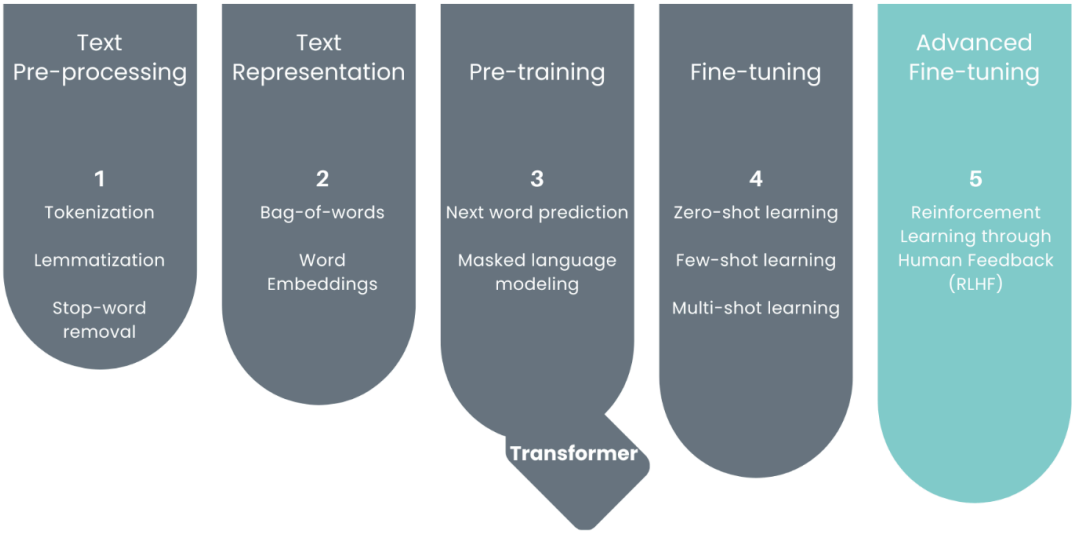
Deep Learning: LLM

Datacamp course: https://www.datacamp.com/courses/large-language-models-llms-concepts
Building Blocks of LLM



Text pre processing

- **Tokenization**: “Working with natural language processing techniques is tricky” → ["Working", "with", "natural", "language", "processing", "techniques", "is", "tricky", "."].
- **Lemmatization**: This process of reducing words to their base form is known as lemmatization. For example, "talking", "talked", and "talk" would be mapped to the root word "talk".
- **Stop Word Removal** - words, such as "with" or "is," are eliminated to identify the most important parts of the sentence.

Text Representation

- **Bag of words** - converting the text into a matrix of word counts without context/meaning
- **Word embedding** - semantic meanings of words and representing them as numbers, allowing for similar words to have similar representations

Pre Training

- **Next word predictions** - supervised learning to generate coherent text by capturing the dependencies between words in the larger context. During training, the model is presented with pairs of input and output examples.
- **Masked language modeling** - training a model to predict a masked word that is selectively hidden in a sentence.
- **Transformer -** multiple parts of the sentence simultaneously- ie preprocessing + positional placement + encoders (relationship between words) + Decoders (neural networks of larger concept)

Fine Tuning

- **Zero shot learning** - perform a task it has not been explicitly trained on. a child has only seen pictures of horses and is asked to identify a zebra with additional information that it looks like a striped horse.
- **Few shot learning** - to learn a new task with very few examples. (one shot is one example)
- **Multi shot learning** - uses the knowledge learned from previous tasks, along with more examples of the new task, to learn and generalize to new tasks - ie. identifying different dog breeds

Prompt Engineering: the art and science of designing and optimizing prompts to guide AI models, particularly LLMs, towards generating the desired responses. lays a vital role in ensuring accurate, relevant, and safe interactions. Think of it as providing a roadmap for the AI, steering it towards the specific output you have in mind.
https://cloud.google.com/discover/what-is-prompt-engineering#types-of-prompts

Fine Tuning vs. RAG
Most organizations currently don't train their own AI models. Instead, they customize pre-trained models to their specific needs, often using RAG or fine-tuning.

Fine-tuning requires adjusting a model's weights, which results in a highly customized model that excels at a specific task. It's a good option for organizations that rely on codebases written in a specialized language, especially if the language isn't well-represented in the model's original training data.

RAG, on the other hand, doesn't require weight adjustment. Instead, it retrieves and gathers information from a variety of data sources to augment a prompt, which results in an AI model generating a more contextually relevant response for the end user.

https://github.blog/ai-and-ml/generative-ai/what-is-retrieval-augmented-generation-and-what-does-it-do-for-generative-ai/

Advanced Fine Tuning
- **RLTHF** Reinforcement Learning Through Human Feedback: external expert to validate the data and avoid these inaccuracies.

Fine-tuning vs. Pre-training
Fine-tuning is more effective since it can help a model learn, or be trained, using a single CPU and GPU, while pre-training may require thousands of CPUs and GPUs to train efficiently. Additionally, fine-tuning can take hours or days, while training a model from scratch may take weeks or months. Furthermore, fine-tuning requires only a small amount of data, typically ranging from a few hundred megabytes to a few gigabytes, compared to hundreds of gigabytes as are necessary for pre-training.