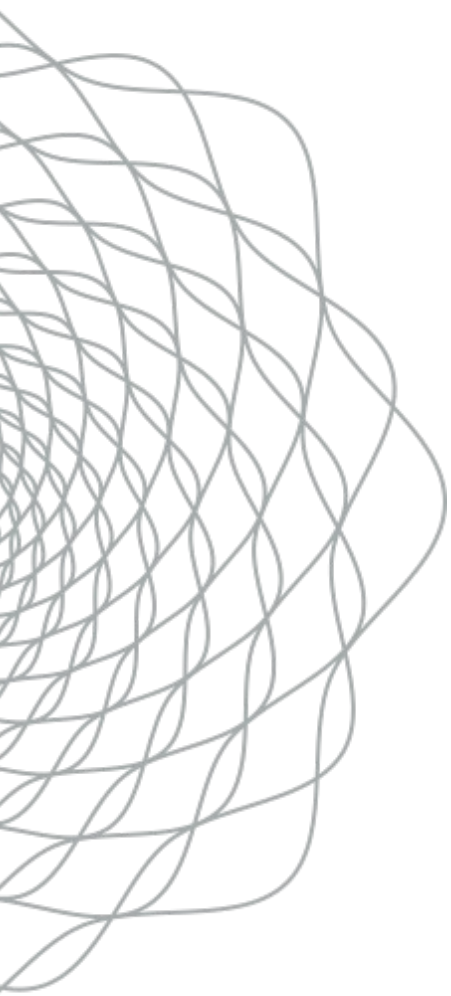


TARPON HEALTH

a healthcare automation community

LLMs + Prompt Engineering

Agenda

1. LLM Basics
2. Prompt engineering
    a. Parameter settings
    b. Templates, Prompts, Examples
    c. Referencing Data

Most of the content from this presentation was taken from Microsoft Azure's OpenAI resources

https://learn.microsoft.com/en-us/azure/ai-services/openai/

AI hierarchy

AI → Generative AI → Large Language Model → Natural Language Processing + Machine Learning

Artificial Intelligence (AI) imitates human behavior to interact with the environment and execute tasks *without explicit directions on what to output*.

Generative AI describes a category of capabilities within AI *that create original content*.

Generative AI models are powered by **large language models (LLMs)**, which use *a specialized type of **machine learning (ML)** model that you can use to perform **natural language processing (NLP)** tasks*. For example:

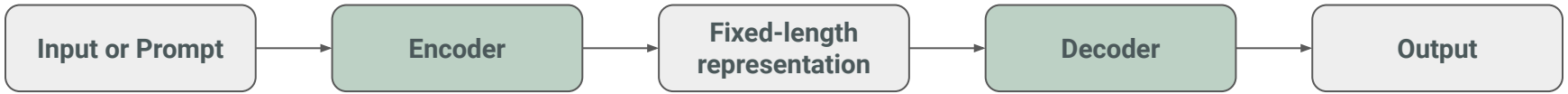
- Determining sentiment or otherwise classifying natural language text.
- Summarizing text.
- Generating new language.
- And more…

Transformer architecture

Most LLMs are based on “transformer” architecture

Transformer models are trained with large volumes of text, enabling them to represent the semantic relationships between words and use those relationships to determine ***probable sequences of text***.

Transformer model architecture consists of Encoder and Decoder blocks



“I love cats” (e.g., in english)

Processes the input sequence to create a fixed-length representation.

A fixed length representation is a condensed representation of the input that captures semantic meaning, context, and relevant information.

Takes the fixed-length representation and uses it to generate the output sequence (e.g., translated text). Decoders condition its predictions on both the fixed-length representation and previously generated tokens

Spanish translation: “Amo a los gatos”

A level deeper

Here is a basic mental model of how encoders and decoders work.

Breaking down language into component parts (e.g. tokens), so they can be used for mathematical analysis.

Plotting tokens numerically based on attributes to determine their proximity to each other. Each token has vector ‘coordinates’.

Layers of modeling to predict which token will come next based on relationships.

It also captures dependencies between tokens for coherent responses.

Tokenization

Tokens are the basic units into which input text is divided

Definition

The first step in training a model is to decompose the text into tokens (or a unique text value).

You can think of each distinct word in the training text as a token. In reality, tokens can be generated for partial words, or combinations of words and punctuation.

Example

“I heard a dog bark loudly at a cat”

To tokenize this text, you can identify each discrete word and assign token IDs to them.

- I (1)
- heard (2)
- a (3)
- dog (4)
- bark (5)
- loudly (6)
- at (7)
- ("a" is already tokenized as 3)
- cat (8)

The sentence can now be represented with the tokens: ***[1 2 3 4 5 6 7 3 8]***.

Similarly, the sentence "I heard a cat" could be represented as *[1 2 3 8]*.

Embedding

Defining the relationship between tokens

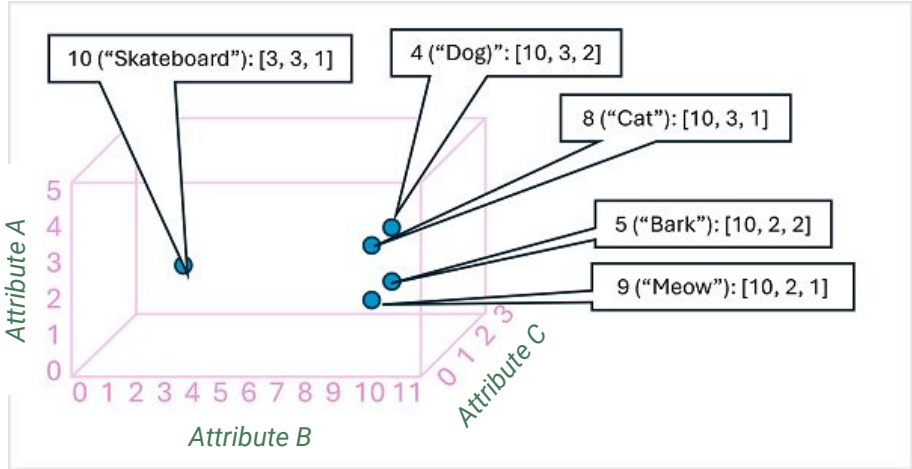
Definition

The next step is to define contextual vectors, known as embeddings. Vectors are multi-valued numeric representations of information. For example, “Cat” has the coordinates of [10, 3, 1] where each numeric element represents a particular attribute of the information.

Note: The attributes and # of dimensions depend on the model being trained.

Example

“Dog”, “Cat”, “Bark”, “Meow” are closer than “Skateboard”. Therefore they have a higher probability of showing up next to one another.



Sequencing

Multiple ‘layers’ of analysis are used to predict the sequence of events

Definition

Attention is a technique used to examine a sequence of text tokens and try to quantify the strength of the relationships between them. In particular, self-attention involves considering how other tokens around one particular token influence that token's meaning.

Example

“**Bark”** can have different meanings. "The **bark** of a tree" means something different than "I heard a dog **bark**."

For each token generated, the model has an attention layer that takes into account the sequence of tokens up to that point. The model considers which of the tokens are the most influential when considering what the next token should be.

The result is accurate predictions of what should come next…

“I heard a (_?_)”
“I heard a dog (_?_)”
“I heard a dog bark (_?_)”

Key takeaway

Large Language Models are ultimately statistical predictions.

The model and inputs you design will enhance that prediction.

Picking a model for the task

TARPON HEALTH

The first step is aligning the right LLM with the challenge at hand.

Models	Description
GPT 4	The latest generation of generative pretrained (GPT) models that can generate natural language and code completions based on natural language prompts.
GPT 3.5	Generate natural language and code completions based on natural language prompts. In particular, GPT-35-turbo models are optimized for chat-based interactions and work well in most generative AI scenarios.
DALL-E	A series of models that can generate original images from natural language.
Whisper	A series of models in preview that can transcribe and translate speech to text.
Text to speech	A series of models in preview that can synthesize text to speech.

Model Tuning vs. Prompt Engineering

Once you have a model, you'll have to decide whether you'll want to tune a model or just enhance your model with fine tuning.

Tuning (Custom)

You can build on an existing model as and train with your own data. This approach is called fine-tuning. It enables you to train a custom model that builds on the pre-trained model, but which is tuned to data that is relevant for your particular scenario.

For example, a legal firm might fine-tune a model with the text from existing contracts and other proprietary legal documents to train a model that is optimized for generating contractual content.

Prompt Engineering (Out-of-the box)

Prompts are ways we tell an application what we want it to do. An engineer can add instructions for the program with prompts.

For example, developers may build a generative AI application for teachers to create multiple-choice questions related to text students read. During the development of the application, developers can add other rules for what the program should do with the prompts it receives.

Parameters

Max Tokens: This parameter specifies the maximum number of tokens (words or subwords) in the generated text. It can help control the length of the output.

Temperature and Top P (Nucleus Sampling): This parameters control how deterministic the model is in generating a response. If you're asking for a response where there's only one right answer, you should specify lower values for these settings. If you're looking for a response that's not obvious, you might want to use higher values.

Frequency Penalty and Presence Penalty: These penalties can be used to discourage repetitive or factually inconsistent responses, respectively. They help in generating more coherent and accurate text.

Best Of: This parameter specifies the number of candidate sequences to generate and select the best one from. It can improve the quality of the generated text by allowing the model to explore different possibilities.

Stop Sequences: You can specify a list of sequences that, if generated, will stop the model from generating further text. This can be useful for controlling the direction of the conversation or preventing the generation of unwanted content.

Prompt engineering

The quality of responses depend greatly on prompt engineering.

System messages - Prompt engineering techniques include defining a system message. The message sets the context for the model by describing expectations and constraints, for example, "You're a helpful assistant that responds in a cheerful, friendly manner".

Writing good prompts - You can get the most useful completions by being explicit about the kind of response you want, for example, “Create a list of 10 things to do in Edinburgh during August”.

Examples - LLMs support “zero-shot” learning in which responses can be generated without prior examples. However, you can also provide one-shot learning prompts that include one, or a few, examples of the output you require.

Retrieval Augmented Generation (RAG) - Allows developers to use supported AI chat models that can reference specific sources of information to ground the response. Adding this information allows the model to reference both the specific data provided and its pretrained knowledge to provide more effective responses.

Revenue Cycle Idea

Medical Necessity Denials LLM use case

1. Medical Necessity denials received; the 835,837, and charge data is stored and observed by LLM
2. LLM is fed LCD/NCD & Payer coverage details
3. LLM is fed clinical documentation to language that correlates with the diagnosis code
4. LLM output is a decision, based on confidence ratings/positive reinforcement, that determines if a diagnosis exists to support an additional diagnosis that would meet medical necessity that exists on LCD/NCD documents
    a. CPT code billed with Dx
    b. X amount of Dx's exist on the clinical docs but were not added
    c. LCD/NCD scrape tell us we have the right Dx or missed one
    d. Queued up for RPA to recode/rebill or escalated in system for coding to correct without extensive review