Master’s Seminar Presentation

Mohammad Amin ASLAMI

ENHANCING LARGE LANGUAGE MODELS WITH RETRIEVAL-AUGMENTED GENERATION: TECHNIQUES, CHALLENGES, AND FUTURE DIRECTIONS

- Retrieval-Augmented Generation (RAG)
- LLMs Compress World Knowledge
- Problem with Text
- Text Embeddings Meaningful
- Text Embeddings & Text Embedding Meaningful
- Text Classification
- Large Language Models (LLMs)
- Next-Word Prediction Paradigm
- Fine-Tuning
- 3 Ways to Fine-tune
- The Problem (LLMs are computationally expensive)
- Full Finetuning, LoRA and QLoRA

Retrieval-Augmented Generation (RAG)

- Retrieval-Augmented Generation (RAG) is a powerful technique in Large Language Models (LLMs) that reduces errors or "hallucinations" by grounding responses in external data sources. By incorporating relevant context from external databases or knowledge bases, RAG significantly improves the cost-effectiveness of LLM operations. This is primarily due to the ease of updating retrieval indexes, which is far less resource-intensive than continuously fine-tuning pre-trained models. In addition, RAG provides streamlined access to current information, saving both time and money and enabling more efficient handling of current data in LLMs [1].

- One way we can mitigate both of these limitations is by using Retrieval-Augmented-Generation (RAG). Augmenting (complementing) LLM with specialized and mutable knowledge base. Basically, we can have a knowledge base that contains domain specific information that is updatable where we can add and remove information as needed. The typically way we will use LLMs is we will pass it a prompt and it will split out a response, the basic usage will rely on the internal knowledge of the model in generating the response based on the prompt [2].

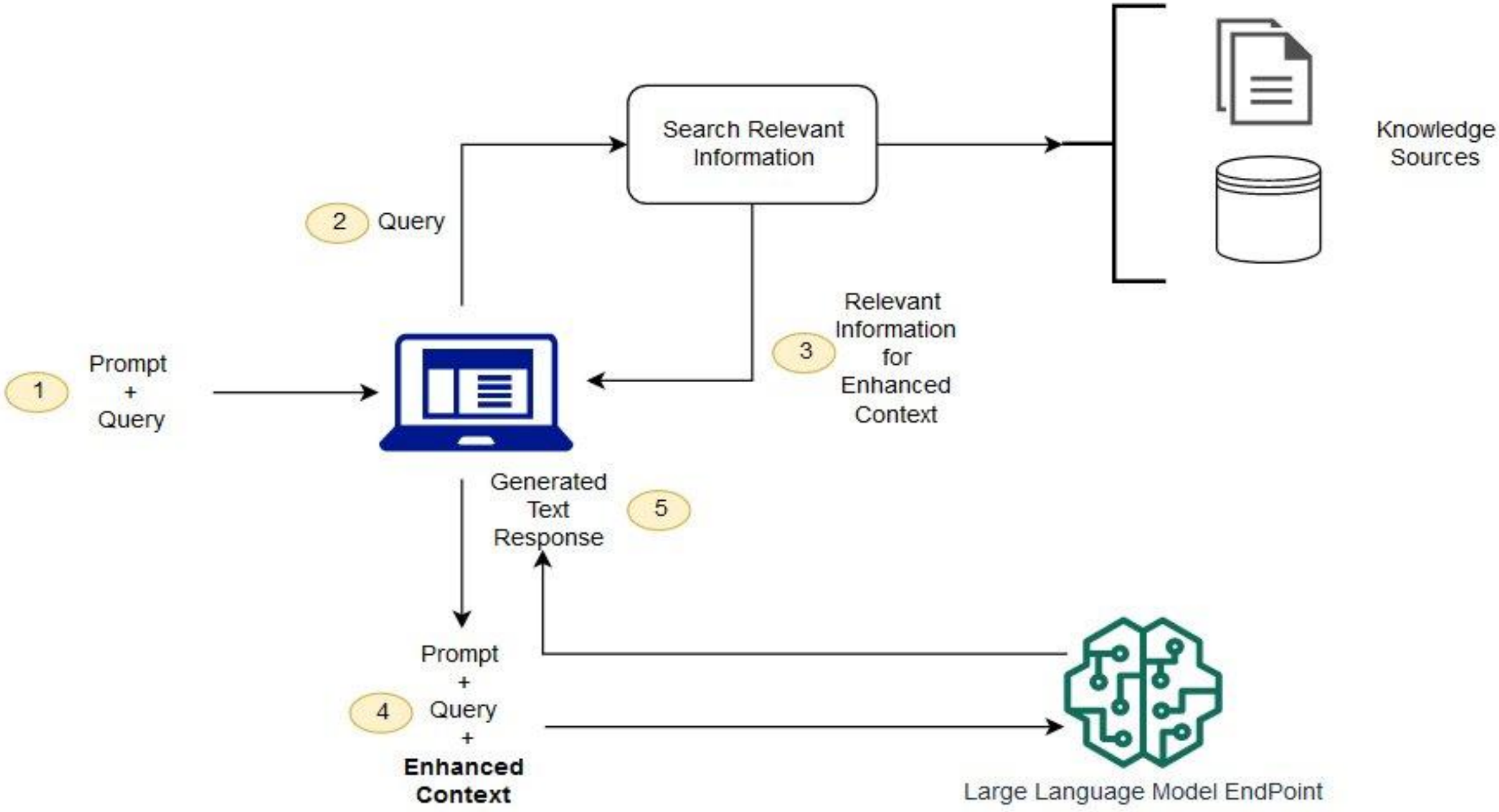


Figure 2.1. Conceptual flow of using RAG with LLMs [3]

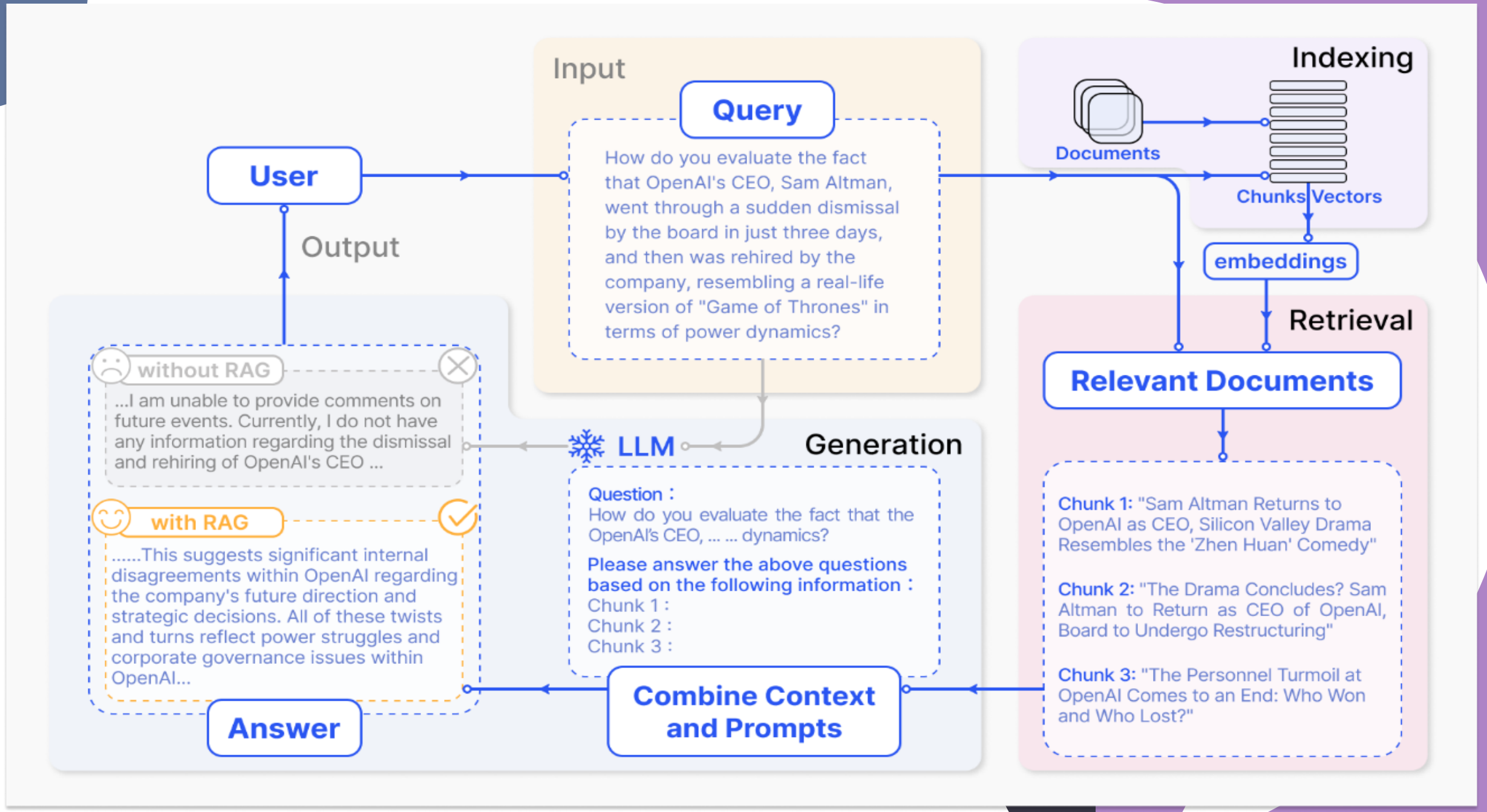


Figure 2.3. Overview of RAG [3]

LLMs Compress World Knowledge

A fundamental LLMs feature of LLMs is the ability to compress world knowledge. The way this works is you take a huge slice of the world's knowledge through more books and doucments than anyone could ever read in their lifetime and you use it to train a LLMs and what happens in this training process is that all the knowledge and concepts theories and events that have happened in the that are represented in the text of the training data they get represented and stored in the models weights so essentially what has happend is we're comressed all that information into single language model well this has led to come of the *biggest AI innovations the world has ever seen there two key limitations* for compressing knowledge this way [2].

LLMs Compress World Knowledge

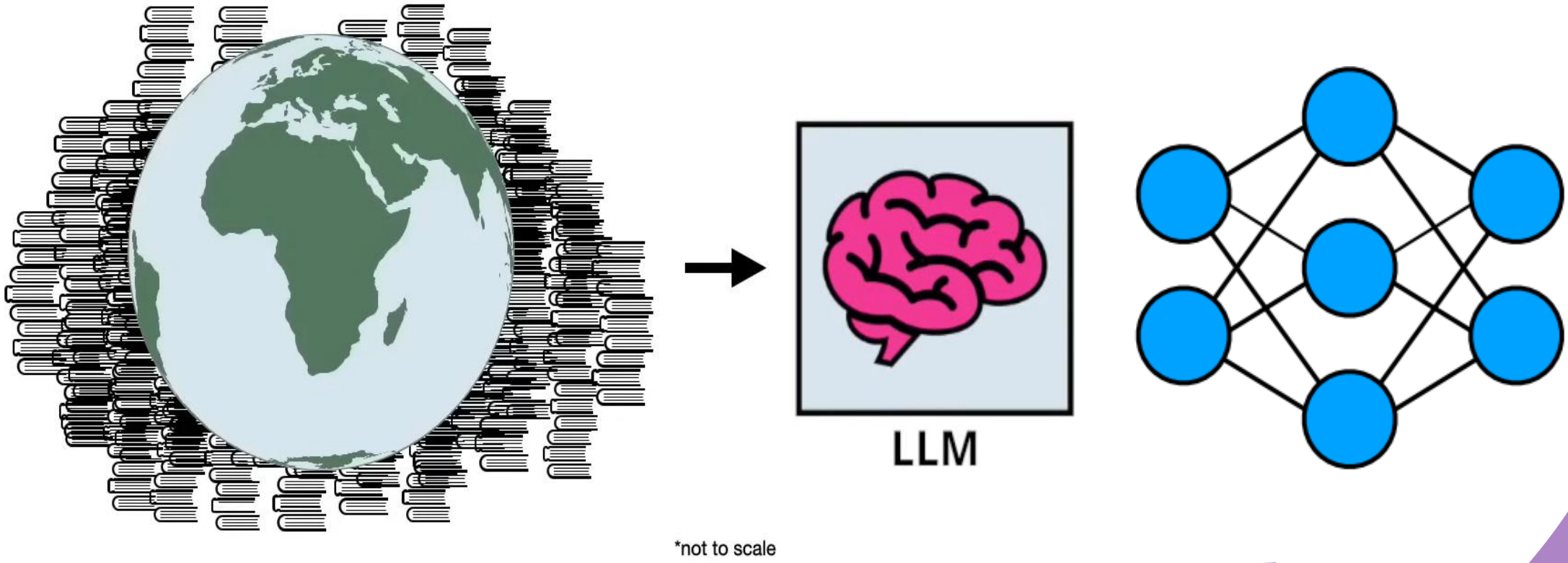
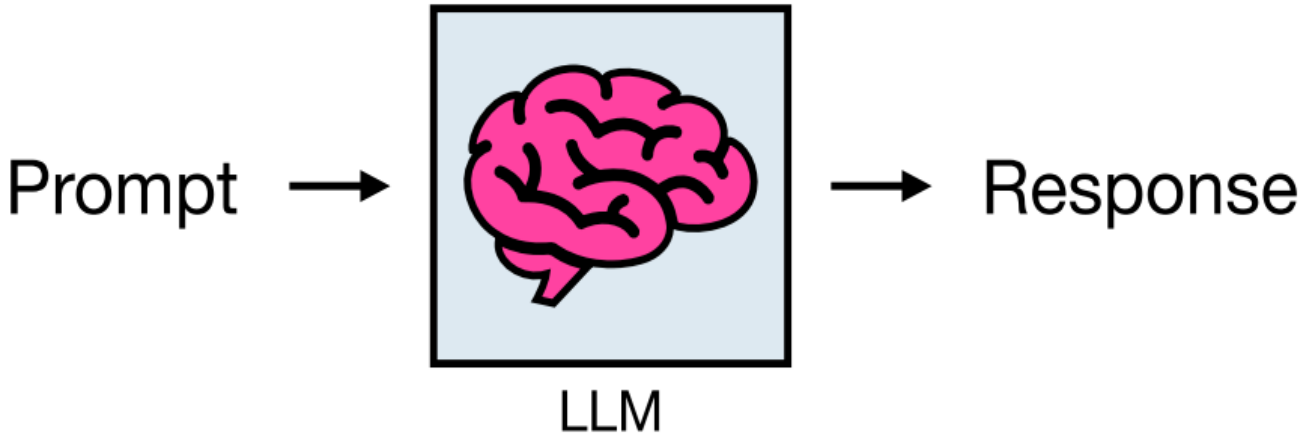


Figure 2.4. LLMs Compress World Knowledge [4]



Why we care?

Why not FT?

Figure 2.7. LLMs Structure [4]

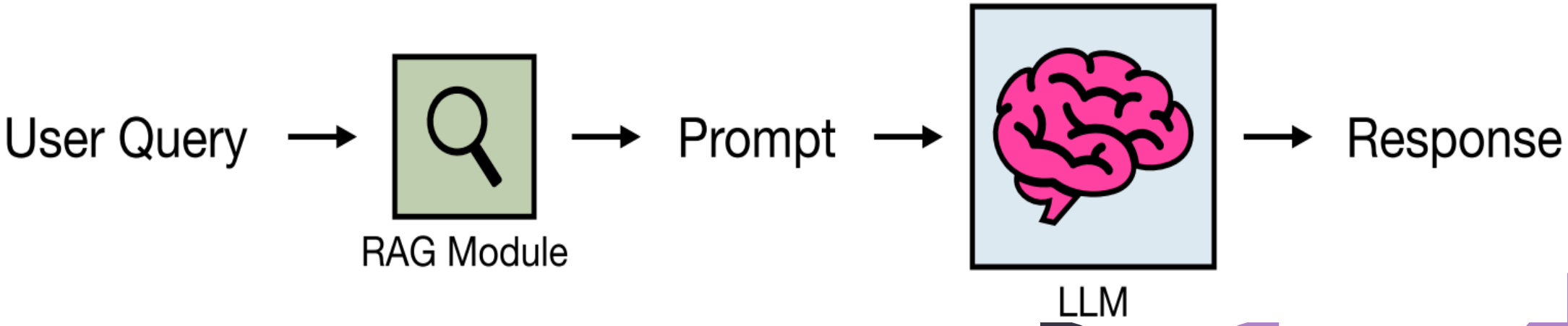


Figure 2.8. RAG and LLMs Togethers [4]

Problem with Text

Summarize Heights
Data Analyst in retail, 5 yrs. Specializes in consumer behavior & predictive analytics.
ML Engineer in fintech, 3 yrs. Develops algorithms for stock market predictions.
Data Scientist in healthcare, 10+ yrs. Uses patient data for better care outcomes.
Database Admin for e-commerce, 7 yrs. Focuses on efficient, secure data storage.
BI Analyst in hospitality, early career. Turns data into insights for growth.
Data Architect, 15 yrs, builds big data systems and data lakes for startups.

Figure 2.4. Summarize Roles & About People [5]

Text Embeddings Meaningful

Translate words into meaningful numbers. That would look something like this where the numbers that we generated in the text embedding define the location of each person's job description We can see the data analyst in retail with 5 years of experience is somewhere here and they are located next to a freelance data visualization specialist with 4 years of experience However these people are relatively far away from this guy who is a data architect with 15 years of experience And so this is the way that text embeddings capture the meaning of the underlying text namely job descriptions that are similar will be located close together while job descriptions that are very different will be located far away from each other From this view the BI analyst in hospitality early career is very different than the data architect with 15 years of experience [6].

Text Embeddings & Text Embedding Meaningful

Job Description	Text Embedding
Data Analyst in retail, 5 yrs	(3.4, 1.5)
ML Engineer in fintech, 3 yrs	(1.8, 3.1)
Data Scientist in healthcare, 10+ yrs.	(6.6, 2)
Database Admin for e-commerce, 7 yrs.	(4.2, 5)
BI Analyst in hospitality, early career.	(0.5, 1)
Data Architect, 15 yrs.	(6.5, 5)
Freelance Data Visualization Specialist, 4 yrs.	(2.6, 1.6)
Senior Data Engineer in automotive, 8 yrs.	(5, 5.5)

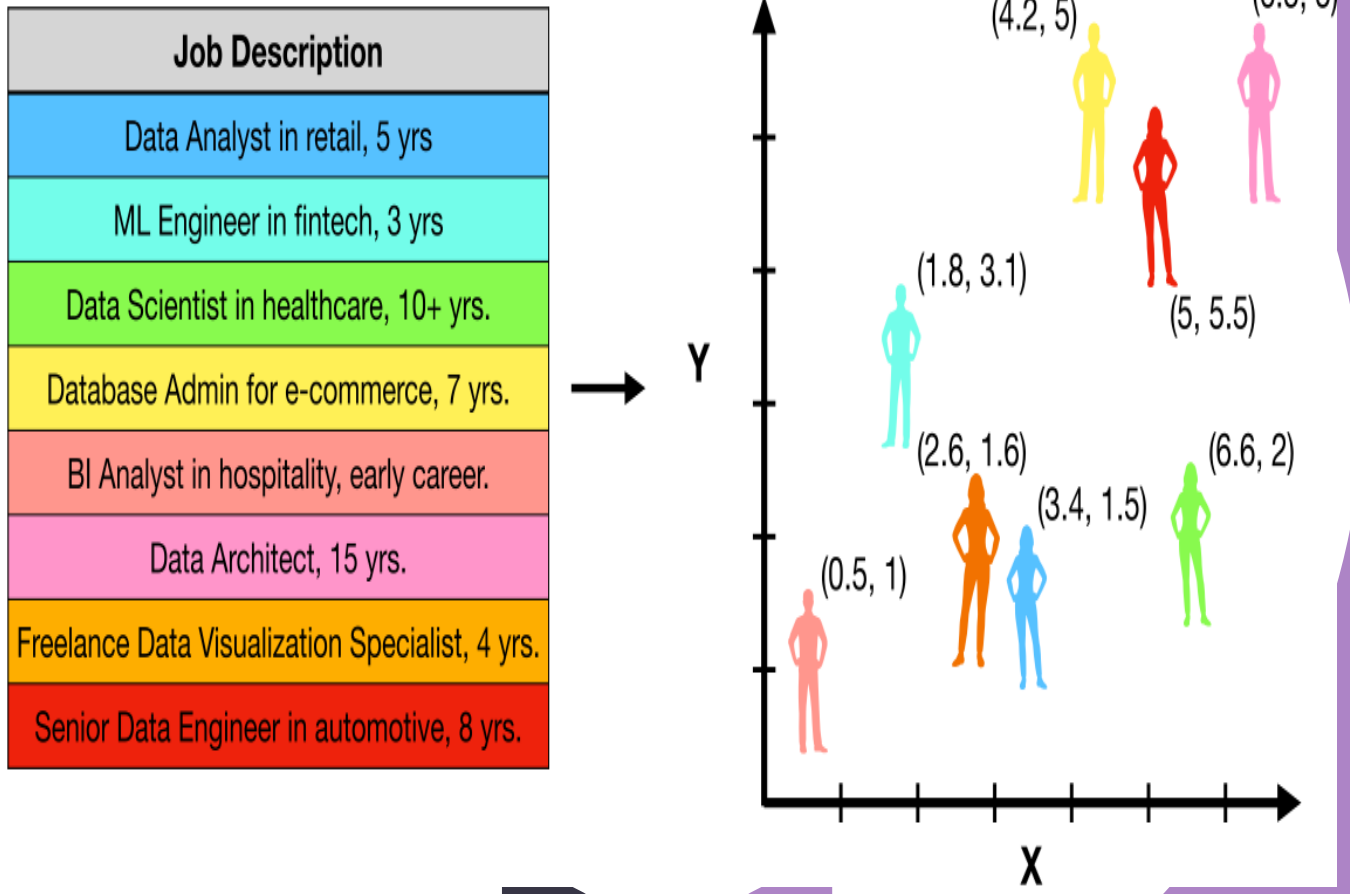
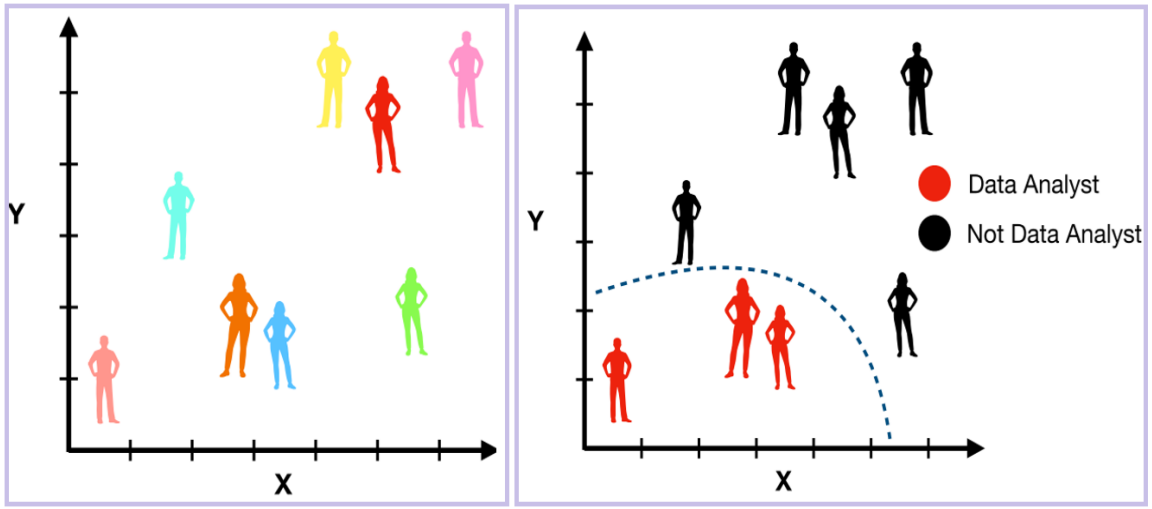


Figure 2.13. Text to Numbers [5]

Figure 2.14. Text Embeddings (Meaningful) [5]

Text Classification

Text classification, which *is the process of assigning a label to a piece of text*. For example, if we were to take all the people from our networking event and their associated job descriptions, a classification task might be to try to determine which people are data analysts and which people are not data analysts based on their respective job descriptions So that might be pretty easy here, because anything to the left of this blue line we could label as a data analyst and anything to the right we could label as not a data analyst [20].



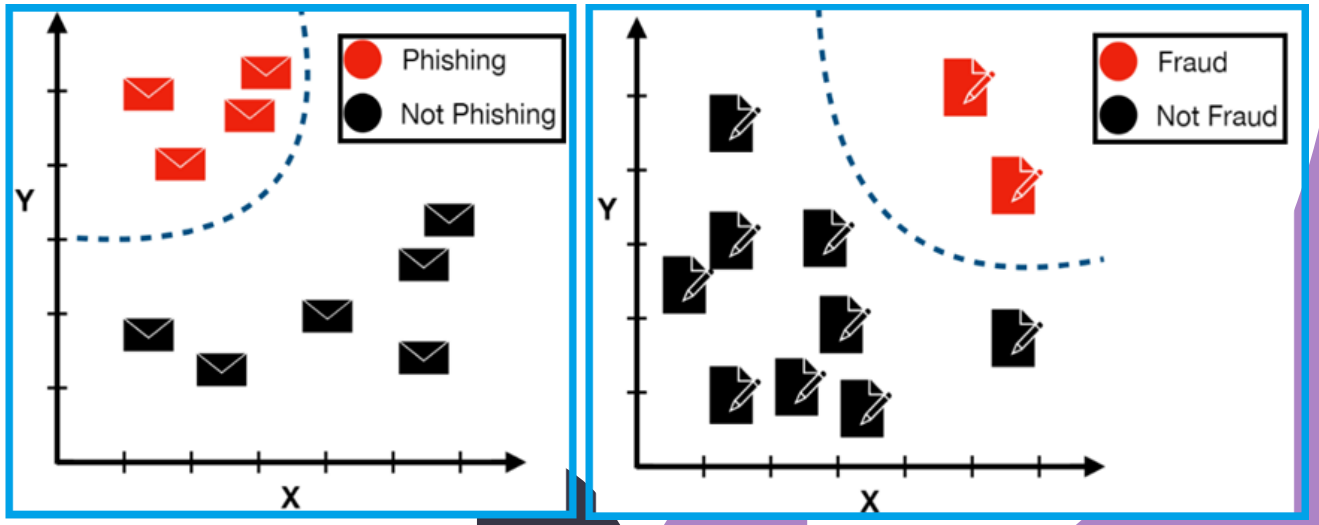


Figure 2.16. Text Classification - Data Analyst [5]

Figure 2.17. Phishing & Not Phishing - Fraud & Not Fraud [5]

Large Language Models (LLMs)

The large language models (LLMs) that can generate human-like text, and I've been using GPT in its various forms for years.

LLMs are an instance of something called a foundation model. Foundation models are pre-trained on large amounts of unlabeled and self-supervised data, meaning that the model learns from patterns in the data in a way that produces generalizable and adaptive output. And LLMs are instances of basic models applied specifically to text and text-like things, in this case things like code. LLMs are trained on large datasets of text, such as books, articles, and conversations. When we say "large", these models can be tens of gigabytes (10 GB) in size and are trained on enormous amounts of text data, we're talking potentially petabytes of data here, a text file that is one gigabyte in size can store about 178 million words, a lot of words just in 1GB and 1PB it's about 1 million. LLMs are also among the largest models when it comes to the number of parameters, a parameter is a value that the model can change independently as it learns and the more parameters a model has the more complex it can be, GPT-3 for example is pre-trained on a corpus of actually 45TB of data and it uses 175 billion ML parameters [7].

A key characteristic of LLMs is their ability to respond to unpredictable requests. A traditional computer program receives commands in its accepted syntax or from a particular set of user inputs. A video game has a finite set of buttons, an application has a finite set of things a user can click or type, and a programming language consists of precise if/then statements [8].

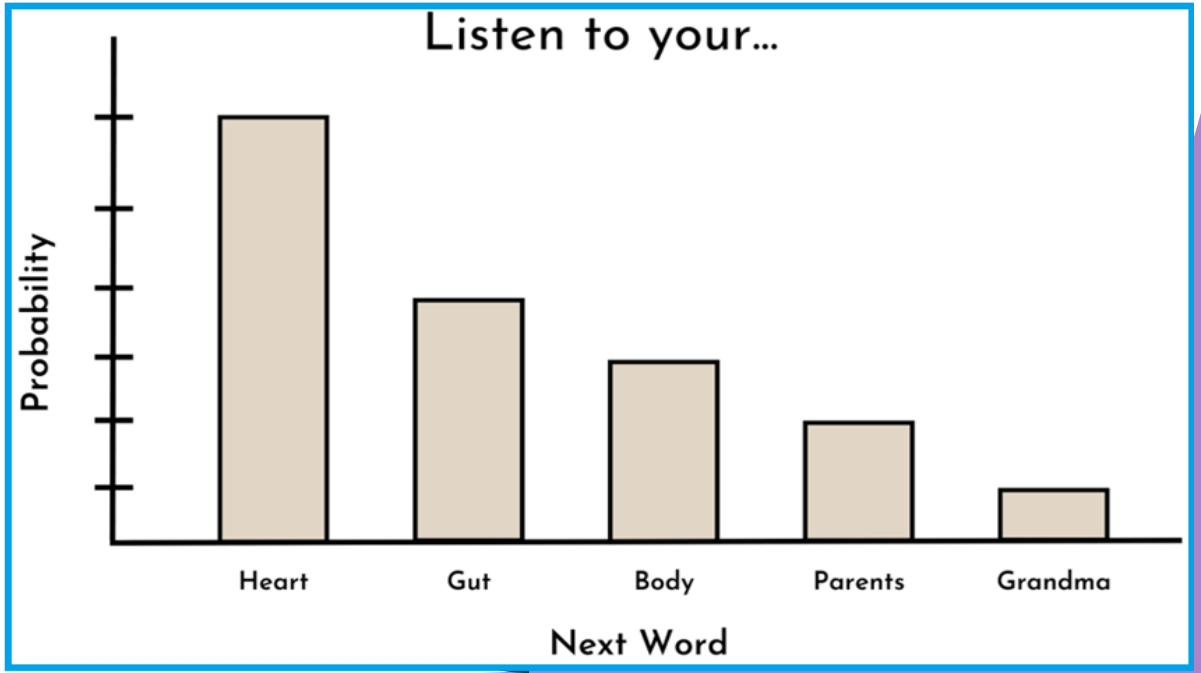
In contrast, an LLM can respond to natural human language and use data analysis to answer an unstructured question or prompt in a way that makes sense. While a typical computer program would not recognize a prompt such as "What are the four greatest funk bands in history?", an LLM might respond with a list of four such bands and a reasonably cogent defense of why they are the best [8].

Next-Word Prediction Paradigm

We have Figure 3.1. text listen to your and want predict what the next word would be but clearly there's not just one word that can go after the string of words there are actually many word we can put after this text and it would make sense in the next word prediction paradigm what language model is training to do is to predict the probability distribution of the next word give the previous words, what is might look is listen your heart might be the most probable next word but another likely word could be gut or listen to your body or listen to your parents and listen to your grandma and so this is essentially the core task that these LLMs are trained to do and the way the LLMs will learn these probabilities is that it will see so many examples in this massive corpus of text that is trained on and it has a massive number of internal parameters so it can efficiently represent all the different statistical associations with different words. An important point here is that context matters if we simply added the word don't to the front this string here and it changed it to don't listen to your then this probability distribution could look entirely different because just by adding one word before this sentence we completely change the meaning of the sentence and so to put this a bit more mathematically we say this is the most technical thing in this part of works [9][10].

Figure 3.1. Next-word prediction [12]

The Large Language Models (LLMs), such as OpenAI's GPT, Google's Bard, etc., are essentially advanced next-word prediction systems. This is the foundation for their much broader capabilities, such as text generation and complex problem solving. Although their applications seem limitless, at their core they are simply predicting the word that is most likely to appear next, based on patterns they have observed in large data sets. This paper discusses next-word prediction, how it works in LLMs, and why this simple mechanism produces remarkable capabilities [11].



Fine-Tuning

Tweaking an existing model for a particular use case. FT is like taking a raw diamond, refining it, and distilling it into something more practical and usable, like a diamond you might put on a diamond ring. In this analogy, the raw diamond is your base model, so that would be something like gpt3, while the final diamond you come away with is your fine-tuned model, which is something like Chat GPT. GPT-3 (base model) is pre-trained and ChatGPT is Fine-tuned. [13].

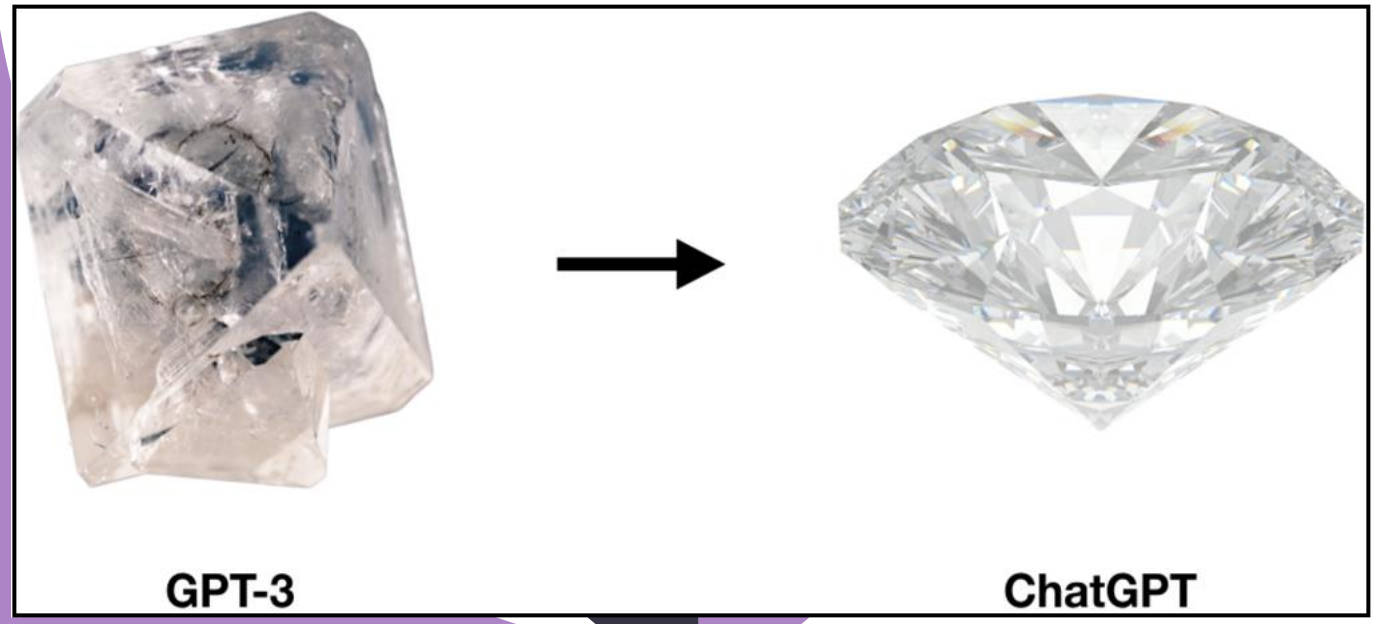
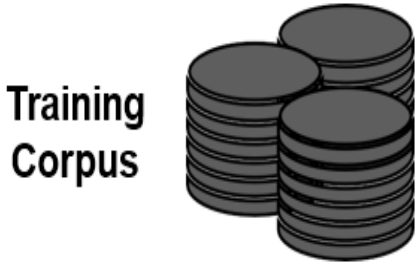
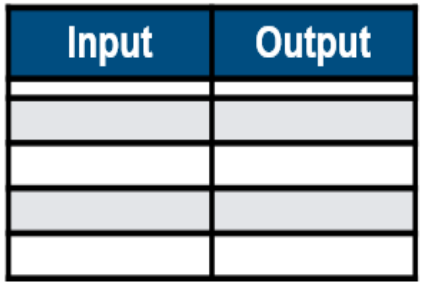
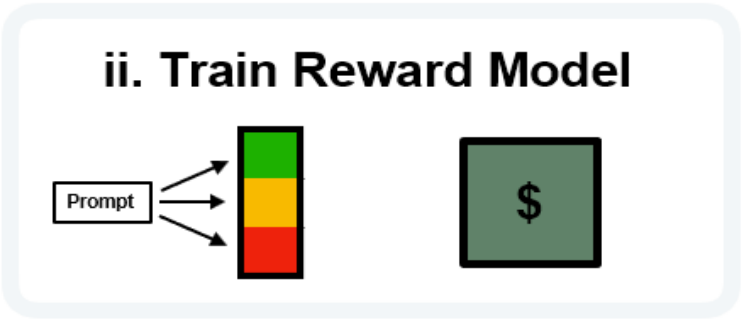


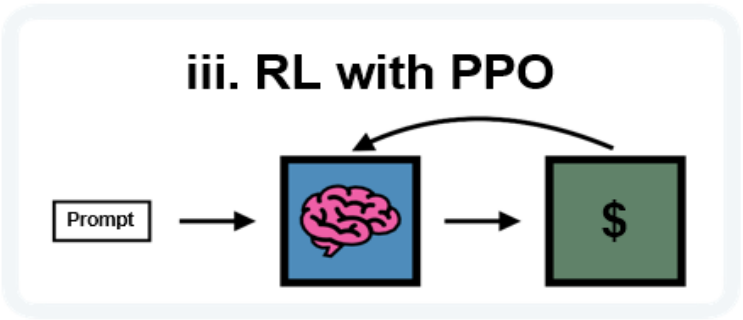
Figure 3.2. Fine-tuning (Raw Diamond to Diamond) [13]

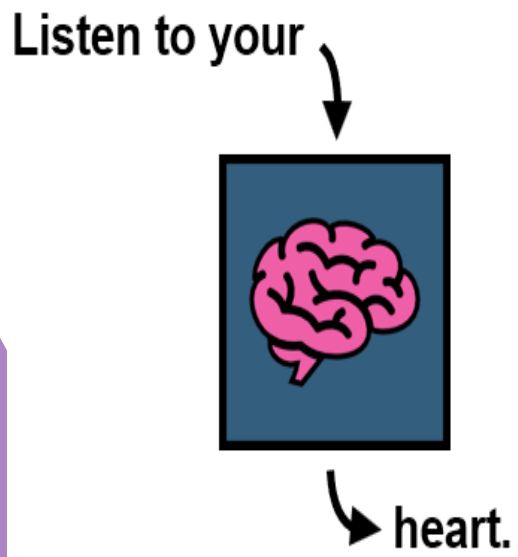
3 Ways to Fine-tune











Input: *Who was the 35th President of the United States?*

Output: *John F. Kennedy*

```
"""Please answer the following question.

Q: {Question}

A: {Answer}"""
```

Figure 3.3. Self-supervised [14]

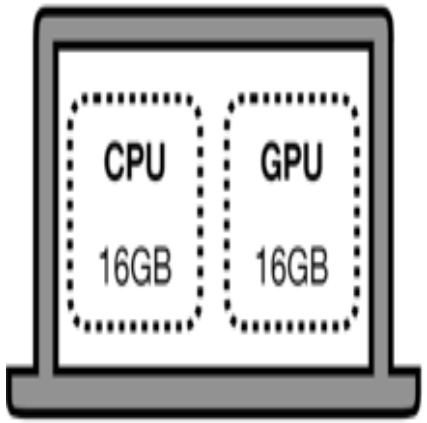
Figure 3.4. Supervised [14]

Figure 3.5. Reinforcement Learning [14]

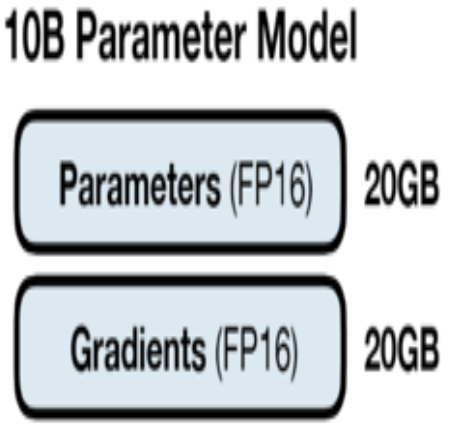
The Problem (LLMs are computationally expensive)

The core problem with fine-tuning LLMs is that they are computationally expensive to get a sense of, say you have a pretty powerful laptop and it comes with a CPU and a GPU where the CPU has 16 GB, the CPU has 16 GB of RAM and your GPU has 16 GB of RAM. We want to fine-tune a model with 10 billion parameters, each of these parameters corresponds to a number that we need to represent on our machine, the standard way to do this is to use the FP16 number format, which requires about two bytes of memory per parameter, so just doing some simple math here, 10 billion parameters times 2 bytes per parameter comes to 20 GB of memory just to store the model parameters, so one problem here is that this 20 GB model won't fit on the CPU or the GPU, but maybe we can get clever in how we distribute the memory so that the load of the model is split between the CPU and the GPU, But when we talk about fine-tuning, we're talking about re-training the model per parameter, which is going to require more than just storing the parameters of the models. Another thing we need is the gradients, these are numbers that we use to update the model parameters in the training process, we have a gradient that's just going to be a number for each parameter in the model, so that adds another 20 GB of memory, so we've gone from 20 to 40, and now even if we get super clever with how we distribute it across our CPU and GPU, it's still not going to fit, so we actually need to add another GPU. To even make that work, but of course that's not the whole story, you also need room for the optimizer states, so if you're using an optimizer like 'Adam' which is very widely used, this is going to take up the memory footprint of model training, where this is coming from is an optimizer like 'Adam' is going to store the memory value and the variance value for each parameter in your model, so we have 2 numbers per parameter In addition, these values have to be encoded with higher precision, so instead of the FP16 format, and those are about $20000 dollars. We are probably talking about like $50000 dollar just for the hardware to fine tune a 10 billion parameter model in the standard way [15].

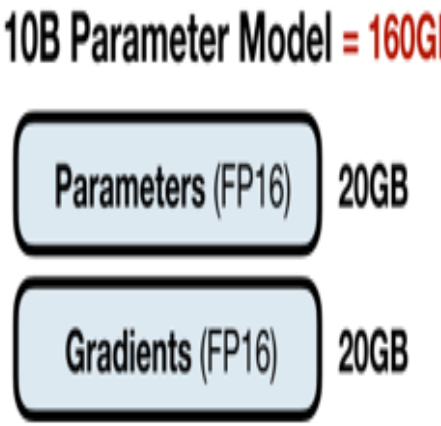
Fine-tuning is Re-training the parameter of model, Gradients is update the number of parameters models



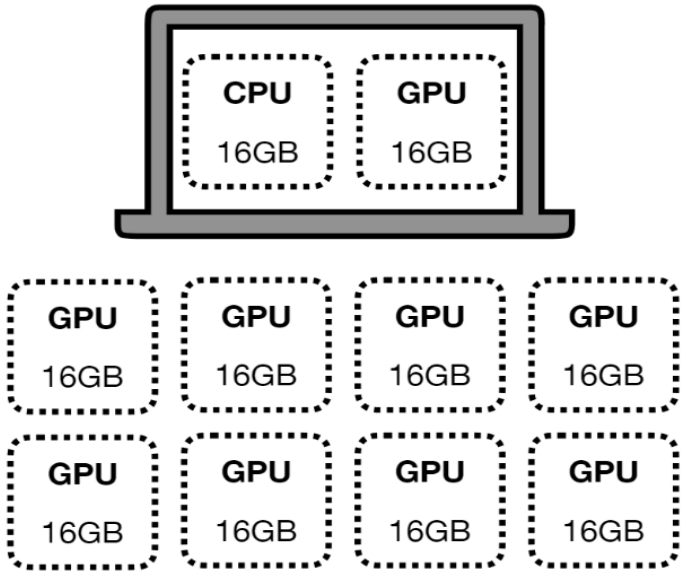
CPU & GPU



10 Billion



160GB = $20000 Dollar



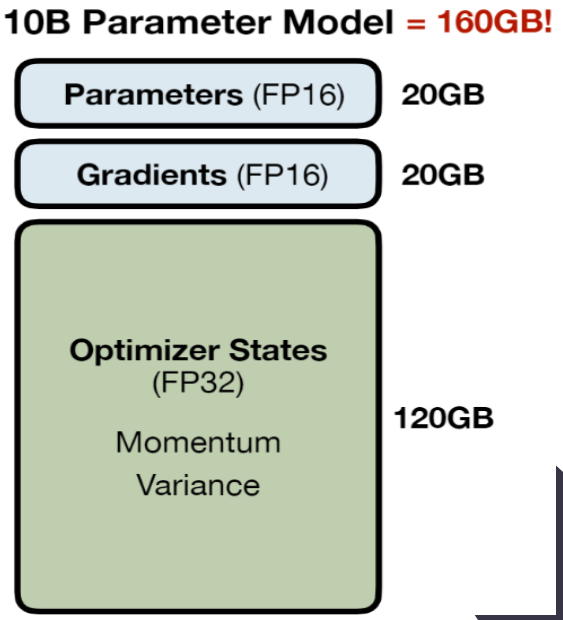


Figure 3.6. Fine The Problem (Fine-Tuning) LLMs Cost [15]

Full Finetuning, LoRA and QLoRA

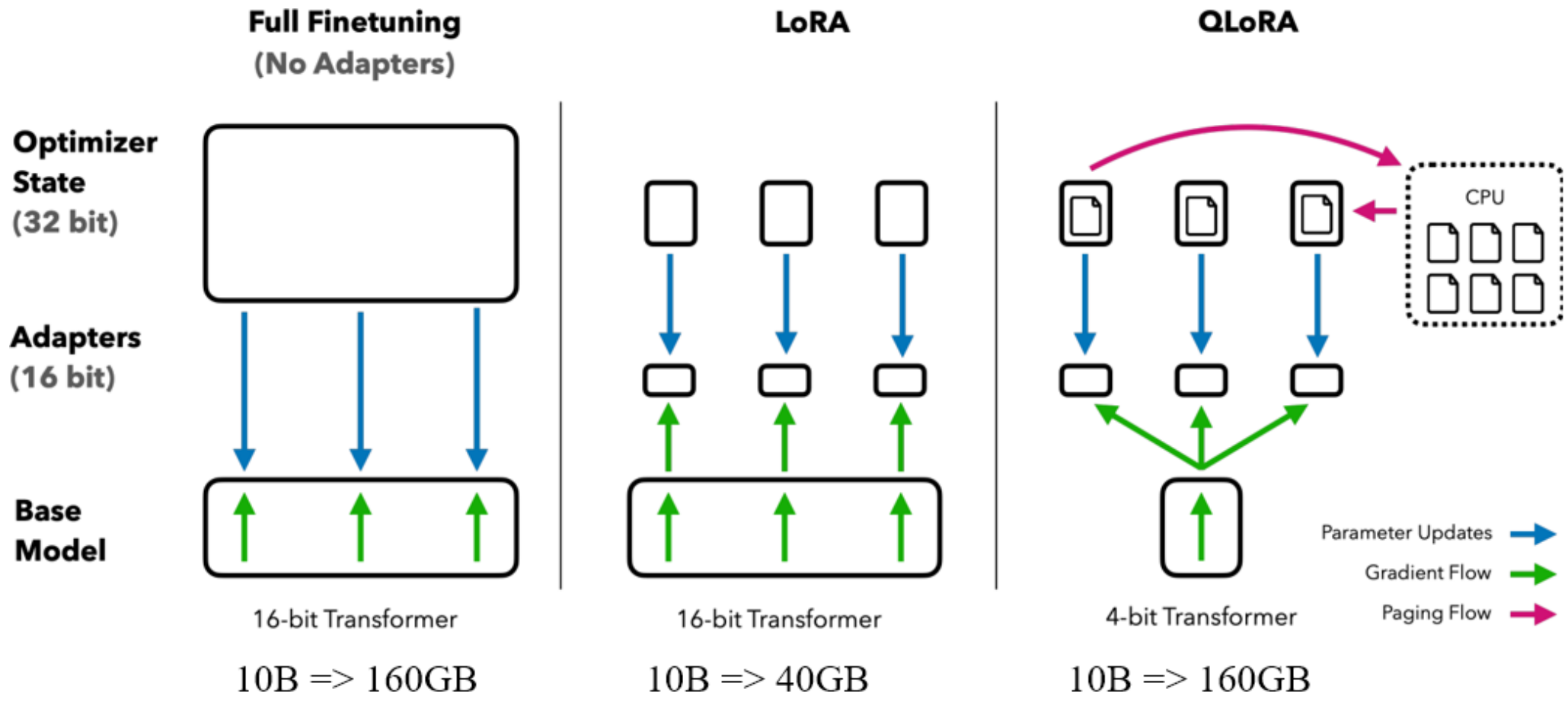


Figure 4.1. Fine The Problem (Fine-Tuning) LLMs Cost [16]

Sources

[1] https://medium.com/@juanc.olamendy/rag-best-practices-enhancing-large-language-models-with-retrieval-augmented-generation-6961c8b834ff

[2] https://youtu.be/Ylz779Op9Pw?si=btCyBEkcRWSgrnC-

[3] https://aws.amazon.com/what-is/retrieval-augmented-generation/

[4] https://github.com/ShawhinT/YouTube-Blog/blob/main/LLMs/_slides/rag.pdf

[5] https://github.com/ShawhinT/YouTube-Blog/blob/main/LLMs/_slides/text-embeddings.pdf

[6] https://youtu.be/sNa_uiqSlJo?si=FrF-JEIJjcHoK9Im

[7] https://www.youtube.com/watch?v=5sLYAQS9sWQ

[8] https://www.cloudflare.com/learning/ai/what-is-large-language-model/

[9] https://youtu.be/tFHeUSJAYbE?si=sswH0bZ5xIQBKY5s

[10] https://arxiv.org/pdf/2303.18223

[11] https://www.researchgate.net/publication/386506764_Large_Language_Models_are_Next_Word_Predictor

[12] https://github.com/ShawhinT/YouTube-Blog/blob/main/LLMs/_slides/intro-to-llms.pdf

[13] https://www.youtube.com/watch?v=XpoKB3usmKc&list=PLz-ep5RbHosU2hnz5ejezwaYpdMutMVB0&index=10

[14] https://github.com/ShawhinT/YouTube-Blog/blob/main/LLMs/_slides/fine-tuning.pdf

[15] https://www.youtube.com/watch?v=XpoKB3usmKc&list=PLz-ep5RbHosU2hnz5ejezwaYpdMutMVB0&index=10

[16] https://arxiv.org/pdf/2305.14314

Thanks

Mohammad Amin ASLAMI

+90 542 654 96 92

aminaslamiaf@gmail.com

Firat University - Faculty of Technology

Department of Software Engineering - Master Degree’s