Section 1: Introduction to RAG

Course: Ultimate RAG Bootcamp Using LangChain, LangGraph and Langsmith
Section Number: 1
Total Videos: 4
Date Created: 2024

Video 1: Introduction to RAG

Video Order: 1/4
Topics: RAG fundamentals, Retrieval, Augmentation, Generation, Architecture overview
Difficulty: Beginner

Content

Hello guys! So I'm super excited to start the series of videos on RAG.

In this video and in the upcoming series of videos, we are going to understand everything about retrieval augmented generation, a very important topic currently. We will talk about various architectures. We'll talk about how to build a RAG pipeline. We'll talk about how to build generative AI with RAG, how to build generative AI applications with RAG.

If I talk about the current scenario, more than 80 percentage of the business use cases currently who are working in generative AI and AI, they are building RAG applications. RAG applications. So it is really important for anyone who is specifically interested in generative AI or genetic AI. They need to know how RAG architecture is basically implemented.

RAG Architecture Overview

Now, with respect to the RAG architecture here, I have shown you one type of architecture where we have three important phases:

1. **Document Ingestion phase**
2. **Query processing phase**
3. **Generation phase**

When we talk about retrieval - what exactly is retrieval? We will discuss about what exactly is augmented. Will discuss about what exactly generation is what we are going to discuss about. And just by seeing this architecture, there are so many things that has been provided over here, but our learning pattern will be in such a way that we will learn one topic at a time. We will implement multiple use cases and at the end of the day, we will try to

understand each and everything. And how do you build real world applications as we go ahead?

Definition of Retrieval Augmented Generation

First of all, as usual we will go ahead with the simple definition. What exactly is retrieval augmented generation?

Simple definition: It is a powerful technique that enhances AI language models by combining the generation capabilities with external knowledge retrieval.

So here you can see again, it is just enhancing the AI language model or any large language model by combining the generation capabilities with external knowledge retrieval.

Understanding RAG Through Analogies

RAG is like giving an AI assistant access to a library where while it's answering questions, instead of relying solely on what it has learned from training, the AI can look up specific, current or specialized information from external sources before generating its response.

Think of it in this way:

- **Traditional large language models** are like students taking a closed book exam. They can only use what they memorized.
- **RAG enabled models** are like students in an open book exam so they can reference materials to provide more accurate details and up to date answers.

Basic LLM Limitations Example

Now here you can see that whenever we have any LLM model. Let's consider that this is my LLM model. I hope everybody knows about what is LLM. Now in this LLM model, let's consider this our basic LLM model. And this model is trained on data till July 2023.

With respect to this particular model, you will be seeing that whenever we give an input to this model, this LLM, since it has been trained with data before July 2023, it will be able to give you some kind of output. So here it will be able to generate some kind of output.

But what if I ask a question "hey what is the current news?" Like let's say today date I give it right. This LLM will not be able to answer because obviously it is trained on this specific data. Right. Till this specific data itself. So LLM will not be able to answer this.

Solution 1: Integration with External Tools

So what LLM does is that if we provide LLM integration with some third party tools, third party tools, let's say one of the tool is a web search tool. So let's say I have given some web search tool option over here with respect to this particular integration.

Now with this web search, what it will do whenever we give this question, "what is the current news", the LLM will be able to make a call to this web search tool and it will be able to get the response. Once it gets the response, the LLM will be able to provide you the output saying that hey, today's news is this. So here the dependency is there that there is some external kind of tool which is nothing but a web search API. And it is able to give you the answer.

Solution 2: Company-Specific Information with RAG

Now this is perfect right. Now in this particular use cases you will be able to see that we have dependency on external tools. Now, let's say if I go ahead and ask, this is my company XYZ. Now for this particular company I say, "hey what is the leave policy or what is the recent leave policy of XYZ?"

Let's say inside this company I'm an employee XYZ company. I'm an employee. And I want to basically know what is the recent leave policy of XYZ. So I asked this particular, let's say this is a chatbot that has got created for this work only. Where I can ask anything company related information and this chatbot will be important because this chatbot is a kind of chatbot, which is important to this particular company because it is being able to provide you some kind of information.

Now, in this particular question, the LLM will be dependent on some kind of retriever. So let's say this company information, whatever information is there, the current information, everything. It will be stored in some kind of database. So let's say this is my database. And this database we say it as **vector database**.

Understanding Vector Databases

So as we go ahead we will learn more about this. What exactly is vector database. How does it work. But I'll just give you an idea. So this is one kind of database which is called as vector database. Now inside this vector database if you know, vector database usually stores text data in the form of vectors.

Now whenever I ask this question "what is the recent leave policy of XYZ". So what this will do, it will try to interact with this vector database. Because these all data information is stored already in this vector database. And based on this we will be able to get a specific response.

The Three Components of RAG: R-A-G

Now this kind of communication when where we are trying to communicate with some kind of vector database. And then we take this data from the vector database as a context. And the LLM will be able to give you the output, the summarized output. So here three important things are basically happening. And that is what RAG is all about.

So here I have RAG:

- **R** - Retrieval
- **A** - Augmented
- **G** - Generation

Retrieval (R)

Retrieval basically means what? Since I am communicating with the database. So this is my vector database. So we are retrieving some content from this particular database. We are retrieving some content from this database.

Generation (G)

LLM taking this context and it is generating some kind of output. So here it can be a summarized output. It can be any kind of output. So some output is basically coming up. That basically means this LLM along with the context that it got from the retriever is generating something.

Augmented (A)

Now what does augmented basically mean? Now before I go ahead and talk more about augmentation, now the next question arises how do we go ahead and retrieve from this vector database. So here different metrics will be used like similarity metrics. Similarity metrics or similarity search. There will be a similarity search. So there can be any kind of different metrics. We'll learn more about this.

So in short what I'm doing is that I am querying a vector database. I'm getting the context. And then I'm trying to generate a summarized output.

Understanding Augmentation in Detail

Whenever we are retrieving any content from this particular vector database or any context from this vector database, augmentation basically means that we will try to **enrich** the retrieved content.

Let's say from my vector database I got my query was something like, let's say that I'm just trying to probably, let's say from this particular retrieval I got something like this: I say I got an answer like "revenue increased by 10%".

So let's say in the retrieval, when we are retrieving from the vector database based on a query, we got this result. So when we say augmentation how we are enriching this content you will be able to see.

If this is the text that I'm getting from the vector database we will try to add more **metadata**. Now how we will be adding more metadata? Augmented basically means the same thing. So augmented for this specific example will look something like this:

Text: "Revenue increased by 10%"

Source: Tesla Annual Report

Date: 2024

The next thing that we are going to get is source. Source is nothing but let's say Tesla. It can be a Tesla annual report. It can be any company report. Let's say I've asked, let's say the query is related to Tesla. So I can get the source. Then I can go ahead and add date.

These all are metadata. Now with the help of this metadata, you know at the end of the day your generation, the summarized output will be really, really, really good. So this way we will be able to very accurately provide an output.

Summary of RAG Process

So this is what is all about RAG. What exactly Retrieval Augmented Generation basically means:

1. **Retrieval** is nothing but we are hitting a query. We will try to based on an input query. We will try to search from the vector database. And then it will do a similarity search. Get the documents or get the some kind of content which is matching this particular vector database.
2. **Augmentation** - Through the augmentation we'll try to enrich the retrieved content.
3. **Generation** - Finally that enriched content that we are giving it to the LLM in the form of context will generate a summarized output.

So this is all about retrieval augmented generation which is nothing but RAG. Now here you'll be able to see that what all things are basically there step by step. We'll try to understand as we go ahead.

Video 2: Some Examples and Advantages Of Using RAG

Video Order: 2/4
Topics: RAG workflow, Similarity search, Augmentation details, Real-world examples, Customer support use case
Difficulty: Beginner-Intermediate

Content

Hello guys. So we are going to continue the discussion with respect to RAG already. We got some idea at least to make you just understand what is RAG and how does RAG actually work.

One is retrieval augmented generation. One very important thing that I missed out in the previous video is that whenever you make an input query, and when we say that, hey, we are going to do some kind of search from the vector database. This is something like a **similarity search**. That basically means based on this input, whatever will be matching that will be available over here will be provided as a context. And then with the help of augmentation, we enrich this retrieved content by adding more metadata. And finally, we give it as a context to the LLM which will be generating the summarized output, which is the part of generation.

Breaking Down RAG Components

So in short, if I talk about retrieval augmented and generation here:

For Retrieval

I will basically say, hey, we are **finding relevant information**. We are finding relevant information from where? From **vector databases**. And there are various types of vector databases which we will be learning in our course.

For Augmentation

Now with respect to the augmentation what exactly it is. It is nothing but **enhancing** which I have already explained. Enhancing context with metadata. And why we are doing this? Because it will actually, our LLM will be able to answer much more accurately based on the input. We will see that you know how things are basically going to happen and all.

For Generation

And finally, with respect to the generation here, you can see it is **producing the answer**. So these are the three main important things with respect to any RAG architecture.

Why Does RAG Actually Matter?

Now let me ask one very important question. And this question is just like everybody will be probably thinking about it. Why does RAG actually matter? You know, why is RAG really helpful?

I'll talk about various companies who are able to just do some amazing things with respect to RAG, you know, are they able to save huge amount of money? Bigger companies like JP Morgan and all. And I will probably share you the entire links also if you want. But amazing work with the help of RAG they are actually able to do.

Restaurant Chef Analogy

Now let's take one simple example and let me explain more about RAG because it is important that you understand with a simple example itself. Let's say, and then we'll try to also compare this with LLM.

Let's say there is a **restaurant**. So I will just try to draw some restaurant over here. So let's say this is my restaurant. And this restaurant over here let's say has a **chef**. Now you can consider this chef as an LLM.

Now let's say that there is one customer and this restaurant is somewhere in some country. So let's say the country A or it is in this country A. Now, let's say in this restaurant there is a person who has come from country B. And he has ordered something. So this person is basically ordering something to the chef.

Scenario 1: Chef's Limited Knowledge (Hallucination)

If the chef knows how to cook it, obviously chef will directly cook it. But if he or she does not know how to cook it because they may not know by heart. They don't know the recipe of this particular dish. Let's say this dish is from the European side. And let's say this dish is in some country like India. Obviously some of the Indian chefs will not know how to properly prepare this.

Why chef is not able to properly go ahead and create this particular dish or prepare this dish? Because it is not trained with this kind of dishes. It is basically trained mostly with country of India dishes itself. It knows how to cook biryani. It knows how to cook Indian food specifically.

Now, in this particular scenario, what chef can actually do in order to prepare the food for this person:

1. **One thing is that chef can just guess something** and he or she can prepare a thing. Prepare the food. So let's say there is some guessing work that is going on. So

this person has guessed and prepared the food. And obviously at the end of the day, the food was terrible because obviously the chef does not know how to prepare the food, but he or she has guessed and prepared most terrible food. And it tends to happen. So this scenario is basically called as **hallucination**. Hallucination because the LLM is not trained with some other data or something new data or recent data, it is trained with some other data itself. But if LLM does not know this, even though it will try to guess something. And then it will probably create some other kind of content. So that is nothing but hallucination.

Scenario 2: Honest Admission

2. **The second scenario** will be that the LLM or chef can directly say that, hey, **I don't know**. It might directly give you the answer that I don't know. I don't know about this dish. Let's say you go and ask how to prepare this dish. It does not know. So it will say that. Hey, I don't know.

Scenario 3: Using External Resources (RAG Approach)

3. **The third option** is that let's say the chef has some **library of books**. Some library of books. So what the chef will do, it will go ahead and check the **recipe book**. And then probably create something. By just following the step. Create something amazing. So this is one option.

So similarly let's say this books will be stored somewhere. Similarly in the case of LLM, you know when it comes in this particular scenario, what we can do is that we can bring some database like **vector database** and store all the text data. Let's say image data. Different kind of data. Media files. Video files in the form of vectors inside this particular vector database.

So here inside this vector database we use something called as **text embeddings**. By using text embeddings we basically convert all these things into vectors. And then the chef how it will be referring to the books. The LLM can refer this vector database. And whatever things are matching based on the search, it will be able to retrieve the result, it will be able to summarize it and it will be able to give the output.

So this is more of a kind of example that I really wanted to show you, by just taking an example of chef itself. So here also as I said, three important steps are happening retrieval augmented and generation.

Real World Scenario: Customer Support

Now let's take one real world scenario so that you get a clear understanding about one of the most problem, important problem statement where RAG is specifically used. One is **customer support**.

So in customer support, let's say I will give you both the scenario. Let's say I have a customer support application. And this is **without RAG**. And let's say this is the next one example that I want to show you is **with RAG**.

Now I will give you one question. Let's say the customer has asked this question. The customer over here have asked this question. The question is very simple. It says that **"What's your return policy for items bought during the Black Friday sale?"**

Without RAG (Generic LLM Response)

So if I just have a LLM model, it will try to give up most generic response what is available in the site. Let say it will say: **"Generally most companies offer 30 days return, but policy may vary"** because obviously this LLM model will not have the data inside this particular company where this chatbot is being used.

So usually what happens is that if I just take a normal LLM model, if I just take a normal LLM model and use it directly for my customer support, then this problem will happen. When I ask, hey, what's your return policy for items bought during the Black Friday sale? Whatever generic internet information it may have, or whatever previous information it has been trained with, some other information that generic answer you'll be able to get. Generally, most companies offer 30 days policy return, but policy may vary.

So here you can see that there are three problems:

1. **Generic**
2. **Unhelpful**
3. **Unhelpful response** in short

With RAG (Company-Specific Response)

But once you start implementing RAG in your company, once you start implementing RAG, what kind of output you may get for this kind of customer support.

Now let's say that I have a AI assistant which uses an **LLM plus company data**, which is stored in vector database. Now the same information when it is asked. And let's say that this is my real data that I will be having. This is how I will be able to get a response.

Customer asks: "What's your return policy for item bought during the Black Friday sale?"

RAG will probably search from the company policy database and this company policy database is in vector database itself. And then it will pick the right kind of result, the most matching results. Obviously it is going to apply augmentation and enriching the content, and then the LLM will finally summarize it.

And you know that every company is also probably change their policies. So based on the recent policy, it will say: **"According to our current policy document, version 3.2, uploaded on November 2024, Black Friday purchases have an extended 60 days return window until 31st."**

This is one example I've taken up right now. Tell me just by thinking whether this was good weather for a client? Whether this kind of response is good or this kind of response is good? Obviously, this kind of response we are able to get it because here we have integrated our LLM with our own company policy databases, and we are specifically using this. And this vector database will be belonging to one specific company.

This is one amazing thing about RAG, you know, again, one really amazing thing that I really want to talk about is like how it is able to create such a very beautiful impact. And this impact, I'll tell you that it has, you know, I did a lot of internet search. Like, who all are specifically now using RAGs and how they are able to do this.

Cost Savings and Business Impact

I also have created one amazing graph over here. So I'll just talk about it, you know, with respect to the cost savings and all. So some of the cost saving anyhow, I will be giving you this.

See, so this was the recent news that we were able to see. And I have actually searched it. If you go ahead and search it, you will be able to even get it in the internet also.

1. **JP Morgan** - They saved **$150 million annually** by implementing RAG for research analysts instead of fine tuning. Now, don't worry about what is fine tuning and all. I will talk about it as we go ahead. What are the differences between fine tuning and all before, you know, JP Morgan used to do a lot of fine tuning. There was a lot of cost that is involved in fine tuning. Now. They have completely moved in RAG, you know, they are building the applications with respect to RAG and all. So that's one very good thing that they have saved so much amount of money.
2. **Microsoft** reported **94% reduction in AI hallucination** after implementing RAG in the copilot product. So if you know Microsoft also has a GitHub copilot and all. So initially they were also hallucinating. I told you, right. What is the reason for hallucinating? Hallucinating basically means if the LLM does not know something, it

will still try to make up something and give it to you. Even though it does not have that particular knowledge, it is not trained with that kind of data. So that is the problem that usually happens.

3. **Bloomberg** updates the financial AI assistant **hourly with new market data** impossible with traditional LLMs.
4. **Healthcare companies** use RAG to ensure AI responses always **cites approved medical resources**. You know, so this is with respect to the compliances.

And if I talk with respect to different job profiles, people are specifically using all this kind of RAG applications and they're building it for the company itself.

So I hope in this particular series of video we understood about RAG. Still, I will discuss about this entire architecture. This architecture is specifically for you. But right now still we have not discussed much. We just discussed about what exactly RAG means. You know, what is retrieval augmented generation. Along with this, we also understood about some of the use cases with respect to customer support. Without implementing RAG, what kind of response I get. And with implementing RAG, what kind of response I get.

So I hope you like this particular video. This was it from my side. I'll see you in the next video. Thank you. Take care.

Video 3: Business Impact Use Cases With RAG

Video Order: 3/4
Topics: Business impact, Cost savings, ROI, Industry adoption, Real case studies
Difficulty: Intermediate

Content

Hello guys. So we are going to continue our discussion with respect to RAG. Already in our previous video, you know, I gave some of the points where I told that how JPMorgan were able to, you know, save $150 million annually using RAG and all. So what I thought is that why RAG actually matters, you know, and what kind of business impact it is probably going ahead and creating.

And let me tell you guys, as I said, **more than 80% of the business use cases** in many, many companies are related to RAG. They are specifically building genetic AI applications. They are building genetic RAG systems.

JPMorgan Case Study

So here you can see that, before, you know, JP Morgan used to spend around **$200 million per year** on fine tuning. And then now, you know, recently, you know, they are just spending around **$50 million per year on RAG**. So here you can see that they are anyhow saving **$150 million annually**. For research analysts this is as said.

Microsoft Case Study - Accuracy Improvement

Then, with respect to the accuracy improvement, you can see Microsoft case studies. If you just go and search for this amazingly in the internet you'll be able to get it. So here **94% reduction in AI hallucination**. So before RAG, you know, **34% hallucination** was there right now after RAG **2% hallucination** is left after implementing RAG in copilot.

Bloomberg Case Study - Real-time Updates

Here, you'll see that traditional LLM six months training cycle, RAG system, real time updates. It is **24 times faster**. You know, this is from the Bloomberg case study. This information has basically come up with respect to compliances.

Healthcare Compliance

RAG has been heavily used in **100% source attribution, regulatory compliances, complete audit trail approved success**. Only, you know, and this is how the RAG adoption trends takes place, you know, and probably till 2026, it is shown that, easily around **85%** at least are there going to specifically use over here with respect to RAG adoptions and all.

Why RAG Matters - Key Metrics

Everything over here with respect to this is given. Why RAG matters:

- The average **ROI** (return of investment) with respect to RAG application is somewhere around **312%**.
- **Implementation time** is very, very fast. Within **6 to 8 weeks**, you'll be able to implement it.

When I talk about implementation, I'm talking about, you know, taking it to the production.

Top Industries Adopting RAG

Top industries adopting RAG are:

1. **Finance**
2. **E-commerce**
3. **Healthcare**

4. **Manufacturing**
5. **Legal**
6. **Education**

So these all are there. And that is the reason why I say that RAG is really, really important. I've trained more than 20,000 plus people. You know, everybody nowadays are just coming up with questions related to RAG. You know how to probably go ahead and implement it. So definitely RAG is a thing that everybody needs to know as we go ahead.

Video 4: Prompt Engineering Vs Fine Tuning Vs RAG

Video Order: 4/4
Topics: Comparison of techniques, Pros and cons, Use case selection, When to use each approach
Difficulty: Intermediate

Content

Hello guys. So we are going to continue the discussion with respect to retrieval augmented generation. In this specific video we are going to discuss about this very important topic. Like what are the differences between prompt engineering versus fine tuning versus RAG?

Let me tell you guys, because here there are a lot of confusion from many, many people. You know, if there is prompt engineering, why should we go ahead with fine tuning or why should we go ahead? If there is fine tuning, why should we go ahead with RAG? Each and everything we will be specifically talking about. We'll also talk about how prompt engineering works. Fine tuning works, RAG works, and we'll also talk about the pros and cons as we go ahead.

Prompt Engineering

So first of all let's understand about prompt engineering. You know, so as you all know prompt engineering is more about giving a prompt to a base LLM. When I say base LLM, let's consider that this is one kind of LLM. It can be a OpenAI GPT four or LLM, or it can be llama three. It can be Google Gemini LLM.

Let's say if I just go ahead and give a kind of prompt. Let's say the prompt is like "act as a chef and provide me a recipe". So if I give this information, so this information is just like a prompt where I'm giving a kind of instruction to the LLM to behave in that way.

So based on this particular prompt, now whatever input I specifically give or whatever question I actually ask, I say that, hey, "please tell me how to prepare pizza". So that is the kind of input. So based on like how a chef will give me an answer, it will go ahead and give you a customized output.

How Prompt Engineering Works

So with respect to prompt engineering it is very simple. We provide a prompt to our base LLM. And our LLM basically works based on that specific prompt.

So here, the first thing is that we provide specific instruction in the prompt. Specific instruction in the prompt.

Second, your prompt that you're giving should be very, very structured prompt. It should be a structured prompt with clear context. This is a really important statement. If the context is not clear it will not be able to, the base will not be able to give you a customized output.

Then the third important main property is that the model will remain unchanged. The model is same. The same base model. It is not going to change its parameters or anything as such.

Pros of Using Prompt Engineering

Now let's go ahead and talk about the pros of using this. Pros of using the prompt engineering. So the prompt engineering part.

So the first pro over here is that the first major advantage is that no technical expertise is needed. So here just a normal English language. And you will be able to perform prompt engineering. So no expertise technical expertise is needed over here.

Second you will be able to get quickly and instant results. Quickly and instant results.

Third, here to write a prompt or you know, when compared to fine tuning and RAG, there is no training cost involved. So no training is required. No additional training is required. It is completely for free. You can just go ahead and change the prompt itself. And the LLM should be able to give you a customized output.

Fourth important main property is that it works with any LLM. So you can go ahead and give your own prompt based on any kind of LLM, then it should be able to work.

Cons of Prompt Engineering

Then what are the cons? Because we need to also understand if we know all the advantages. What are the disadvantages.

Now with respect to disadvantages here most of the base LLM models you know it will be limited. It will have a limited base knowledge, you know. So in this scenario, since we have a base LLM, it is always limited by models base knowledge. So here we don't have any external information. So whatever model is basically trained with the previous data, only that information I have right now.

When we keep on changing prompt, sometime it is there are scenarios that we will get inconsistent results. So let's say if I go ahead and put a prompt on one LLM model, I may get a different result. I may just tweak that particular prompt. Then again I will be getting a different result. So inconsistent result. This is the most important thing. Inconsistent result.

Coming to the third important con here. Always remember every model has a token limit restriction. So with respect to that particular point I will go ahead and write it down over here. Token limit restrict complexity. So this is one specific point that I really want to mention.

And the fourth important thing is that it can't add new knowledge. That's it because we cannot add it. Here we are specifically using the base LLM. We are modifying the prompt and we are getting the output.

When Should We Use Prompt Engineering?

Now when should we specifically use prompt engineering? You know, when should we focus on that? See, let's say that you want to quickly go ahead and check, so I'll just go ahead and give a statement what it is best for.

So the first point here, I will say that if you are trying to do a small scale application quickly, do it. Try to do it with the help of prompt, because most of the use cases will be able to solve it for them. And definitely this is not suitable for, see, whenever we use some base model, we are always going to get generic answers. So not suitable for any company specific work. Let's say I want to create an AI assistant only for companies. At that point of time. I will definitely not use prompt engineering for that. I will either go ahead with fine tuning or RAG.

So here it is. This is also used for generic public task or sorry generic. I'm saying public but it is just used for generic purpose task. Generic purpose. All my answers will be mostly generic itself. Small scale application.

Third is that I will also be using this for quickly doing my quick prototyping. So this can also be very, very helpful.

So these are some of the points that we have specified here in the prompt engineering. We just give the prompt. And that prompt is combined with our base LLM. And whenever we get a given input we usually get a customized output.

Fine Tuning

Now the second thing is with respect to fine tuning. Now fine tuning why it is specifically necessary you know. So let's say that I have a company where I am planning to create an AI assistant, which should completely behave like how I want, you know, each and every instruction that I am actually giving to the AI assistant. It should be very specific to my company.

Let's say if there is a chatbot that I have actually created, an AI assistant that I have created. If I say "hi". It should probably reply you back that "Hi. Welcome to XYZ company. How may I help you? Can I provide you the services of this specific company?"

So here specifically we are focusing on a company domain use case. It is completely like how my company actually wants that AI assistant. Similarly I will be able to create it.

How Fine Tuning Works

So here how we actually do the fine tuning. Now with respect to fine tuning it is nothing. But here we basically say that we are going to teach the base LLM through training.

So let's say that there is a base LLM model. What we do is that we get a new training data. This new training data will be the company training data. And then we try to train the base LLM with this specific training data.

You know once we train with this training data, obviously the LLMs usually they use different different architecture. And there there will be weights that weights gets modified. It gets modified. And once we specifically take that modified weights and apply it into our LLM model. It is nothing, but it is called as fine tuned LLM model.

So this is nothing. But this is a specialized fine tuned LLM model which will be specifically used by the company for their problem statement. So this is how entirely and again this is a very big process. Fine tuning techniques are there like LoRA LoRA methods and all. But our course syllabus is mainly focused towards RAG you know.

So here you'll be able to see that with respect to any companies who are specifically focusing in creating their own chatbot AI assistant, which will behave like what the company specifically want that time we'll be using fine tuning itself.

Key Points of How Fine Tuning Works

Now first of all, we will try to understand how it works. So for this, the first important thing is that we have to prepare. What are the requirements.

So we have to prepare a domain specific training data. Domain specific training data. When I say domain specific training data, that basically means it is nothing but the company data for which we are basically creating the chatbot.

Second is that we train model on the data, on company's data. So this is the second important point.

Third is that model weights that are changing. They are permanent. They are permanently changed. You know it will again I'll not go back to my base model. So once the training is done based on this new data, the weights will be completely modified. So it basically creates a specialized version. I can say that. Specialized version. So these are some of the important points.

Pros of Fine Tuning

Now let's talk about the pros and cons. So that you will be able to get a very clear understanding what are the pros and cons with respect to this?

So the first pro that I would like to talk about is that whenever you have specific specialized domain knowledge problem statement, at that point of time for a specific company, this will be very, very helpful. So deeply specialized, deeply specialized knowledge. You'll be able to create this kind of chatbot.

Second, your chatbot will also have a very consistent behavior since we have trained this with our own data.

Third, no prompt engineering needed. Here we specifically don't require any kind of prompt engineering because already my model has been trained with fine tuning data.

It can learn new writing styles, you know, new behaviors, and it is always better for a specific domain. Let's say I want to create a chatbot assistant, which is a legal expert. So what I will do, I will specifically train that model with only legal data. So that are some of the important pros that we can probably bring up while we are specifically creating AI assistant or LLMs by fine tuning.

Cons of Fine Tuning

Now obviously there are pros. They need to be some kind of cons. So what are some of the cons. So let's talk about this. What are some of the cons.

So when with respect to the cons the first important thing that I will be talking about it is expensive. So the entire fine tuning process is expensive. Why? Because obviously we will be requiring GPUs in order to train all this specific models. Not suitable for a startup, you know, who are having less funding. And all. Obviously they don't want to spend a lot of monies with respect to fine tuning in GPUs, renting out GPUs itself.

Second important thing is that over here, when you are trying to create a chatbot with the help of fine tuning, you require good expertise. So require ML expert or AI expertise. You definitely require this because the task and tree training model is very, very huge.

Then regularly again once you start retraining your model. Regularly retraining is required. Definitely. Why? For updates. Once you train your model, retraining is definitely required for updating your model again and again.

So these are some of the cons. And that is the reason only companies whenever it is only required they will go ahead with fine tuning the entire models. But they are also companies who will be specifically doing all this stuff.

When to Use Fine Tuning

Now finally what it is best for? We need to understand that. Let's say I want to go ahead and provide some specific style, specific style in how my chatbot is behaving. I will definitely go in, go ahead with fine tuning my entire AI assistant.

Whenever I have huge data, high volume data, I should definitely go ahead with this. Because they are consistent task and all.

And third, if I'm looking for, you know, where my accuracy should be very, very high. So definitely I will go ahead with fine tuning. So here when accuracy matters. If accuracy does not matter much, if accuracy is critical, then definitely I will go ahead with fine tuning.

So these were the differences between prompt engineering and fine tuning.

RAG (Retrieval Augmented Generation)

Now comes RAG. So so many things we have discussed. RAG simple is like that. You're teaching through retrieval. So here, let me just talk about this RAG. And from this particular diagram, you should be able to clearly understand in a RAG you definitely have some knowledge. External source. It can be a database. It can be APIs. It can be anything as such.

Then based on the user query we use the base LLM that is there. Based on this particular query, we retrieve the documents from this particular knowledge base. We retrieve the

documents. Once we retrieve the documents, then we summarize that document. We augment that document. And finally we generate that specific response.

So here you'll be able to see that there will be something like an external knowledge base where the data is actually put up. It can be APIs, it can be vector databases, it can be anything as such, it can be in-memory databases, it can be documents, it can be anything as such.

So from there we retrieve the documents and then based on the user queries. Now this user query, we try to match it up on various different metrics. Usually when I talk about this knowledge database. It is some kind of databases itself, specifically vector database. I'm not talking about third party API. Also, I can go ahead and interact with other vector databases which are in the cloud by using APIs and all. But at the end of the day here, what you are trying to do is that you are able to retrieve some documents, and then you are able to enrich that documents, and finally you are able to get the output. So here that is the most important property about RAG.

How RAG Works

Now let's talk about some of the important scenarios. So first important thing is that how it works. You know first of all all the documents are stored in vector database. So we store documents in vector DB. Because at the end of the day retrieval is actually created from this.

Second important point is that we retrieve we retrieve relevant documents relevant docs from DB. From. By using queries or like for each query, you can say for each query. But at the end of the day, similarity search is basically happening from that particular vector db. Retrieve relevant docs for each query.

Now, the third important point over here is that it is very, very simple here. You'll be able to see that LLM generates answer from context. Answer from context. So whatever context you specifically get, you are able to generate the answers.

Pros of RAG

Now let's talk about the pros. So here I have my pros. So the first pros is that see why do we use RAG. Let's say if there if company is having some kind of information that is getting regularly updated, then we cannot just go ahead and do the fine tuning. There are a lot of costs that will be specifically involved. So for this particular scenario, you know, whenever there is an up to date information, always go ahead with RAG. So here I will say that it will always have up to date information info.

Second important thing is that no training required. This is the major cost. No training required. We don't have to train the base LLM because there will be huge number of parameters. You know, when we talk about cost effective, it is also very, very cost effective.

And based on the techniques of RAG, the accuracy is also very, very high. Accuracy is high but not that high as fine tuning. But I think a good, like right now, the kind of RAGs that we are specifically able to see is quite amazing. So the accuracy is increasing again and again.

The fifth important point is that it can also handle private. Private or proprietary data. Proprietary data because everything will be stored in some kind of databases.

Cons of RAG

Now let's talk about cons. Cons is one thing. Obviously the first con is that this requires an infrastructure setup. Infrastructure setup basically means obviously you need to have a vector database and a and all this specific stuff.

The second thing is that sometimes the retrieval the retrieval quality affects results. Like how we are able to retrieve from the vector database that will affect the results.

Third there is always a context window limitation. See every LLM models will be having some kind of context window limitations. So based on that only they'll be able to get the context from the retriever itself. There will be a limitation based on different different models.

Then other than this, you can also think for what all things it is best for. I will keep this as an assignment, but maximum number of use cases like customer support and all can be usually done. Real time data also can be updated in the vector database and based on that you will be able to generate.

Final Summary

So that was more about talking about prompt engineering. So guys I have completely explained the differences between prompt engineering, fine tuning and RAG. What I've done is that one image. Also I've created it over here. Which method you should use. What are the pros and cons. Each and every thing has been mentioned over here. You can just go ahead and see to this, you know um very easily if you're not able to understand my handwriting, the same thing. I've written it over here step by step, which much more proper explanation.

So I hope you like this particular video. I will see you in the next video. Thank you. Take care.