A Beginner's Guide to Large Language Models

Part 1

Contributors:
Annamalai Chockalingam
Ankur Patel
Shashank Verma
Tiffany Yeung

Table of Contents

Preface

Language has been integral to human society for thousands of years. A long-prevailing theory, laryngeal descent theory or LDT, suggests that speech and, thus, language, may have evolved about 200,000 or 300,000 years ago, while newer research shows it could've happened even sooner.

Regardless of when it first appeared, language remains the cornerstone of human communication. It has taken on an even greater role in today's digital age, where an unprecedented portion of the population can communicate via both text and speech across the globe.

This is underscored by the fact that 347.3 billion email messages are sent and received worldwide every day, and that five billion people – or over 63% of the entire world population – send and receive text messages.

Language has therefore become a vast trove of information that can help enterprises extract valuable insights, identify trends, and make informed decisions. As an example, enterprises can analyze texts like customer reviews to identify their products' best-selling features and fine-tune their future product development.

Similarly, language production – as opposed to language *analysis* – is also becoming an increasingly important tool for enterprises. Creating blog posts, for example, can help enterprises raise brand awareness to a previously unheard-of extent, while composing emails can help them attract new stakeholders or partners at an unmatched speed.

However, both language analysis and production are time-consuming processes that can distract employees and decision-makers from more important tasks. For instance, leaders often need to sift through vast amounts of text in order to make informed decisions instead of making them based on extracted key information.

Enterprises can minimize these and other problems, such as the risk of human error, by employing **large language models (LLMs)** for language-related tasks. LLMs can help enterprises accelerate and largely *automate* their efforts related to both language production and analysis, saving valuable time and resources while improving accuracy and efficiency.

Unlike previous solutions, such as rule-based systems, LLMs are incredibly versatile and can be easily adapted to a wide range of language-related tasks, like generating content or summarizing legal documentation.

The goal of this book is to help enterprises understand what makes LLMs so groundbreaking compared to previous solutions and how they can benefit from adopting or developing them. It also aims to help enterprises get a head start by outlining the most crucial steps to LLM development, training, and deployment.

To achieve these goals, the book is divided into three parts:

- **Part 1** defines LLMs and outlines the technological and methodological advancements over the years that made them possible. It also tackles more practical topics, such as how enterprises can develop their own LLMs and the most notable companies in the LLM field. This should help enterprises understand how adopting LLMs can unlock cutting-edge possibilities and revolutionize their operations.

- **Part 2** discusses five major use cases of LLMs within enterprises, including content generation, summarization, and chatbot support. Each use case is exemplified with real-life apps and case studies, so as to show how LLMs can solve real problems and help enterprises achieve specific objectives.

- **Part 3** is a practical guide for enterprises that want to build, train, and deploy their own LLMs. It provides an overview of necessary pre-requirements and possible trade-offs with different development and deployment methods. ML engineers and data scientists can use this as a reference throughout their LLM development processes.

Hopefully, this will inspire enterprises that have not yet adopted or developed their own LLMs to do so soon in order to gain a competitive advantage and offer new SOTA services or products. The most benefits will be, as usual, reserved for early adopters or truly visionary innovators.

Glossary

Terms	Description
Deep learning systems	Systems that rely on neural networks with many hidden layers to learn complex patterns.
Generative AI	AI programs that can generate new content, like text, images, and audio, rather than just analyze it.
Large language models (LLMs)	Language models that recognize, summarize, translate, predict, and generate text and other content. They're called large because they are trained on large amounts of data and have many parameters, with popular LLMs reaching hundreds of billions of parameters.
Natural language processing (NLP)	The ability of a computer program to understand and generate text in natural language.
Long short-term memory neural network (LSTM)	A special type of RNNs with more complex cell blocks that allow it to retain more past inputs.
Natural language generation (NLG)	A part of NLP that refers to the ability of a computer program to generate human-like text.
Natural language understanding (NLU)	A part of NLP that refers to the ability of a computer program to understand human-like text.
Neural network (NN)	A machine learning algorithm in which the parameters are organized into consecutive layers. The learning process of NNs is inspired by the human brain. Much like humans, NNs "learn" important features via representation learning and require less human involvement than most other approaches to machine learning.
Perception AI	AI programs that can process and analyze but not generate data, mainly developed before 2020.
Recurrent neural network (RNN)	Neural network that processes data sequentially and can memorize past inputs.

Terms	Description
Rule-based system	A system that relies on human-crafted rules to process data.
Traditional machine learning	Traditional machine learning uses a statistical approach, drawing probability distributions of words or other tokens based on a large annotated corpus. It relies less on rules and more on data.
Transformer	A type of neural network architecture designed to process sequential data non-sequentially.
Structured data	Data that is quantitative in nature, such as phone numbers, and can be easily standardized and adjusted to a pre-defined format that ML algorithms can quickly process.
Unstructured data	Data that is qualitative in nature, such as customer reviews, and difficult to standardize. Such data is stored in its native formats, like PDF files, before use.
Fine-tuning	A transfer learning method used to improve model performance on selected downstream tasks or datasets. It's used when the target task is similar to the pre-training task and involves copying the weights of a PLM and tuning them on desired tasks or data.
Customization	A method of improving model performance by modifying only one or a few selected parameters of a PLM instead of updating the entire model. It involves using parameter-efficient techniques (PEFT).
Parameter-efficient techniques (PEFT)	Techniques like prompt learning, LoRa, and adapter tuning which allow researchers to customize PLMs for downstream tasks or datasets whil preserving and leveraging existing knowledge of PLMs. These techniques are used during model customization and allow for quicker training and often more accurate predictions.
Prompt learning	An umbrella term for two PEFT techniques, prompt tuning and p-tuning, which help customize models by inserting virtual token embeddings among discrete or real token embeddings.
Adapter tuning	A PEFT technique that involves adding lightweight feed-forward layers, called adapters, between existing PLM layers and updating only their weights during customization while keeping the original PLM weights frozen.
Open-domain question answering	Answering questions from a variety of different domains, like legal, medical, and financial, instead of just one domain.
Extractive question answering	Answering questions by extracting the answers from existing texts or databases.

Terms	Description
Throughput	A measure of model efficiency and speed. It refers to the amount of data or the number of predictions that a model can process or generate within a pre-defined timeframe.
Latency	The amount of time a model needs to process input and generate output.
Data Readiness	The suitability of data for use in training, based on factors such as data quantity, structure, and quality.

Introduction to LLMs

A large language model is a type of artificial intelligence (AI) system that is capable of generating human-like text based on the patterns and relationships it learns from vast amounts of data. Large language models use a machine learning technique called deep learning to analyze and process large sets of data, such as books, articles, and web pages.

Large language models unlocked numerous unprecedented possibilities in the field of NLP and AI. This was most notably demonstrated by the release of OpenAI's GPT-3 in 2020, the then-largest language model ever developed.

These models are designed to understand the context and meaning of text and can generate text that is grammatically correct and semantically relevant. They can be trained on a wide range of tasks, including language translation, summarization, question answering, and text completion.

GPT-3 made it evident that large-scale models can accurately perform a wide – and previously unheard-of – range of NLP tasks, from text summarization to text generation. It also showed that LLMs could generate outputs that are nearly indistinguishable from human-created text, all while learning on their own with minimal human intervention.

This presented an enormous improvement from earlier, mainly rule-based models that could neither learn on their own nor successfully solve tasks they weren't trained on. It is no surprise, then, that many other enterprises and startups soon started developing their own LLMs or adopting existing LLMs in order to accelerate their operations, reduce expenses, and streamline workflows.

Part 1 is intended to provide a solid introduction and foundation for any enterprise that is considering building or adopting its own LLM.

What Are Large Language Models (LLMs)?

Large language models (LLMs) are deep learning algorithms that can recognize, extract, summarize, predict, and generate text based on knowledge gained during training on very large datasets.

They're also a subset of a more general technology called language models. All language models have one thing in common: they can process and generate text that sounds like natural language. This is known as performing tasks related to **natural language processing (NLP)**.

Although all language models can perform NLP tasks, they differ in other characteristics, such as their size. Unlike other models, LLMs are considered *large* in size because of two reasons:

1. They're trained using large amounts of data.

2. They comprise a huge number of learnable parameters (i.e., representations of the underlying structure of training data that help models perform tasks on new or never-before-seen data).

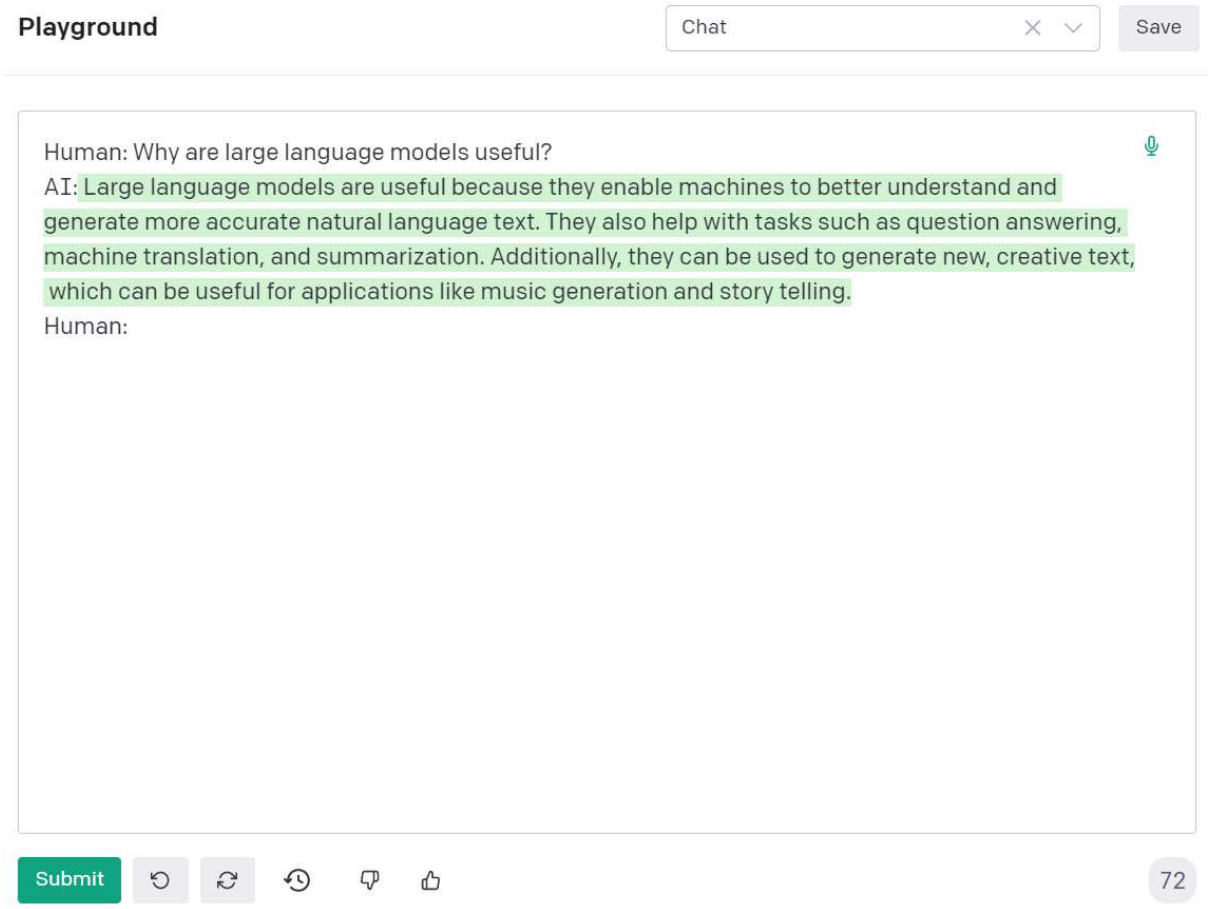
Table 1 showcases two large language models, MT-NLG and GPT-3 Davinci, to help clarify what's considered *large* by contemporary standards.

Table 1. Comparison of MT-NLG and GPT-3

Large Language Model	**Number of parameters**	**Number of tokens in the training data**
NVIDIA Model: Megatron-Turing Natural Language Generation Model (MT-NLG)	530 billion	270 billion
OpenAI Model: GPT-3 Davinci Model	175 billion	499 billion

Since the quality of a model heavily depends on the model size and the size of training data, larger language models typically generate more accurate and sophisticated responses than their smaller counterparts.

Figure 1. Answer Generated by GPT-3.



However, the performance of large language models doesn't just depend on the model size or data quantity. Quality of the data matters, too.

For example, LLMs trained on peer-reviewed research papers or published novels will usually perform better than LLMs trained on social media posts, blog comments, or other unreviewed content. Low-quality data like user-generated content may lead to all sorts of problems, such as models picking up slang, learning incorrect spellings of words, and so on.

In addition, models need very diverse data in order to perform various NLP tasks. However, if the model is intended to be especially good at solving a particular set of tasks, then *fine-tune* it using a more relevant and narrower dataset. By doing so a foundation language model is transformed — from one that's good at performing various NLP tasks across a broad set of domains – into a fine-tuned model that specializes in performing tasks in a narrowly scoped domain.

Foundation Language Models vs. Fine-Tuned Language Models

Foundation language models, such as the aforementioned MT-NLG and GPT-3, are what is usually referred to when discussing LLMs. They're trained on vast amounts of data and can perform a wide variety of NLP tasks, from answering questions and generating book summaries to completing and translating sentences.

Thanks to their size, foundation models can perform well even when they have little domain-specific data at their disposal. They have good general performance across tasks but may not excel at performing any one specific task.

Fine-tuned language models, on the other hand, are large language models derived from foundation LLMs. They're customized for specific use cases or domains and, thus, become better at performing more specialized tasks.

Apart from the fact that fine-tuned models can perform specific tasks better than foundation models, their biggest strength is that they are lighter and, generally, easier to train. But how does one actually fine-tune a foundation model for specific objectives?

Currently, the most popular method is customizing a model using parameter-efficient customization techniques, such as p-tuning, prompt tuning, adapters, and so on. Customization is far less time-consuming and expensive than fine-tuning the entire model, although it may lead to somewhat poorer performance than other methods. Customization methods are further discussed in Part 3.

Evolution of Large Language Models

AI systems were historically about processing and analyzing data, not generating it. They were more oriented toward perceiving and understanding the world around us rather than on generating new information. This distinction marks the main difference between *Perceptive* and *Generative AI*, with the latter becoming increasingly prevalent since around 2020, or after companies started adopting transformer models and developing increasingly more robust LLMs at a large scale.

The advent of large language models further fueled a revolutionary paradigm shift in the way NLP models are designed, trained, and used. To truly understand this, it may be helpful to compare large language models to previous NLP models and how they worked. For this purpose, let's briefly explore three regimes in the history of NLP: pre-transformers NLP, transformers NLP, and LLM NLP.

1. **Pre-transformers NLP** was mainly marked by models that relied on human-crafted rules rather than machine learning algorithms to perform NLP tasks. This made them suitable for simpler tasks that didn't require too many rules, like text classification, but unsuitable for more complex tasks, such as machine translation. Rule-based models also performed poorly in edge-case scenarios because they couldn't make accurate predictions or classifications for never-before-seen data for which no clear rules were set. This problem was somewhat solved with simple neural networks, such as RNNs and LSTMs, developed during the later phases of this period. RNNs and LSTMs could memorize past data to a certain extent and, thus, provide context-dependent predictions and

classifications. However, RNNs and LSTMs could not make predictions over long spans of text, limiting their effectiveness.

2. **Transformers NLP** was set in motion by the rise of the transformer architecture in 2017. Transformers could generalize better than the then-prevailing RNNs and LSTMs, capture more context, and process more data at once. These improvements enabled NLP models to understand longer sequences of data and perform a much wider range of tasks. However, from today's point of view, models developed during this period had limited capabilities, mainly due to the general lack of large-scale datasets and adequate computational resources. They also mainly sparked attention among researchers and experts in the field but not the general public, as they weren't user-friendly nor accurate enough to become commercialized.

3. **LLM NLP** was mainly initiated by the launch of OpenAI's GPT-3 in 2020. Large language models like GPT-3 were trained on massive amounts of data, which allowed them to produce more accurate and comprehensive NLP responses compared to previous models. This unlocked many new possibilities and brought us closer to achieving what many consider "true" AI. Also, LLMs made NLP models much more accessible to non-technical users who could now solve a variety of NLP tasks just by using natural-language prompts. *NLP technology was finally democratized.*

The switch from one methodology to another was largely driven by relevant technological and methodological advancements, such as the advent of neural networks, attention mechanisms, and transformers and developments in the field of unsupervised and self-supervised learning. The following sections will briefly explain these concepts, as understanding them is crucial for truly understanding how LLMs work and how to build new LLMs from scratch.

Neural Networks

Neural networks (NNs) are machine learning algorithms loosely modeled after the human brain. Like the biological human brain, artificial neural networks consist of neurons, also called nodes, that are responsible for all model functions, from processing input to generating output.

The neurons are further organized into layers, vertically stacked components of NNs that perform specific tasks related to input and output sequences.

Every neural network has at least three layers:

- **The input layer** accepts data and passes it to the rest of the network.
- **The hidden layer**, or multiple hidden layers, performs specific functions that make the final output of an NN possible. These functions can include identifying or classifying data, generating new data, and other functions depending on the specific NLP task in question.
- **The output layer** generates a prediction or classification based on the input.

When LLMs were first developed, they were based on simpler NN architectures with fewer layers, mainly recurrent neural networks (RNNs) and long short-term memory networks (LSTMs). Unlike other neural networks, RNNs and LSTMs could take into account the context, position, and relationships between words even if they were far apart in a data sequence. Simply put, this meant they could memorize and consider past data when generating output, which resulted in more accurate solutions to many NLP tasks, especially sentiment analysis and text classification.

The biggest advantage that neural networks like RNNs and LSTMs had over traditional, rule-based systems was that they were capable of learning on their own with little to no human involvement. They analyze data to create their own rules, rather than learn the rules first and apply them to data later. This is also known as representation learning and is inspired by human learning processes.

Representations, or features, are hidden patterns that neural networks can extract from data. To exemplify this, let's imagine we're training an NN-based model on a dataset containing the following tokens:

"cat," "cats," dog," "dogs"

After analyzing these tokens, the model may identify a representation that one could formulate as:

Plural nouns have the suffix "-s."

The model will then extract this representation and apply it to new or edge-case scenarios whose data distribution follows that of training data. For example, the assumption can be made that the model will correctly classify tokens like *"chairs"* or *"table"* as plural or singular even if it had not encountered them before. Once it encounters irregular nouns that don't follow the extracted representation, the model will update its parameters to reflect new representations, such as:

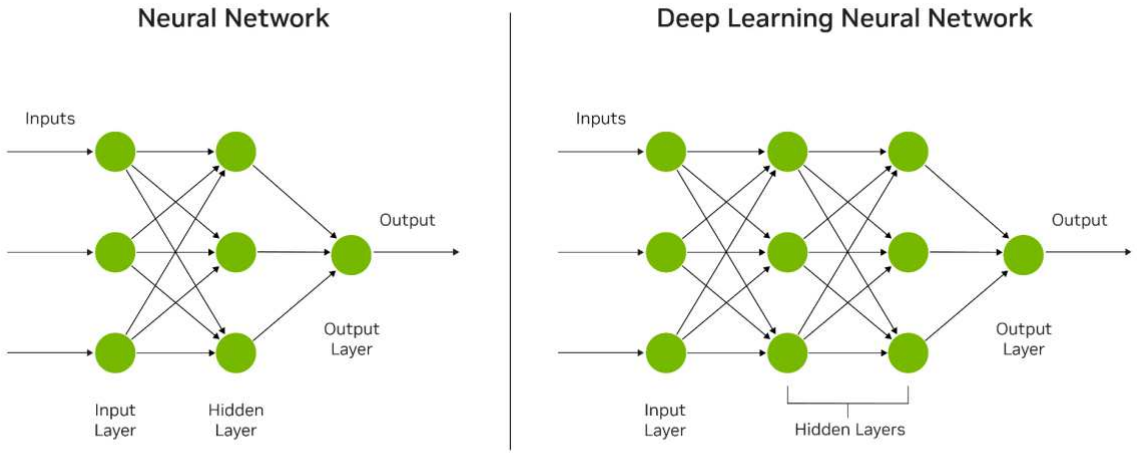
Plural nouns are followed by plural verbs.

This approach enables NN-based models to generalize better than rule-based systems and successfully perform a wider range of tasks.

However, their ability to extract representations is very much dependent on the number of neurons and layers comprising a network. The more neurons neural networks have, the more complex representations they can extract. That's why, today, most large language models use deep learning neural networks with multiple hidden layers and, thus, a higher number of neurons.

Figure 2 shows a side-by-side comparison of a single-layer neural network and a deep learning neural network.

Figure 2. Comparison of Single-Layer vs. Deep Learning Neural Network



While this may seem like an obvious choice today, consider that developing deep neural networks did not make sense before the hardware evolved to be able to handle massive workloads. This only became possible after ~1999 when NVIDIA introduced “the world’s first GPU,” or graphics processing unit, to the wider market or, more precisely, after a wildly successful CNN called AlexNet popularized their use in deep learning in 2012.

GPUs had a highly parallelizable architecture which enabled the rapid advances in deep learning systems that are seen today. Among other advancements, the advent of GPUs ushered in the development of a new type of neural network that would revolutionize the field of NLP: transformers.

Transformers

While RNNs and LSTMs have their advantages, especially compared to traditional models, they also have some limitations that make them unsuitable for more complex NLP tasks, such as machine translation. Their main limitation is the inability to process longer data sequences and, thus, consider the overall context of the input sequence. Because LSTMs and RNNs cannot handle too much context well, their outputs are prone to being inaccurate or nonsensical. This and other challenges have been largely overcome with the advent of new, special neural networks called transformers.

Transformers were first introduced in 2017 by Vaswani et al. in a paper titled "Attention is All You Need." The title alluded to attention mechanisms, which would become the key component of transformers.

“We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely.” - Vaswani et. al, “Attention is All You Need”

The paper proposed that attention mechanisms would render recurrence and convolutions obsolete in regard to sequential data and make transformers more suitable for machine translation than RNNs and CNNs. This prophecy would soon come true, as transformers would go on to become the dominant architecture not only for machine translation but for all NLP tasks.

Attention mechanisms solved the problem of inadequate context handling by allowing models to selectively pay attention to certain parts of the input while processing it. Instead of having to capture the entire context at once, models could now focus on the most important tokens regarding a specific task.

To demonstrate this, let’s imagine the desired model is a transformer-based model to predict the next words for the following input sentence:

```
Mary had a little lamb.
```

Attention mechanisms – or, rather, *self-attention layers* that are based on attention mechanisms – would first calculate attention weights for each word in our input. Attention weights represent the importance of each token, so the more weight a token is assigned, the more important it's deemed. For example, the attention mechanism might give more weight to the word *"lamb"* than the word *"a,"* as it’s likely to have more influence on the final output.

The model would then use these weights to dynamically emphasize or downplay each word as it generates output. If one assumes that the most weight was assigned to the word *“lamb,”* the model may produce a continuation such as:

```
"whose fleece was white as snow"
```

To determine how important each token is, self-attention layers examine its relationships with other tokens in a sequence:

1. If a token has many relevant relationships with other tokens with respect to the task being performed, then that token is deemed as important and, potentially, *more* important than other tokens in the same sequence.

2. If a token doesn’t have many relationships with other tokens, or if they are irrelevant to a specific task, that token is considered less important or completely unimportant. This means the model will virtually ignore it when generating the output.

So, by enabling models to handle context more effectively, attention mechanisms allowed them to generate more accurate outputs than models based on RNNs and LSTMs. Simultaneously, this new approach to data processing also allowed transformer-based models to generate outputs more quickly than RNN- and LSTM-based models.

LSTMs and RNNs need more time to generate output because they process input *sequentially*. To clarify what this means, let's explore how LSTMs would approach processing our original input sentence:

Mary had a little lamb.

Since LSTMs process data sequentially, they would need to process one word in our sequence at a time: *Mary, had, a, little, lamb*. This significantly slows down inference, especially with longer data sequences. For example, just imagine how long it would take LSTMs and RNNs to process a single Wikipedia page. *Too long.*

Transformers, on the other hand, process data *in parallel*, which means they "read" all input tokens at once instead of processing one at a time. It also means they are able to perform NLP tasks faster than LSTMs and RNNs.

However, despite being slow, sequential data processing has one big advantage. By processing one word at a time, LSTMs and RNNs are always able to tell which word came first, second, and so on. They know the word order of the input sequence because they use that same order to process it. Conversely, transformers are not initially "aware" of the original word order because they process data non-sequentially. While this may seem like only a minor problem at first, analyzing the sentences below may illustrate otherwise:

1. Mary had a little lamb.

2. A little lamb had Mary.

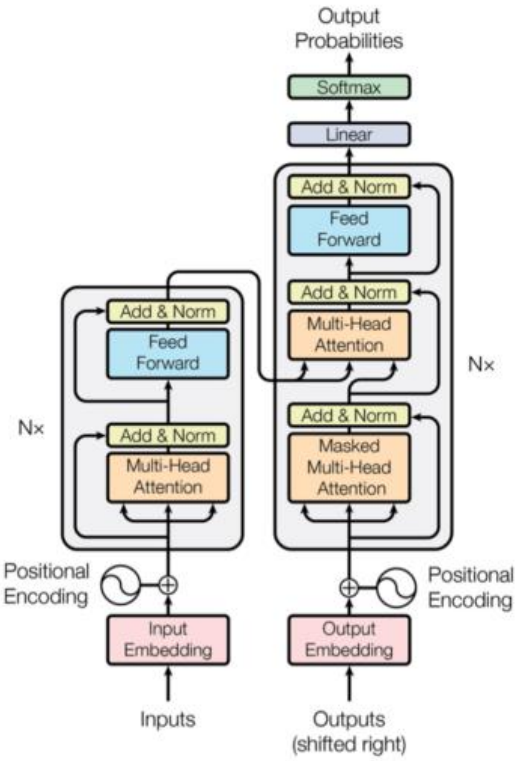
3. Had a little lamb Mary.

Sentence (2) shows how a slight change in word order can distort the intended meaning, while sentence (3) exemplifies an even bigger issue — how changes in word order can result in completely nonsensical and grammatically incorrect variations.

To overcome this challenge, transformers use positional encodings that help them retain position information. Positional encodings are additional inputs, or vectors, associated with each token. They can be fixed or trainable, depending on whether the desire is for the model to refine them during training or not.

Figure 3. Sequential Data Processing in a LSTM

Source: Attention Is All You Need



Researchers and companies would soon start implementing these new mechanisms and building new transformer-based models, with Google releasing its famous BERT in 2018.

BERT

Google's BERT (Bidirectional Encoder Representations from Transformers) is one of the first transformer-based language models. It's a masked language model (MLM), which means it's trained on sentences containing masked tokens. The model needs to predict the masked token by considering its surrounding context. To illustrate this, let's imagine that a model is given the following input sentence:

```
"I [have] a mask."
```

BERT's task is to predict the masked word *"have."* It does so by analyzing the tokens on both of its sides, namely *"I,", "a,"* and *"mask."* This is what makes it *bidirectional*, as well as more accurate than previous language models that could only consider the context on the left of the masked token. In this case, *unidirectional* models would only consider the word *"I"* when predicting the masked word, which provides little context. The chances of a unidirectional model generating the right predictions are smaller.

BERT was the first model to show how bidirectionality can model performance NLP tasks. It has been used for various purposes, including improving the accuracy of Google's search results by allowing for a better understanding of the context and meaning of queries.

Other Large Language Models

Google's BERT and new advancements in the field inspired other companies to start building their own large language models. In the table below, we've listed some models developed prior to the truly groundbreaking GPT-3, released in June 2020.

Table 2 shows a timeline of consequential LLM releases.

Table 2. Large Language Model Release Timeline

Model	Company	Year	Brief Description
GPT-2	OpenAI	2019	A transformer-based language model designed to generate human-like text and perform various NLP tasks, including language translation, summarization, and question answering.
RoBERTa	Facebook	2019	A BERT-based model designed to improve the performance of NLP tasks by training the model on a larger dataset and using a more efficient training method.
DeBERTa	Microsoft	2020	A BERT-based model designed to improve the performance of NLP tasks by decoupling the encoder and decoder components of the model.
GPT-3	OpenAI	2020	An upgraded model of GPT-2 trained on a more massive dataset and capable of generating higher-quality outputs.

By proving that LLMs can be used for few-shot learning and excel without "large-scale task-specific data collection or model parameter updating," GPT-3 would inspire companies to build even larger models, like Megatron-Turing Natural Language Generation with 530 billion parameters, PaLM with 540 billion, and WuDao 2.0 with impressive 1.75 trillion parameters.

Unsupervised and Self-Supervised Learning

BERT wasn't revolutionary just because it was a bidirectional model, but also because it was trained using *unsupervised learning*. Unsupervised learning refers to machine learning algorithms finding patterns in unlabeled datasets with no human intervention. In BERT's case, the model had to extract patterns from plain-language Wikipedia pages on its own during training. This is often considered to be AI in its purest form.

Unsupervised learning models use feedback loops to learn and improve their performance. This involves getting feedback on whether a prediction or classification was right or wrong, which the model uses to guide its future decisions.

Feedback loops are the reason why some differentiate between unsupervised and *self-supervised learning*. Self-supervised learning models don't have feedback loops but rather use supervisory signals

to get feedback during training. These signals are generated automatically from data without human annotation.

Both unsupervised and self-supervised learning techniques have one key advantage over supervised learning: they rely on the model to create labels and extract features on its own, rather than demanding human intervention. This helps companies train models without time-consuming data labeling processes or providing human feedback on the model's outputs.

Self-supervised learning is currently the dominant approach to pre-training large language models and is often recommended to enterprises that want to build their own.

Benefits of GPT Vs Bert

GPT (Generative Pre-trained Transformer) and BERT (Bidirectional Encoder Representations from Transformers) are both highly advanced and widely used natural language processing (NLP) models. However, they differ in their architectures and use cases.

GPT is a generative model that is trained to predict the next word in a sentence given the previous words. This pre-training enables GPT to generate coherent and fluent sentences from scratch, making it ideal for language generation tasks such as text completion, summarization, and question answering.

In distinction, BERT is a discriminative model that is trained to classify sentences or tokens into different categories such as sentiment analysis, named entity recognition, and text classification. It is a bidirectional model that considers both the left and right contexts of a sentence to understand the meaning of a word, making it highly effective for tasks such as sentiment analysis and question answering.

In terms of architecture, GPT uses a unidirectional transformer, whereas BERT uses a bidirectional transformer. This means that GPT can only consider the left context of a word when making predictions, while BERT considers both the left and right contexts.

Both GPT and BERT are powerful models that have revolutionized the field of NLP. Their choice depends on the specific task at hand, and researchers and practitioners often use a combination of both models to achieve optimal results.

How Enterprises Can Benefit From Using Large Language Models

Enterprises need to tackle language-related tasks every day. This includes more obvious text tasks, such as writing emails or generating content, but also tasks like analyzing patient data for health risks or providing companionship to customers. All of these tasks can be automated using large language models.

Models, or applications powered by large language models, can help enterprises speed up many complex tasks and often execute them with a higher level of precision than human agents. For example, tech enterprises can use them to write code faster, while banks can use them to minimize the risk of human error when analyzing documents for fraud indications.

Automating complex, but often tedious tasks further allow employees to focus on more important tasks instead and make progress faster. We'll see, for example, how healthcare enterprises can use LLMs to generate synthetic clinical data and use it to speed up medical research in Part 2.

LLMs can benefit enterprises in many other ways, depending on how they are used. Some use cases, like LLM-based sentiment analysis, provide them with deeper insights about their audience, while customer churn prediction allows them to encourage customers to stay with their company just as they were about to leave it. Additionally, enterprises can use LLMs to offer new, conversation-based services, such as specialized AI companions.

Challenges of Large Language Models

Enterprises that want to start using large language models, or applications powered by large language models, should be aware of a few common LLM-related pitfalls. Below are some general ones that are applicable regardless of whether a model is being customized, fine-tuned, or built from scratch.

1. Large language models are vulnerable to adversarial examples. Adversarial examples are inputs specifically crafted to fool the models into making a mistake. This can raise security concerns, particularly for enterprises in sensitive industries like healthcare or finance.

2. Large language models can lack interpretability. Interpretability refers to the ability to interpret and predict models' decisions. Models with low interpretability can be difficult to troubleshoot and evaluate, as it may not be clear how they are making their decisions or how accurate or unbiased those decisions are. This can be especially problematic in the context of high-stake use cases, such as fraud detection, and in industries that require a high level of transparency, such as healthcare and finance.

3. Large language models may provide un-customized, generic answers. As such, LLMs may not always respond well to human input or understand the intent behind it. This can be improved with techniques such as Reinforcement Learning from Human Feedback (RLHF), which help models improve their performance over time based on positive or negative human feedback. Even so, LLMs can sometimes reproduce the text data they've seen during training. This is problematic only from an ethical angle but may also expose enterprises to unwanted copyright and legal issues.

4. Using large language models can raise ethical concerns. It's questionable whether enterprises should use LLMs for important decision-making tasks, such as deciding which candidate is the most qualified based on collected resumes, especially without human supervision. Additionally, it should be assessed whether it's ethical to use LLMs for tasks that would normally be performed by human, mainly white-collar workers.

5. Large language models can generate inappropriate and harmful content. On that note, enterprises should keep in mind that LLMs are often trained on large corpora of Internet texts, which may make them prone to generating toxic, biased, and otherwise inappropriate and harmful content.

Enterprises that want to build proprietary LLMs from scratch also need to address additional challenges, like whether they have sufficient computing power, storage, and datasets, expertise, and financial resources to develop, implement, and maintain the models.

Ways to Build LLMs

Building large language models from scratch doesn't always make sense, especially for enterprises whose core business is not related to AI or NLP technologies. Since the process can be extremely time-consuming and resource-exhaustive, most enterprises are more likely to opt for customizing existing models to their needs.

Customizing existing base models – also called pre-trained models or PLMs – can be typically split into three essential steps:

1. **Finding a well-suited foundation model (PLM).** This requires considering ideal model size, training tasks and datasets, LLM providers, and more.

2. **Fine-tuning the model.** Base models can be fine-tuned on a specific corpus and for a specific use case. For example, text classification base models may be fine-tuned for sentiment analysis or trained using legal records to become proficient in legal terminology.

3. **Optimizing the model.** Models can be further optimized using techniques such as Reinforcement Learning from Human Feedback (RLHF), where the model is updated based on positive or negative human feedback on its predictions or classifications. RLHF seems particularly promising, partially due to being used in widely popular ChatGPT.

Alternatively, enterprises may choose to *only* customize base models using parameter-efficient techniques like adapters and p-tuning. Customization can yield especially accurate models when the base model is trained on tasks similar to the selected downstream tasks. For example, a base text classification model may be a good candidate for customization for sentiment analysis, as the two tasks are very similar. Thanks to being trained on text classification, the model can draw upon the knowledge it gained during training to perform sentiment analysis tasks more easily.

How to Evaluate LLMs

Large language models (LLMs) use deep learning techniques to analyze and generate natural language. These models have become increasingly popular due to their ability to perform a wide range of language-related tasks such as language translation, text summarization, and question-answering. However, evaluating the performance of LLMs is not a straightforward task, and it requires a careful analysis of different factors such as training data, model size, and speed of inference.

The most crucial element in evaluating LLMs is the quality and quantity of the training data used. The training data should be diverse and representative of the target language and domain to ensure that the LLM can learn and generalize language patterns effectively. Moreover, the training data should be annotated with relevant labels or tags to enable supervised learning, which is the most used approach in LLMs.

Another important factor is the size of the model. Generally, larger models have better performance, but they also require more computational resources to train and run. Therefore, researchers often use a trade-off between model size and performance, depending on the specific task and resources available. It is also worth noting that larger models tend to be more prone to overfitting, which can lead to poor generalization performance on new data.

Speed of inference must also be used in evaluation, especially when deploying LLMs in real-world applications. Faster inference time is desirable as it enables the LLM to process large amounts of data in a timely and efficient manner. Several techniques, such as pruning, quantization, and distillation, have been proposed to reduce the size and improve the speed of LLMs.

To evaluate the performance of LLMs, researchers often use benchmarks, which are standardized datasets and evaluation metrics for a particular language-related task. Benchmarks enable fair comparisons between different models and methods and help identify the strengths and weaknesses of LLMs. Common benchmarks include GLUE (General Language Understanding Evaluation), SuperGLUE, and CoQA (Conversational Question Answering).

Notable Companies in the LLM Field

The release of BERT in 2018 and, more notably, the release of GPT-3 in 2020, prompted both large tech companies and smaller startups to enter the race with their own LLMs and innovative approaches to model development. The most notable companies developing their own LLMs at the time of publication are listed in Table 3.

Table 3. Notable Companies Developing LLMs

Company	LLM
OpenAI	GPT-3 Davinci (175B)
AI21 Labs	Jurassic-1-Jumbo (178B)
EleutherAI	GPT-NeoX (20B)
Anthropic	Anthropic-LM (52B)
Cohere	Cohere xlarge v20220609 (52.4B)
NVIDIA/Microsoft	Megatron-Turing Natural Language Generation (MT-NLG 530B)
Microsoft	Turing Natural Language Generation (T-NLG 17B)
Google	Pathways Language Model (PaLM 540B)
Meta	Open Pretrained Transformer (OPT-175B)

Some of these companies offer other organizations access to their models. For example, enterprises can customize pre-trained modes developed by OpenAI, Cohere, or NVIDIA for downstream tasks or integrate them into their products and internal systems via API.

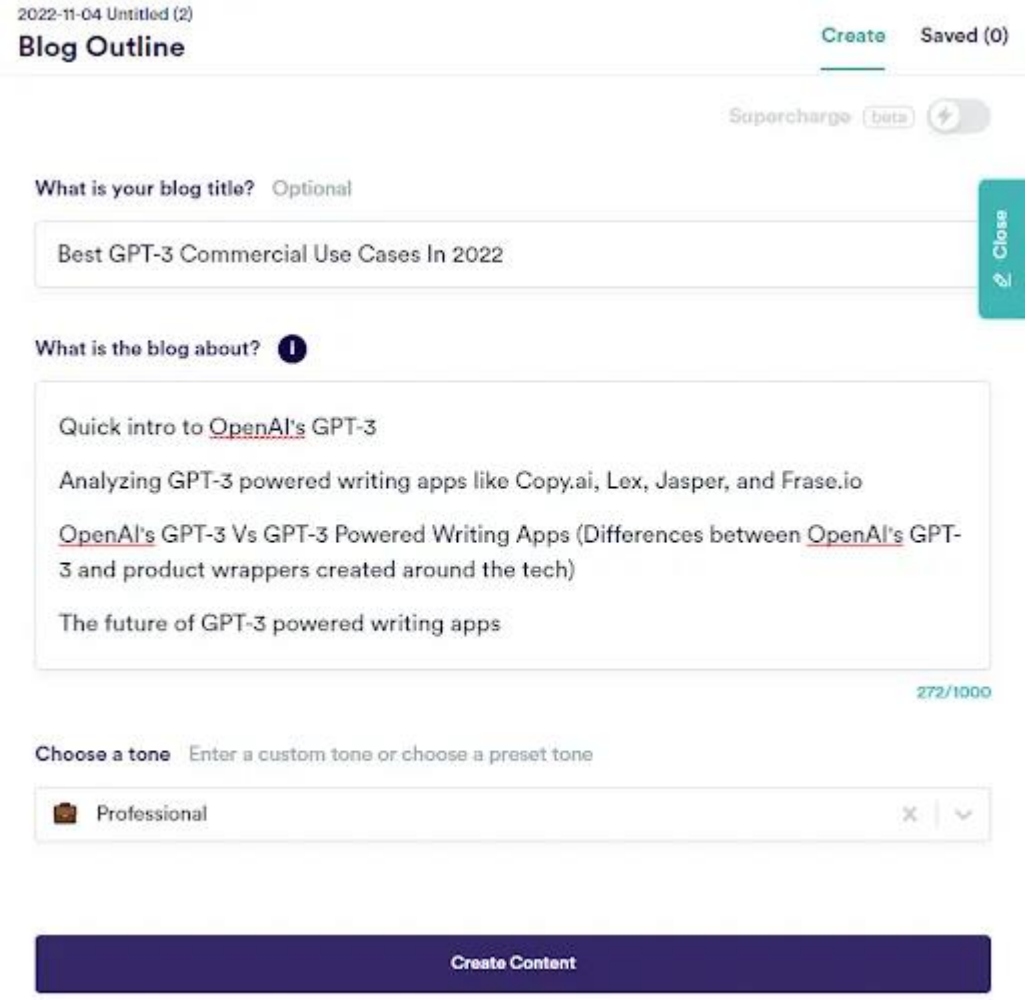
Popular Startup-developed LLM Apps

OpenAI's ChatGPT is by far the most popular LLM-powered app developed to date. It's estimated that it attracted over 100 million users in just two months after its launch, making it the "fastest-growing consumer application in history."

However, many other startups entered the ring with their own, often more specialized and commercialized LLM-powered apps. One of the most popular such apps are LLM-powered content generators like Jasper and Copy.ai. For comparison, Jasper boasts catering to over 100,000 global teams, while Copy.ai claims it has attracted over 5,000,000 users since its launch.

Figure 4 shows an example of a natural-language prompt that users can enter in to Copy.ai to generate a blog post outline.

Figure 4. Example of Results of a Natural Language Prompt



Other examples of popular LLM-powered apps include the beloved grammar-checking and writing tool, Grammarly, and GitHub Copilot, a Codex-powered coding assistant that can help developers write and learn code.

Part 2 will cover more ways in which enterprises and startups can leverage LLMs to build specialized apps for content generation, anomaly detection, toxicity classification, and other advanced NLP use cases. It will also provide concrete examples of how they can be further customized to answer the needs of various industries, such as finance, healthcare, and telecommunications, in hopes of inspiring organizations to use LLMs to unlock new possibilities in their respective industries.

Notice

This document is provided for information purposes only and shall not be regarded as a warranty of a certain functionality, condition, or quality of a product. NVIDIA Corporation (“NVIDIA”) makes no representations or warranties, expressed or implied, as to the accuracy or completeness of the information contained in this document and assumes no responsibility for any errors contained herein. NVIDIA shall have no liability for the consequences or use of such information or for any infringement of patents or other rights of third parties that may result from its use. This document is not a commitment to develop, release, or deliver any Material (defined below), code, or functionality.

NVIDIA reserves the right to make corrections, modifications, enhancements, improvements, and any other changes to this document, at any time without notice.

Customer should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

NVIDIA products are sold subject to the NVIDIA standard terms and conditions of sale supplied at the time of order acknowledgement, unless otherwise agreed in an individual sales agreement signed by authorized representatives of NVIDIA and customer (“Terms of Sale”). NVIDIA hereby expressly objects to applying any customer general terms and conditions with regards to the purchase of the NVIDIA product referenced in this document. No contractual obligations are formed either directly or indirectly by this document.

NVIDIA products are not designed, authorized, or warranted to be suitable for use in medical, military, aircraft, space, or life support equipment, nor in applications where failure or malfunction of the NVIDIA product can reasonably be expected to result in personal injury, death, or property or environmental damage. NVIDIA accepts no liability for inclusion and/or use of NVIDIA products in such equipment or applications and therefore such inclusion and/or use is at customer’s own risk.

NVIDIA makes no representation or warranty that products based on this document will be suitable for any specified use. Testing of all parameters of each product is not necessarily performed by NVIDIA. It is customer’s sole responsibility to evaluate and determine the applicability of any information contained in this document, ensure the product is suitable and fit for the application planned by customer, and perform the necessary testing for the application in order to avoid a default of the application or the product. Weaknesses in customer’s product designs may affect the quality and reliability of the NVIDIA product and may result in additional or different conditions and/or requirements beyond those contained in this document. NVIDIA accepts no liability related to any default, damage, costs, or problem which may be based on or attributable to: (i) the use of the NVIDIA product in any manner that is contrary to this document or (ii) customer product designs.

No license, either expressed or implied, is granted under any NVIDIA patent right, copyright, or other NVIDIA intellectual property right under this document. Information published by NVIDIA regarding third-party products or services does not constitute a license from NVIDIA to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property rights of the third party, or a license from NVIDIA under the patents or other intellectual property rights of NVIDIA.

Reproduction of information in this document is permissible only if approved in advance by NVIDIA in writing, reproduced without alteration and in full compliance with all applicable export laws and regulations, and accompanied by all associated conditions, limitations, and notices.

THIS DOCUMENT AND ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, “MATERIALS”) ARE BEING PROVIDED “AS IS.” NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL NVIDIA BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Notwithstanding any damages that customer might incur for any reason whatsoever, NVIDIA’s aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms of Sale for the product.

Trademarks

NVIDIA, the NVIDIA logo are trademarks and/or registered trademarks of NVIDIA Corporation in the U.S. and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

Copyright

NVIDIA Corporation | 2788 San Tomas Expressway, Santa Clara, CA 95051
http://www.nvidia.com