations help companies maintain some control over how their models are used, while still allowing broad access for smaller developers.

7. Understanding Popular Large Language Models

Many LLMs have been developed, each showing remarkable performance across various domains, and the competition between companies with the resources and budgets to build these models continues to grow. These models are designed to tackle a wide range of NLP tasks, and each has been developed with specific goals, architectures, and licensing strategies in mind. Among these models are GPT, Gemini, Llama, and others, each with unique strengths and applications.

Some of these models are proprietary, such as GPT, while others are open-source, like Llama. As we mentioned above, Proprietary models, developed by companies such as OpenAI and Google, are typically kept closed to protect the underlying technology, and access is provided through APIs. Open-source models, on the other hand, are made available for developers to download, modify, and fine-tune, giving the broader research and developer community more flexibility. The choice between proprietary and open-source models often depends on factors such as business strategy, intellectual property protection, and the desire to foster community innovation. Open-source models are often released to enable more widespread experimentation and to promote transparency in AI research.

Let us dive deeper into these models to understand what they are based on, why one might be better than another for certain tasks, and the benefits of open-source versus proprietary models.

7.1. GPT (Generative Pre-trained Transformer)

GPT is a **decoder-only transformer** model that generates text based on the previous sequence of words. It uses a unidirectional approach where the model predicts the next word using the context of all prior words.

- **Developer**: OpenAI
- **Type**: Proprietary
- **Strengths**:
  - ✓ **Versatility**: GPT models like **GPT-3** and **GPT-4** excel in generating coherent, human-like text for a wide range of applications, including chatbots, content generation, and question answering.
  - ✓ **Few-shot Learning**: GPT is highly adaptable to new tasks with minimal examples, allowing it to generalize across different domains without extensive task-specific training.

Why Proprietary?

OpenAI keeps GPT proprietary to control its distribution and monetization, offering access through paid APIs. By doing so, OpenAI protects its intellectual property and ensures responsible usage of powerful language models, mitigating risks of misuse. However, this limits customization and experimentation by the broader AI community.

7.2. Gemini

Gemini, which builds on Google's previous LLMs like **PaLM 2**, is a **multimodal** model that can process both text and visual inputs, similar to GPT-4 but with even greater emphasis on combining various types of data for more complex tasks.

- **Developer**: Google DeepMind
- **Type**: Proprietary
- **Strengths**:
  - ✓ **Multimodal Capabilities**: Gemini's ability to handle multiple forms of data, such as text and images, allows it to perform tasks that go beyond text generation, such as image captioning and multimodal question answering.
  - ✓ **Advanced Reasoning**: Gemini is built with advanced reasoning capabilities, making it highly suitable for complex tasks that require deeper understanding and analysis.

Why Proprietary?

Like GPT, Gemini is proprietary, which allows Google to maintain control over its use and commercialization. Given its advanced capabilities, proprietary models like Gemini are typically part of broader service offerings (e.g., through Google Cloud) to support enterprise-level applications.

7.3. Llama (Large Language Model Meta AI)

Llama is a family of **decoder-only transformer** models, similar to GPT, but designed to be smaller and more efficient. It is optimized to run on less powerful hardware, making it more accessible for research and development.

- **Developer**: Meta (formerly Facebook)
- **Type**: Open-source
- **Strengths**:
  - ✓ **Efficiency**: Llama is specifically designed to be more resource-efficient, allowing developers and researchers to use it in environments where large computational resources are not available.
  - ✓ **Customization**: Being open-source, Llama can be downloaded, modified, and fine-tuned by developers for specific applications, offering much more flexibility than proprietary models.

Why Open-source?

Meta made Llama open-source to foster innovation and democratize access to powerful AI models. By making Llama available to the public, Meta allows researchers and developers to experiment with cutting-edge technology without the financial and resource burdens associated with proprietary models.

7.4. Other Models: BERT, T5, RoBERTa

- **BERT (Bidirectional Encoder Representations from Transformers)**: Developed by **Google**, BERT is a **bidirectional transformer** model that reads text from both directions, making it highly effective for tasks like question answering and text classification. BERT's open-source nature has made it widely adopted and fine-tuned for a variety of tasks.

- **T5 (Text-to-Text Transfer Transformer)**: Also developed by **Google**, T5 frames every NLP task as a text-to-text problem, offering unparalleled flexibility. T5 is open-source, making it highly adaptable to a wide range of applications, from translation to summarization.

- **RoBERTa**: Developed by **Facebook AI**, RoBERTa is an optimized version of BERT, designed for better performance through more extensive training and removing certain constraints. RoBERTa is open-source and available for developers to fine-tune for specific tasks.

The table provides a comprehensive summary of various LLM models, covering essential details such as the type, model name, number of parameters, release date, base models, open-source status, token count, and training dataset. This information helps illustrate each model's capabilities and intended use cases, allowing for a clear comparison between models based on their specifications and unique features.

Type	Model Name	#Params	Date of Release	Base Models	Open Source	# Tokens	Training Dataset

Type	Model Name	#Params	Date of Release	Base Models	Open Source	# Tokens	Training Dataset
Encoder-Only	BERT	110M, 340M	2018	-	√	137B	BooksCorpus, English Wikipedia
	RoBERTa	355M	2019	-	√	2.2T	BooksCorpus, English Wikipedia, CC-NEWS, STORIES (a subset of Common Crawl), Reddit
	DeBERTa	-	2020	-	√	-	BooksCorpus, English Wikipedia, STORIES, Reddit content
	XLNet	110M, 340 M	2019	-	√	32.89B	BooksCorpus, English Wikipedia, Giga5, Common Crawl, ClueWeb 2012-B
Decoder-Only	GPT-1	120M	2018	-	√	1.3B	BooksCorpus
	GPT-2	1.5B	2019		√	10B	Reddit outbound
Encoder-Decoder	T5 (Base)	223M	2019	-	√	156B	Common Crawl
	MT5 (Base)	300M	2020	-	√	-	New Common Crawl-based dataset in 101 languages (m Common Crawl)
	BART (Base)	139M	2019	-	√	-	Corrupting text
GPT Family	GPT-3	125M, 350M, 760M, 1.3B, 2.7B, 6.7B, 13B, 175B	2020	-	×	300B	Common Crawl (filtered), WebText2, Books1, Books2, Wikipedia
	CODEX	12B	2021	GPT	√	-	Public GitHub software repositories

	WebGPT	760M, 13B, 175B	2021	GPT-3	×	-	ELI5
	GPT-4	1.76T	2023	-	×	13T	-
LLaMa Family	LLaMA1	7B, 13B, 33B, 65B	2023	-	√	1T, 1.4T	Online sources
	LLaMA2	7B, 13B, 34B, 70B	2023	-	√	2T	Online sources
	Alpaca	7B	2023	LLaMA1	√	-	GPT-3.5
	Koala	13B	2023	LLaMA1	√	-	GPT-3.5
	Mistral-7B	7.3B	2023	LLaMA	√	-	-
	Code Llama	34B	2023	LLaMA2	√	500B	Publicly available code
PaLM Family	PaLM	8B, 62B, 540B	2022	-	×	780B	Web documents, books, Wikipedia, conversations, GitHub code
	PaLM 2	340B	2023	-	√	3.6T	Web documents, books, code, mathematics, conversational data
	Med-PaLM	540B	2022	PaLM	×	780B	HealthSearchQA, MedicationQA, LiveQA
	Med-PaLM 2	-	2023	PaLM 2	×	-	MedQA, MedMCQA, HealthSearchQA, LiveQA, MedicationQA
	FLAN	137B	2021	LaMDA-PT	√	-	Web documents, code, dialog data, Wikipedia
	ERNIE 4.0	10B	2023	-	×	4TB	Chinese Text
	LaMDA	137B	2022	-	×	168B	public dialog data and web documents

Other Popular LLMs	ChinChilla	70B	2022	-	×	1.4T	Massive Text
	Falcon 180B	180B	2023	-	√	3.5T	RefinedWeb
	Gemini	1.8B, 3.25B	2023	-	√	-	Web documents, books, and code, image data, audio data, video data
	DeepSeek-Coder	1B, 7B	2024	-	√	2T	GitHub's Markdown and StackExchange
	DocLLM	1B, 7B	2024	-	×	2T	IIT-CDIP Test Collection 1.0, DocBank
	StartCoder	15.5B	2023	-	√	35B	GitHub
Type	**Model Name**	**#Params**	**Date of Release**	**Base Models**	**Open Source**	**# Tokens**	**Training Dataset**

Table 2: Overview of Large Language Models with Key Specifications

8. Key Considerations for Training LLMs

As we explained before, training LLMs from scratch is a difficult task that can have high cost and complexity. Here are some of the key challenges.

8.1. Infrastructure

LLMs are trained on huge text corpora, typically at least 1000 GB in size. Furthermore, the models employed for training on such datasets are enormous, with billions of parameters. An infrastructure with multiple GPUs is essential to be able to train such large models.

To illustrate the computational requirements, training GPT-3, a previous-generation model with 175 billion parameters, would take 288 years to train on one NVIDIA V100 GPU. Typically, LLMs are trained on thousands of GPUs in parallel. For example, Google trained its PaLM model with 540 billion parameters by distributing training over 6,144 TPU v4 chips.

8.2. Cost

However, acquiring and hosting such a large number of GPUs is not possible for most organizations. Even OpenAI, creator of the GPT series of models and the popular ChatGPT, did not train their models on their own infrastructure, but instead relied on Microsoft's Azure cloud platform. In January 2023, Microsoft announced a multiyear, multibillion-dollar investment in OpenAI, marking the third phase of their partnership. This agreement builds upon previous investments made in 2019 and 2021.

8.3. Model Distribution Strategies

Beyond the scale and cost, there are also complex considerations in how to run LLM training on the computing resources.

In particular:

- LLMs are first trained on a single GPU to get an idea of their resource requirements.
- Model parallelism is an important strategy. This involves distributing models across numerous GPUs, with optimal partitioning designed to enhance memory and I/O bandwidth.
- With very large models, there is a need for Tensor model parallelism. This approach distributes individual layers of the model across multiple GPUs. This requires precise coding, configuration, and careful implementation for accurate and efficient execution.
- LLM training is iterative in nature. Various parallel computing strategies are often used, and researchers experiment with different configurations, adjusting training runs to the specific needs of the model and available hardware.

8.4. Impact of Model Architecture Choices

The chosen LLM architecture has a direct impact on training complexity.

Here are a few guidelines for adapting the architecture to the available resources:

- The model's depth and width (in terms of number of parameters) should be selected to achieve a balance between available computational resources and complexity.
- It is preferable to use architectures with residual connections. This makes it easier to optimize resource utilization.
- Determine the need for a Transformer architecture with self-attention, because this imposes specific training requirements.
- Identify the functional needs of the model, such as generative modeling, bi-directional/masked language modeling, multi-task learning, and multi-modal analysis.
- Perform training runs with familiar models like GPT, BERT, and XLNet to understand their applicability to your use case.
- Determine your tokenization technique: Word-based, subword, or character based. This can impact vocabulary size and input length, directly impacting computations requirements.

Conclusion

This module provided a comprehensive exploration of large language models (LLMs), expanding on the foundational principles of AI transformers. We explored the advancements that have enabled LLMs to process and generate human language with unprecedented accuracy and scale. From tracing their historical evolution to distinguishing them from traditional language models, we explored the innovations that have propelled LLMs to the forefront of AI development.

LLMs, as a specialized subset of generative AI, have revolutionized industries by powering diverse applications, including chatbots, language translation, and code generation. Their versatility, achieved through extensive pre-training, fine-tuning, and access to massive datasets, has made them an indispensable component of modern NLP solutions.

By analyzing leading models such as GPT, BERT, and Llama and their unique capabilities, we highlighted their transformative impact and the growing accessibility of LLMs through APIs, which simplify adoption and lower barriers to entry for organizations.

Looking ahead, we will delve deeper into how LLMs are trained, the critical role of datasets in both training and fine-tuning, and strategies for integrating LLMs into practical applications. This next phase will focus on data preparation for LLMs, fine-tuning methods, and optimization techniques, providing you with the insights and tools needed to fully harness the power of LLMs in real-world scenarios.

Other Resources:

LLMs vs GenAI

History, Development, and Principles of LLMs

The difference between a CPU, GPU, and TPU

LLMs Vs Traditional Language Models

Real-Life Applications of LLMs

Popular LLMs