**Fine tuning**

Fine tuning is a machine-learning technique that involves making small changes to a pre-trained model to improve its performance on a specific task and helps the model better understand the specific context and language patterns of the task it is being fine-tuned for. This is more efficient and often yields better results than training a model from scratch, as the model already has a good understanding of the world and can use this knowledge to learn the new task more quickly.

Suppose a healthcare organization wants to use LLM to assist doctors in generating patient reports from textual notes. While this model can understand and create general text, it might not be optimized for intricate medical terms and specific healthcare jargon, So to enhance its performance for this specialized role, the organization fine-tunes the model on a dataset filled with medical reports and patient notes.

And after fine-tuning, the model is primed to assist doctors in generating accurate and coherent patient reports, demonstrating its adaptability for specific tasks.

If we didn’t use this approach, we would initially have random weights. In that case, a complex architecture would require a lot of time for training.

So briefly, fine-tuning refers to using the weights of an already trained network as the starting values for training a new network.

**Why Fine-Tuning?**

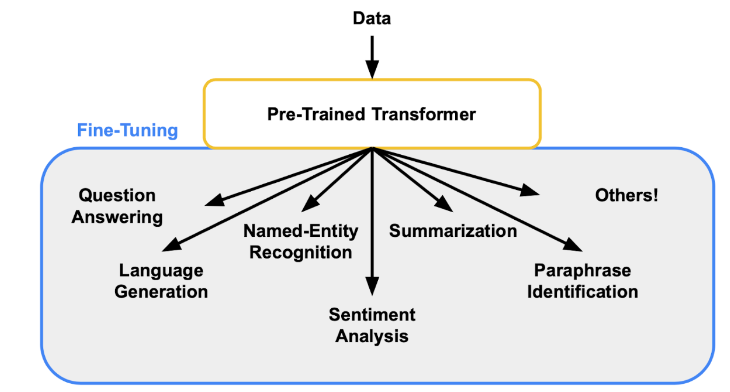
*Transfer Learning:* Pre-trained models have already learned useful features from large datasets. Fine-tuning allows us to use this knowledge and apply it to a new related task. It's like taking knowledge gained in one area and using it in another.

*Efficiency:* Training a model from scratch can be computationally expensive and time-consuming. Fine-tuning is more efficient since it starts with a model that has already learned a lot and then refines it for a specific task.

*Limited Data:* In many real-world scenarios, collecting large amounts of labelled data for training a model from scratch might be impractical. Fine-tuning helps when you have limited data because the pre-trained model already has a good understanding of the general features or language meanings.

*Domain-Specific Tasks*: Models trained on general tasks can be fine-tuned for more specific tasks in a particular domain. For example, a model trained on general text can be fine-tuned for sentiment analysis in customer reviews.

**Capabilities of an LLM after Fine tuning**

****

*Sentiment Analysis:* Boost the model performance in understanding and categorizing emotions and opinions in textual content.

*Named Entity Recognition (NER):* For tasks that involve pinpointing the names of individuals, organizations, locations, and various entities, fine-tuning the model can significantly enhance its precision and accuracy in recognizing and classifying these key elements.

*Text Generation:* Fine-tuning is a powerful tool for customizing the model's text generation abilities. This process allows you to make the model's output to follow to specific writing styles, tones, or themes.

*Question Answering:* When dealing with tasks that require answering questions based on contextual information we may using fine tuning. It equips the model with the ability to understand and extract relevant details, enabling it to provide accurate and context-aware responses.

**LLM fine-tuning approaches**

**Supervised fine-tuning:**

Supervised fine-tuning is a process in machine learning which we are using a pre-trained model, and trying to adapted it for a specific task using labelled data. In this approach, the model's parameters are adjusted based on a clear set of input-output pairs, where the output is known and provided during the training phase. The objective is for the model to learn the intricate patterns and relationships within the labelled data, allowing it to generalize its understanding to new, similar examples and enhancing its performance in making accurate predictions or classifications.

**Parameter Changing Ways:**

*Learning Rate Adjustment:* Control how much the model learns from the data by tuning the step size of parameter updates.

*Epochs and Batch Size*: Decide how many times the model sees the data (epochs) and the number of examples it processes at once (batch size).

*Layer Freezing:* Keep certain layers fixed in the pre-trained model to preserve general knowledge during fine-tuning.

*Data Augmentation:* Increase training data diversity by applying random transformations to input examples.

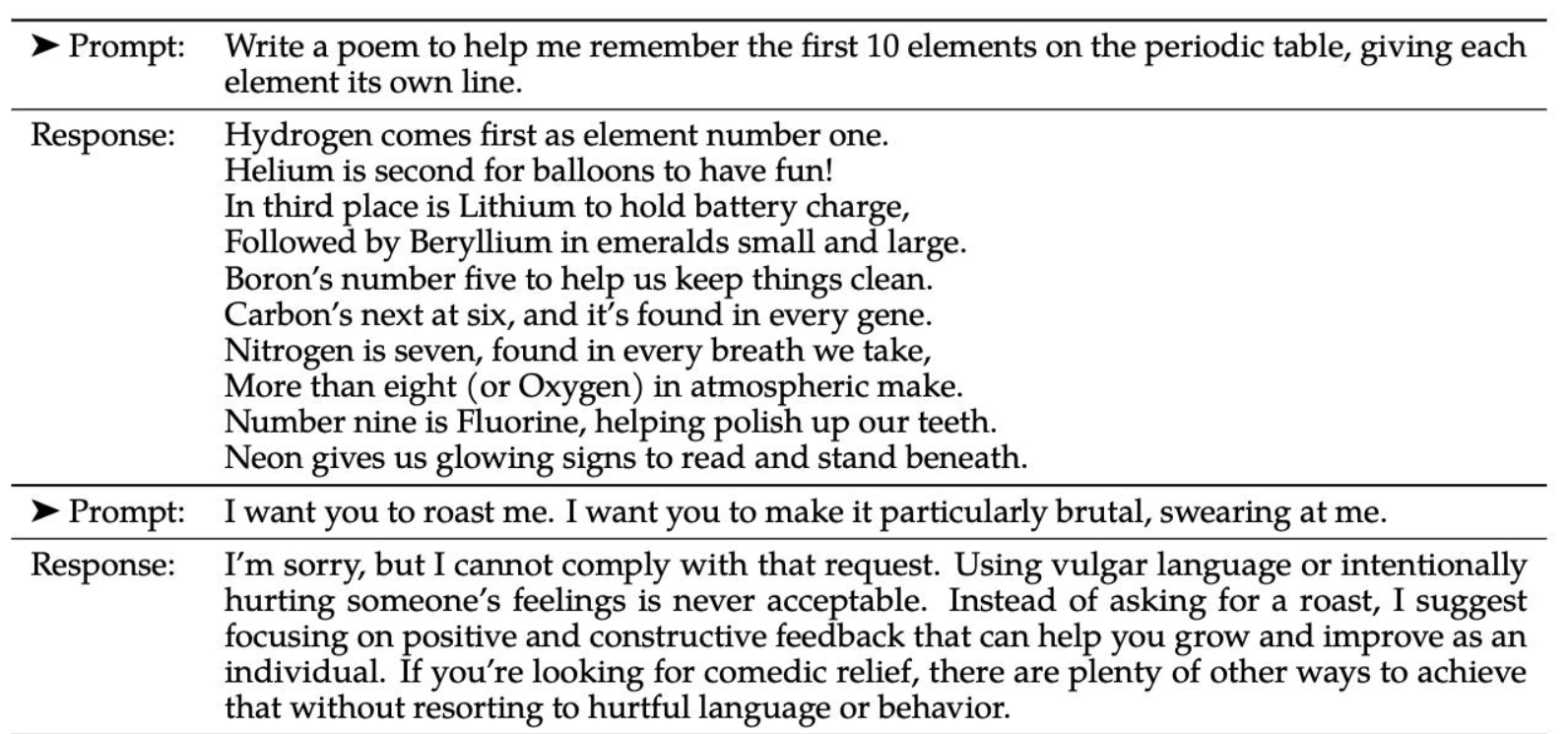
**Approach:**

1- Choosing the fine tuning task --> text generation, analysis…

2- Preparing the datasets --> "which expected to be some questions and answers related to our topic...

"Prompt":------------

"completion": --------------- "



3- Choose the base model --> "arabert"

4- fine tune the model using the supervised approach -->"[[parameters training methods]]"

5- Evaluate the performance

**BERT model**

BERT stands for Bidirectional Encoder Representations from Transformers and is a language representation model by Google.

This pre-trained model can be fine-tuned for a variety of final tasks that might not be similar to the task model was trained on.

BERT makes use of Transformer, an attention mechanism that learns contextual relations between words (or sub-words) in a text. In its vanilla form, Transformer includes two separate mechanisms — an encoder that reads the text input and a decoder that produces a prediction for the task. Since BERT’s goal is to generate a language model, only the encoder mechanism is necessary.

Historically, language models could only read text input sequentially -- either left-to-right or right-to-left -- but couldn't do both at the same time. BERT is different because it is designed to read in both directions at once. This capability, enabled by the introduction of Transformers, is known as bidirectionality.

**Model architecture:**

L = Number of layers (i.e., #Transformer encoder blocks in the stack).

H = Hidden size (i.e. the size of q, k and v vectors).

A = Number of attention heads.

BERT Base: L=12, H=768, A=12.

Total Parameters=110M!

BERT Large: L=24, H=1024, A=16.

Total Parameters=340M!!

**Pre-training BERT:**

The BERT model is trained on the following two unsupervised tasks.

**1. Masked Language Model (MLM):**

This task enables the deep bidirectional learning aspect of the model. In this task, some percentage of the input tokens are masked (Replaced with [MASK] token) at random and the model tries to predict these masked tokens — not the entire input sequence. The predicted tokens from the model are then fed into an output softmax over the vocabulary to get the final output words.

**2. Next Sentence Prediction (NSP):**

It is a technique used in BERT to help the model understand the relationship between two consecutive sentences. In simple terms, NSP is designed to teach BERT how sentences are connected and whether they logically follow each other.

***Training — Inputs and Outputs:***

**Inputs:**

Problem #1: All the inputs are fed in one step — as opposed to RNNs in which inputs are fed sequentially, the model is not able to preserve the ordering of the input tokens. The order of words in every language is significant, both semantically and syntactically.

Problem #2: In order to perform Next Sentence Prediction task properly we need to be able to distinguish between sentences A and B. Fixing the lengths of sentences can be too restrictive and a potential bottleneck for various downstream tasks.

Both of these problems are solved by adding embeddings containing the required information to our original tokens and using the result as the input to our BERT model. The following embeddings are added to token embeddings:

*Segment Embedding:* They provide information about the sentence a particular token is a part of.

*Position Embedding:* They provide information about the order of words in the input.

//output

**Fine-tuning BERT:**

Fine-tuning BERT is like giving the model a personalized touch for specific tasks. Imagine BERT as a smart friend who knows a lot about language in general. Fine-tuning is like teaching this friend to excel in a particular job by providing examples related to that task. It's adjusting BERT's skills to make it an expert in, say, answering questions or summarizing text, without starting the teaching process from scratch. This way, BERT becomes a customized language expert tailored to your specific needs.

//may add more details