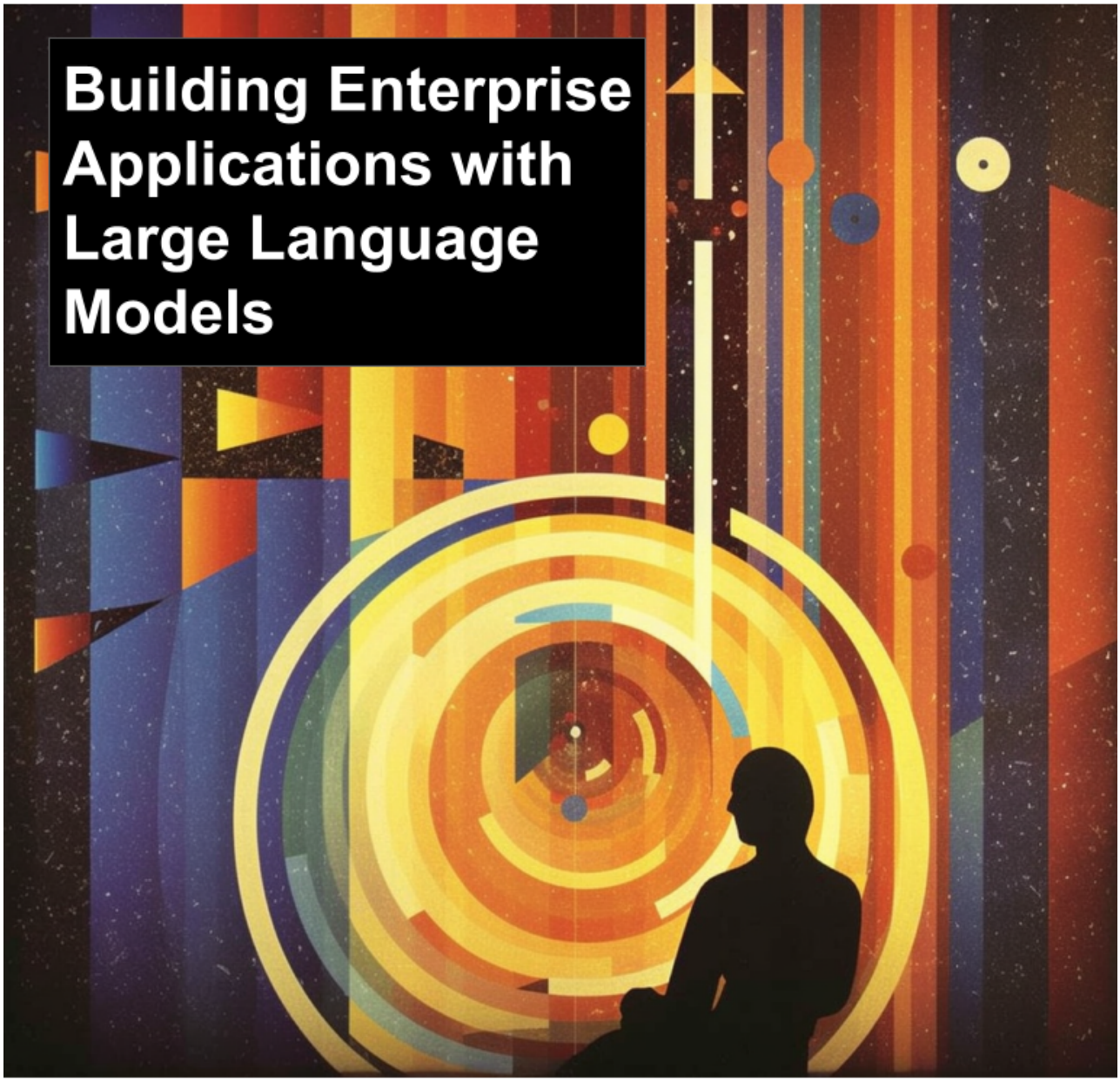

Part 1: An Introduction to

Large Language Models

Author: Josh Patterson
2023 Vol 1

Building Enterprise Applications with Large Language Models

Part 1 - An Introduction to Large Language Models (LLMs)

Josh Patterson

Contents

Introduction

Purpose of this series:

> To understand the role of large language models in enterprise applications and some of the components, tools, and technologies relevant to building these applications.

This space is moving fast and it's valuable to get a mental framework on how to think about LLMs to better understand how to apply them in your projects and organization. This series also sets context for a better understanding of new developments in LLMs. Our technical series are based on private reports we produce for our enterprise customers and other entities we advise as a company.

The intended audience for this series is:

> Individual practitioners, enterprise data teams, and enterprise architects

Series:

- An Introduction to Large Language Models (LLMs)
- Appendix A: What is Artificial Intelligence?

Appendix A gives context on the arena of artificial intelligence based on the history of the field. This context and history gives the reader a better viewpoint on recent developments in large language models.

What are Large Language Models (LLMs)?

Large language models, such as GPT-3.5, are advanced artificial intelligence systems designed to understand and generate human-like text. These models are trained on vast amounts of data to learn the patterns, structures, and semantics of language. They can then generate coherent and contextually relevant responses to various prompts.

Training a large language model involves utilizing a massive dataset, which often includes a wide range of texts from books, articles, websites, and other written sources. This diverse corpus helps the model learn the nuances of human language and develop a broad understanding of various topics.

Once trained, large language models can be used for a wide range of applications. They can:

- generate human-like text
- answer questions
- assist with language translation
- write code
- summarize articles
- create conversational agents

and much more. They achieve this by leveraging the knowledge and patterns they have learned during training.

It's important to note that while large language models like GPT-3.5 can generate impressive and coherent text, they don't possess true understanding or consciousness. They are statistical models that rely on patterns and associations in the training data rather than true comprehension. LLMs are good at understanding written language in the form of plain text input. LLMs output plain text based on the plain text input and have no memory of previous conversations between the context provided in the input text.

Recently it has been declared that large language models are having their "stable diffusion" moment, in that an audience beyond the research community could see that they were useful in many tasks beyond their initial focus.

To get a better idea of the key topics in large language models, let's take a look at some key terminology in the field.

Terminology in Large Language Models (LLMs)

There are a lot of research topics that I can mention in the world of LLMs but for the purpose of this series I'll focus on the following concepts:

- Prompt
- Prompt Engineering
- In-Context Learning
- Embeddings

- Vector databases

Let's now jump into prompts and prompt engineering.

Prompts

Prompts are the primary means by which a user leverages LLMs, thus engineering prompts (prompt engineering) is key to fully utilize the strengths of LLMs. A prompt can be a question, a statement, or an incomplete sentence that sets the context for the model to generate a coherent and relevant output.

Prompts direct generation/retrieval of responses/text. Prompts can be decomposed into four components, which aren't strictly necessary for every prompt. Instructions tie the other pieces together, directing how each of them should be used. External info/contexts are additional sources of knowledge. It's important to note that contexts do not have to be explicitly provided, and can be retrieved/referenced. User input/query is self explanatory. Output indicators mark where generated text begins.

When providing a prompt, users specify their desired outcome or request, and the model generates a response based on the patterns and information it has learned during training. The prompt serves as a guide for the model to understand the user's intent and generate a suitable response accordingly.

Prompt Engineering

Prompt engineering, which involves optimizing the wording and structure of prompts, has become an important practice to improve the quality and reliability of model-generated responses. Researchers and developers often experiment with different prompt formulations to achieve the desired outcomes and mitigate potential biases or limitations of the language model.

Considerations to keep in mind while engineering prompts

- Understanding the task/domain is crucial to leveraging an LLM.
- Any bias in the input will likely be amplified in the output, especially since LLMs are not immune from bias.

- Good prompts should be consistent, even when variations are made from them. A common practice seems to be testing the response to many slight variations in order to uncover issues.
- Iteration is important, thus testing and feedback can often be vital to arriving at desired outcomes.

The effectiveness of a prompt depends on its clarity, specificity, and relevance to the desired task. Well-crafted prompts help elicit accurate and meaningful responses from the model, while ambiguous or poorly phrased prompts may yield inaccurate or undesired outputs.

More resources on prompts and prompt engineering:

- https://exchange.scale.com/public/events/llm-prompt-engineering-and-rlhf-history-and-techniques-2023-03-09
- https://lilianweng.github.io/posts/2023-03-15-prompt-engineering/
- https://github.com/dair-ai/Prompt-Engineering-Guide

In-Context Learning

In-context learning refers to a technique used with large language models to fine-tune or adapt them to specific tasks or domains by providing additional training on task-specific examples or data. This process allows the model to specialize and improve its performance on specific tasks or to better align with the requirements of a particular application.

In large language models like GPT-3.5, the initial training involves exposing the model to a diverse range of text from various sources. However, this general training may not be sufficient for certain specific tasks or domains. In-context learning addresses this limitation by fine-tuning the model with additional examples or data that are relevant to the target task.

In-context learning is useful if we don't have direct access to the model or we don't have the means to retrain the model.

Embeddings

Embeddings are a fundamental concept in machine learning models, including Large Language Models (LLMs). They are numerical representations of data, such as text, images, or audio, that capture the essence or semantic meaning of the data. The process of embedding involves converting the input data into vectors of numbers, allowing the machine learning model to understand and process the data effectively.

By mapping data into a continuous vector space, embeddings allow for similarity comparisons (through the use of algorithms like nearest neighbors) and the clustering of textual data.

Prompts are used in conjunction with vector databases and embeddings. Effective prompts can include specific instructions that will retrieve relevant data from vector databases. To optimize this, prompts must be constructed with consideration given to embedding space and the relationships between embeddings.

Vector Databases

Vector databases play a crucial role in machine learning (ML) and large language model (LLM) applications. They are designed to efficiently store, index, and retrieve high-dimensional vectors, which are numerical representations of data points. These vectors can represent various entities such as words, documents, images, or user preferences. Vector databases work hand in hand with the embedding models that create such high-dimensional vectors.

The importance of vector databases lies in the many utilities and operations it allows embeddings to perform once the semantic meaning of data is vectorized.

Deep Learning, Transformers, and the Evolution of Large Language Models

To give a better idea of how large language models got to this point, let's look at some recent history in the field of neural networks. Deep learning, transformers, and large language models are interconnected concepts that play crucial roles in the development and success of advanced natural language processing (NLP) systems.

Deep learning is a subfield of machine learning that focuses on training artificial neural networks with multiple layers to learn and extract complex patterns and representations from data. Deep learning algorithms, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), have revolutionized various fields, including computer vision, speech recognition, and natural language processing.

Before transformers, recurrent neural networks were commonly used as generative language models but they didn't have nearly as many abilities as today's large language models.

Transformers are a specific type of deep learning architecture that has significantly advanced NLP tasks. They were introduced in the seminal paper "Attention is All You Need" by Vaswani et al. in 2017. Transformers employ a self-attention mechanism to capture contextual dependencies between words in a sequence, allowing the model to weigh the importance of different words when generating responses. This attention mechanism helps transformers effectively model long range dependencies and improves their ability to understand and generate coherent text. A list of different transformer types can be found here and a wonderful visual explanation of transformers can be found here.

Large language models, such as GPT-3.5, are deep learning models built using transformer architectures specifically designed for language understanding and generation tasks. These models have been trained on massive amounts of text data, learning to predict the next word or sequence of words given a context. By leveraging the power of transformers and deep learning techniques, large language models can generate human-like text, answer questions, perform language translation, and assist with a wide range of natural language processing tasks.

The success of large language models is largely attributed to the transformer architecture's ability to capture long-range dependencies and learn contextual representations effectively. The deep learning techniques used in training these models allow them to understand complex patterns and relationships in language data, making them capable of generating coherent and contextually relevant text.

Deep Learning is also about using the correct architecture for a data type so that the architecture can perform automated feature engineering. Transformers are a Deep Learning architecture for natural language and use raw text as input and output. So it's worth noting that Transformer-based systems require no manual feature engineering for training.

Deep learning forms the foundation of large language models, and transformers provide the architecture that empowers these models with the ability to understand and generate natural language text. Together, they have revolutionized the field of NLP and enabled the development of sophisticated language understanding and generation systems.

Now that we've established the connection to deep learning and transformers, let's take a look at the architecture of a popular large language model, GPT.

The GPT Architecture

GPT-3 is a language model that creates near-human level quality text content. It comes in eight different sizes, ranging from 125 million to 175 billion parameters. The largest GPT-3 model is ten times bigger than the previous record holder, T5-11B. The smallest GPT-3 model is approximately the same size as BERT-Base and RoBERTa-Base.

All GPT-3 models follow the same attention-based architecture as their predecessor, GPT-2. The smallest GPT-3 model has 12 attention layers, each with 12 sets of 64-dimensional heads. In contrast, the largest GPT-3 model has 96 attention layers, each with 96 sets of 128-dimensional heads.

GPT-3 achieved a substantial increase in capacity compared to GPT-2, expanding it by three orders of magnitude. This was accomplished through the addition of more layers, wider layers, and utilizing more training data.

However, it's important to note the immense computational requirements of training the largest GPT-3 model. The 175 billion parameter model necessitated 3.14E23 floating point operations per second (FLOPS) during training. Even with the most efficient hardware, such as the V100 GPU with a theoretical peak performance of 28 TFLOPS, it would take 355 GPU-years and cost approximately $4.6 million for a single training run. Similarly, using a single RTX 8000 GPU with an assumed performance of 15 TFLOPS, the training process would take approximately 665 years. These numbers provide perspective on the significant computational resources needed for training the largest GPT-3 model.

ChatGPT

ChatGPT is based on the GPT-3.5 architecture, which is part of the GPT (Generative Pre-trained Transformer) series developed by OpenAI. ChatGPT is designed to generate human-like text responses and engage in conversations on a wide range of topics. It has been trained on a diverse corpus of text data to

acquire knowledge and language patterns, enabling it to understand and generate coherent and contextually relevant responses.

Reinforcement Learning with Human Feedback (RLHF), a method that uses human demonstrations and preference comparisons to guide the model toward desired behavior, optimizes ChatGPT for dialogue. ChatGPT can be used for various purposes, including answering questions, providing explanations, assisting with creative writing, and engaging in interactive dialogues.

Understanding LLM Abilities and How to Measure Them

Any new technology that wows us is always fun to play with, but if we hope to build any applications of value, we need to put a fine edge on the 1-2 things the technology does well. When deep learning roared onto the tech scene, similarly many pundits and The Twitterati ascribed to it magical powers. The market had to come back to its senses and break down what we could realistically expect from deep learning (image object detection, generative models, etc).

In this section I will lay out the abilities and limitations of LLMs and then forecast where I think they'll be of most value in enterprise applications. For a broader view on the topic of artificial intelligence, I've also provided an appendix from our O'Reilly book ("Deep Learning: A Practitioner's Approach") (Thanks O'Reilly). With that, let's jump into the abilities of large language models, focusing on GPT-3.

GPT-3 Abilities and Evolution

It's worth listing some of the core abilities we see large language models commonly performing:

- Reasoning about instructions and context in the input (the "prompt")
- Creating plans of actions, and then execute each step of the plan with further calls back to the LLM model
- Integrating with external tools (search engines, Wikipedia, Python interpreters, etc.) to try out commands and examine the input in a later LLM chained call

Large language models are not running code for themselves (yet?) or creating their own computing environments.

However, one of the more compelling things large language models have shown is the ability to learn a task from only a few examples shown in the prompt itself, as described in the paper ("Can Foundation Models Wrangle Your Data?")[https://arxiv.org/abs/2205.09911]:

> Emergent Behaviors Interestingly, the biggest GPT-3 variant (175B parameters) has the capacity to solve natural language tasks with only a few examples (called few-shot prompting), and in some cases, just a task description (e.g. "Translate French to English"). Unlike traditional finetuning, no model parameters are updated to fit the task. Few-shot prompting has proven to be effective on tasks widely different from the FMs pretraining objective. Some examples include code generation [84], Trivia QA [18, 48] and common sense reasoning tasks [18]. Smaller models (less than 10B parameters) typically require some form of task-specific finetuning to perform well.

There has been much writing on the topic of "emergent abilities" in large language models (for and against the idea), but researcher [Jason Wei does a great job listing out some of the specific emergent abilities on his blog]](https://www.jasonwei.net/blog/emergence).

Beyond listing abilities, Yao Fu goes further and focuses his analysis on the 3 important abilities that the initial GPT-3 exhibit:

- **Language generation**: to follow a prompt and then generate a completion of the given prompt.
- **In-context learning**: to follow a few examples of a given task and then generate the solution for a new test case.
- **World knowledge**: including factual knowledge and commonsense.

For those inclined in the details, Fu's article is wonderfully detailed and contains considerable insight into the evolution of the GPT-series of LLMs.

> It is interesting to note that, although being a language model, the original GPT-3 paper barely talks about "language modeling" — the authors devoted their writing efforts to their visions of in-context learning, which is the real focus of GPT-3.

Fu goes on to describe the origin of the abilites of GPT-3:

> Generally, the above three abilities should come from large-scale pretraining — to pretrain the 175B parameters model on 300B tokens

Further breaking down the source of the training data:

- 60% 2016 - 2019 Common Crawl
- 22% WebText2
- 16% Books
- 3% Wikipedia

Fu goes on to further hypothesize where specific abilites in GPT-3 come from:

- The **language generation** ability comes from the language modeling **training objective**.
- The **world knowledge** comes from the 300B token **training corpora** (or where else it could be).
- The **175B model size** is for **storing knowledge**, which is further evidenced by Liang et al. (2022), who conclude that the performance on tasks requiring knowledge correlates with model size.
- The source of the **in-context learning** ability, as well as its generalization behavior, **is still elusive**. Intuitively, this ability may come from the fact that data points of the same task are ordered sequentially in the same batch during pretraining. Yet there is little study on why language model pretraining induces in-context learning, and why in-context learning behaves so differently than fine-tuning.

For reference, A 750 word document will be about 1000 tokens. If we say the average book has 80,000 words in it, and 2 million tokens is roughly the equivalent to 1.5 million words, we can calculate the total books GPT-3 was trained on to be **around 2.8 million books** based on a 300 billion token training corpora.

For comparison, the average person might read around 700 books in their lifetime.

The entire series is wonderful for insight into how large language models such as GPT-3 get certain types of "abilities" such as "code generation" and "in-context learning".

Key point of Section: "In-context learning is a big deal, and doesn't require retraining the model"

To put the above in perspective, let's review the core ideas of knowledge and reasoning:

- Knowledge provides the foundation and raw material for reasoning. It serves as a basis for making informed decisions, solving problems, and understanding the world. Knowledge provides context and background information that can be used in the reasoning process.
- Reasoning employs logical and cognitive processes to analyze and manipulate knowledge. It helps in organizing, connecting, and utilizing knowledge to derive new insights, solve problems, make judgments, and make predictions.

Knowledge is a prerequisite for effective reasoning. Reasoning heavily relies on the availability and accuracy of relevant knowledge. Lack of knowledge can limit the quality and accuracy of reasoning. However, reasoning can operate even in the absence of complete knowledge. It can make use of partial or incomplete information to draw conclusions or make decisions. Reasoning can also be employed to fill gaps in knowledge and acquire new knowledge.

Knowledge and reasoning are interrelated cognitive processes. Knowledge provides the information and foundation for reasoning, while reasoning employs logical and cognitive processes to manipulate knowledge and arrive at conclusions. Knowledge represents accumulated information, while reasoning is a dynamic process of drawing inferences and making decisions based on available information.

If we take those ideas, and then consider that Yao Fu mentions:

> The two important but different abilities of GPT-3.5 are **knowledge** and **reasoning**.

He further concludes that:

> Generally, it would be ideal if we could **offload the knowledge part to the outside retrieval system and let the language model only focus on reasoning.**

allowing for the advantages of using more up-to-date knowledge to answer questions using an outside retrieval system. It's also worth noting that a lot of the GPT-3 175B parameter model was used for storing knowledge, we could potentially use much smaller models coupled with an external knowledge system (e.g., data warehouse) and have models that are much smaller but still with effective reasoning capabilities.

The added benefit of outsourcing knowledge for LLM applications is that the knowledge within LLMs is unreliable and cannot be verified. However, knowledge from databases and search engines has more capacity and you can easily verify credibility of search results by checking the source.

The simple conclusion to draw from this section is:

- Databases are already good at knowledge
- We like LLMs for their reasoning abilities

These early ideas begin to inform what large language model application architectures could look like, which I explore further in part 2 of this series. Now, let's take a look at some of the early known limitations of large language models.

Limitations of GPT-3

Some of the things GPT-3 and large language models in general struggle with are:

- They cannot on-the-fly overwrite the model's beliefs
- Formal reasoning
- Biases and unfairness
- Lack of common sense
- Contextual misunderstandings
- Sensitivity to input phrasing

From an application prototyping standpoint, the issues "sensitivity to input phrasing" and "outdated knowledge" come up early. The phrasing or wording of a prompt can significantly impact the generated response. Small changes in the input can lead to different outputs, making the models less reliable for critical or sensitive tasks. Sometimes this needs to be tested thoroughly before letting folks loose on a new application. Language models have a knowledge cutoff, meaning they are not aware of events or information that occurred after their training data. Therefore, they may provide outdated or incorrect information about recent events. These are just two of the early issues our team has hit developing LLM application prototypes.

Meauring LLM Reasoning Ability

If we focus our applications on leveraging a large language model's ability to reason, and we're to try to offload some knowledge requirements to an outside system to use smaller models, then it makes sense to understand how to measure the relative reasoning ability of different models.

For some great background reading on the type types of reading, check out Google's article on Types of reasoning in LLMs.

Further, two key LLM reasoning benchmarking resources are:

1. The Chain of Though Hub (background: paper)
2. The Weight Watcher Leaderboard

Each ranking site uses a set of tests to measure how well each model can do different types of reasoning tasks. Some tests (e.g., MMLU) are used by both ranking sites.

For example, the Chain of Thought Hub complex reasoning tasks:

- math (GSM8K)
- science (MATH, TheoremQA)
- symbolic (BBH)
- knowledge (MMLU, C-Eval)
- coding (HumanEval)
- factual (SummEdits)

So based on what kind of reasoning problem you were trying to solve, these individual benchmarks would help you understand what tradeoffs you could make in terms of parameter size vs reasoning ability. For example, if you had to run a model that was under 10B parameters, and the application called for math and science heavy reasoning, these benchmarks would help you filter and rank which models would be most effective for your criteria.

So now let's close out part 1 of this series by looking at the value of large language models and why there is so much interest in the space.

Why Are Large Language Models Compelling?

Large language models are compelling because they are what I call a "tectonic-plate shift"class of technology; These technologies, which introduced, shift the landscape of multiple other technologies and industries because they change:

1. what is possible with other technologies
2. the cost of certain tasks
3. the speed at which tasks can be performed

Examples of previous tectonic-plate shift technologies:

- databases
- big data platforms
- cloud platforms
- deep learning

Being able to embed applied reasoning into applications will change how tasks are accomplished in every industry.

From my analysis of large language model research and early technology, the 3 themes I see as most compelling right now in the space are:

1. you can create a natural language interface for any application with an API
2. you don't have to re-train models to create specific applications
3. knowledge work is beginning an evolution similar to the introduction of the textile loom in the 1800s

In the sections below, I lightly touch on each of these themes.

Natural Language as a Driver for Any Application

In prototyping LLM applications the thing that immediately struck me was that "ChatGPT was a nice demo, but there is a lot more here." After seeing the raw reasoning ability of LLMs up close, and then realizing that frameworks such as LangChain allowed you to "glue stuff together with LLMs" (effectively), my imagination ran wild.

Pretty soon we had all sorts of APIs wired up to LLM natural language input, and we could see how the LLM was able to reason about what sort of parameters should be sent to an API as well. This allows you to "speak" to an application in ways that just are not possible with classical "keyword" parsing methods. Once I saw these effects first-hand in our prototypes, I began to think about LLMs as a tectonic-plate-shift-class technology.

You Don't Have to Re-Train the Foundation Models to Do Things

In class machine learning, you learned quickly that you had to collect training data for your domain-specific application and re-train a new model to apply machine learning in your domain. It took a good decade for enterprises to come to fully appreciate this fact.

And then large language models come along and introduce In-Context Learning, and all of that gets turned on its head.

It's quickly evident in experimenting with LLM applications that base models plus a framework for prompt engineering is flexible enough to solve many problems without ever considering the idea of re-training a new LLM model.

That was my second realization that this was a tectonic-plate-shift-class technology, as it has been so hard for enterprises to collect good data and get models re-trained. LLMs could be applied more similarly to traditional software engineering practices — not exactly, but closer than "having to re-train a new model every time" in traditional machine learning.

Once I could feel the tectonic plate shift coming, I began to ruminate on the coming evolution of knowledge work.

Large Language Models and the Evolution of Knowledge Work

People are unsettled by rapid advances in technology.

We've seen this historically with the introduction:

- the 1800s textile loom (and the Luddites)
- the steam engine
- the introduction of computers to banking

For reference, the Luddites:

> The Luddites were members of a 19th-century movement of English textile workers which opposed the use of certain types of cost-saving machinery, often by destroying the machines in clandestine raids. They protested against manufacturers who used machines in "a fraudulent and deceitful manner" to replace the skilled labour of workers and drive down wages by producing inferior goods. Members of the group referred to themselves as Luddites, self-described followers of "Ned Ludd", a legendary weaver whose name was used as a pseudonym in threatening letters to mill owners and government officials.

Technology shifts have a way of reverberating uncomfortably through business, employment, and culture. I saw it first-hand when I had multiple physicians I

know call me and start asking questions about "this new GPT thing" (and you know something has touched a nerve when the employment class that is used to playing god is concerned).

However, I reminded these physicians that certain luminaries have forecasted (in the past decade) that 'Radiologists will be out of business before long.' (because of deep learning).

Truckers, lawyers, and textile workers have all gotten similar treatment with past technology cycles. Yet we have more truck loads, lawsuits, and clothes being produced than ever.

Workers aren't being replaced by new technology, but work does change and sometimes whole industries even evolve.

For example: I do more with MidJourney in terms of graphic design output than I did previously and no jobs were lost (previously, I just would have less graphics work done). The work product improves and the rate at which I can produce new artwork for articles increases and that's generally how technology advances have driven change in the workforce.

Another example is the effect computers had on banks; Banks have existed for centuries, and their operations have evolved over time. In the past, before the arrival of the computer age, banks relied primarily on manual processes for record-keeping and transactions. This included using ledgers to record deposits, withdrawals, and other financial transactions by hand. Banks also used paper checks and cash for most transactions. These manual processes were time-consuming and prone to errors, but they were the norm for many years. As technology advanced, banks began to adopt computers and automation to streamline their operations and improve efficiency. Before computers arrived, many banks were only open 4 days a week because they needed downtime to do record-keeping reconciliation work, etc. With the introduction of computers, some pundits forecasted that bank workers could take the 5th workday off because computers would handle the reconciliation.

However, that's not what happened. Banks quickly realized, under competitive pressure, that they now had the option to be open 5 days per week, and they took advantage of that opportunity.

The Red Queen never lets up, it seems.

Summary

In this article I introduced the idea of large language models, their abilities, and ways to measure these abilities. I also hypothesized that enterprise applications would leverage the reasoning ability of LLMs and use existing databases and data warehouses for knowledge retrieval, allowing us to use smaller and more efficient language models.

In part 2 of this series, we'll look at some design patterns for building enterprise large language model applications.

If you have further questions about large language models, or want to talk directly about how Patterson Consulting can help your enterprise build large language model applications, email me at josh@pattersonconsultingtn.com.

Thanks for reading.

References

Fu, Yao; Peng, Hao and Khot, Tushar. (Dec 2022). How does GPT Obtain its Ability? Tracing Emergent Abilities of Language Models to their Sources. Yao Fu's Notion.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is All you Need. Neural Information Processing Systems, 30, 5998–6008.