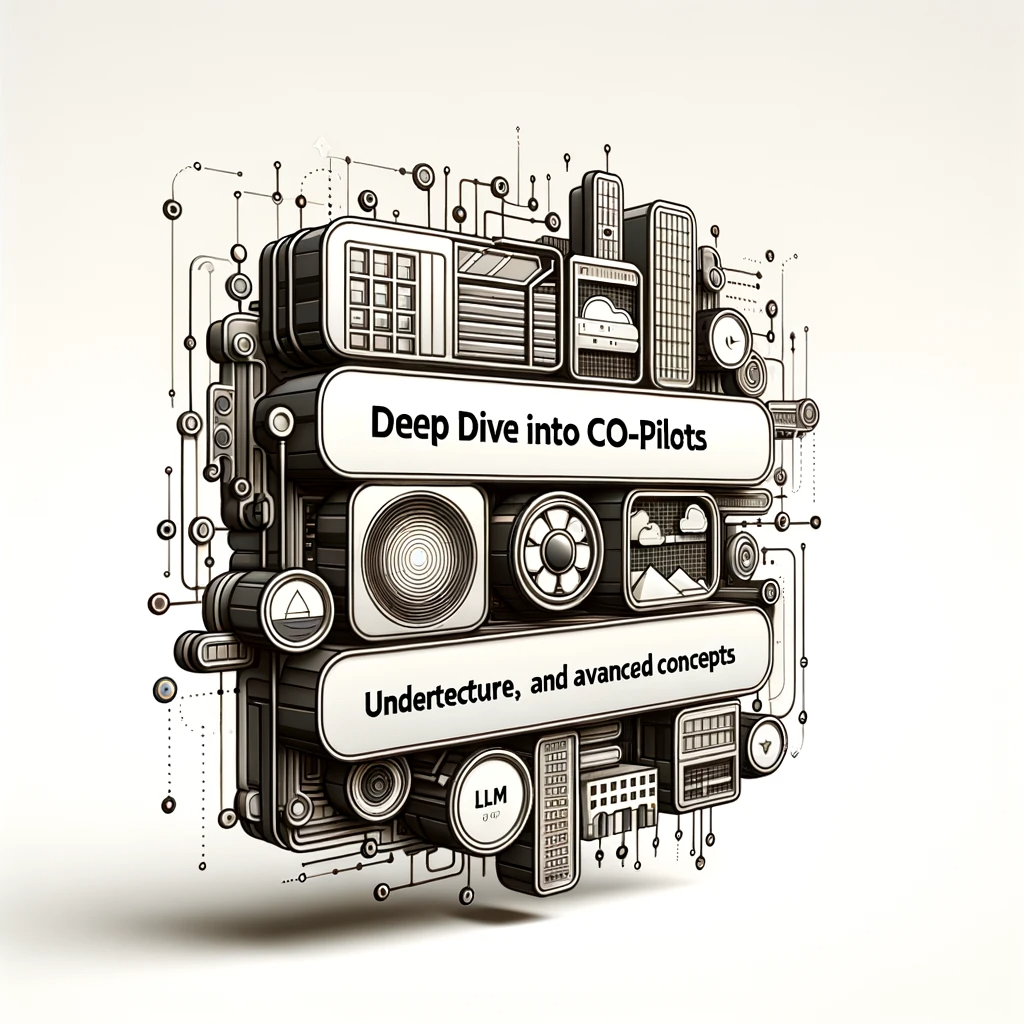
Deep Dive into Co-Pilots: Understanding Architecture, LLMs, and Advanced Concepts

[11. December 2023](https://jannikreinhard.com/2023/12/11/deep-dive-into-co-pilots-understanding-architecture-llms-and-advanced-concepts/) [jannikreinhard](https://jannikreinhard.com/author/jannikreinhard/)[23 Comments](https://jannikreinhard.com/2023/12/11/deep-dive-into-co-pilots-understanding-architecture-llms-and-advanced-concepts/#comments)

In this blog post I want to deep dive with you how LLMs and CoPilots work, want to give an explanations into the most important aspects and show you some important architecture aspects and concepts. We will not build an own Copilot but I will share also some reference architectures and a tool I created to answer your question with informations of you own Intune tenant. Let me know if I should create also an second blog post to go with you step by step through the process of implementing an own bot which use your own data and a LLM model to help you in your daily business working with Intune.



Content

1. [Content](https://jannikreinhard.com/2023/12/11/deep-dive-into-co-pilots-understanding-architecture-llms-and-advanced-concepts/#content)
2. [What is an co pilot?](https://jannikreinhard.com/2023/12/11/deep-dive-into-co-pilots-understanding-architecture-llms-and-advanced-concepts/#what-is-an-co-pilot)
3. [What is an large language model?](https://jannikreinhard.com/2023/12/11/deep-dive-into-co-pilots-understanding-architecture-llms-and-advanced-concepts/#what-is-an-large-language-model)
4. [What is the different between LLM to classical NLP models?](https://jannikreinhard.com/2023/12/11/deep-dive-into-co-pilots-understanding-architecture-llms-and-advanced-concepts/#what-is-the-different-between-llm-to-classical-nlp-models)
5. [What is grounding](https://jannikreinhard.com/2023/12/11/deep-dive-into-co-pilots-understanding-architecture-llms-and-advanced-concepts/#what-is-grounding)
6. [What is prompt engineering?](https://jannikreinhard.com/2023/12/11/deep-dive-into-co-pilots-understanding-architecture-llms-and-advanced-concepts/#what-is-rag)
7. [What is fine tuning?](https://jannikreinhard.com/2023/12/11/deep-dive-into-co-pilots-understanding-architecture-llms-and-advanced-concepts/#what-is-fine-tuning)
8. [Fine tuning vs. grounding](https://jannikreinhard.com/2023/12/11/deep-dive-into-co-pilots-understanding-architecture-llms-and-advanced-concepts/#fine-tuning-vs-grounding)
9. [How can I ground the model with internal knowledge](https://jannikreinhard.com/2023/12/11/deep-dive-into-co-pilots-understanding-architecture-llms-and-advanced-concepts/#how-can-i-ground-the-model-with-internal-knowledge)
10. [How does an more professional concept look like?](https://jannikreinhard.com/2023/12/11/deep-dive-into-co-pilots-understanding-architecture-llms-and-advanced-concepts/#how-does-an-more-professional-concept-look-like)
11. [How can I also integrate a web search to my App?](https://jannikreinhard.com/2023/12/11/deep-dive-into-co-pilots-understanding-architecture-llms-and-advanced-concepts/#how-can-i-also-integrate-a-web-search-to-my-app)
12. [How can I build an solution together with Intune](https://jannikreinhard.com/2023/12/11/deep-dive-into-co-pilots-understanding-architecture-llms-and-advanced-concepts/#how-can-i-build-an-solution-together-with-intune)
13. [How does the architecture form the M365 CoPilot looks like?](https://jannikreinhard.com/2023/12/11/deep-dive-into-co-pilots-understanding-architecture-llms-and-advanced-concepts/#how-does-the-architecture-form-the-m365-copilot-looks-like)
14. [Let me explain the CoPilot stack](https://jannikreinhard.com/2023/12/11/deep-dive-into-co-pilots-understanding-architecture-llms-and-advanced-concepts/#let-me-explain-the-copilot-stack)
15. [What is orchestration?](https://jannikreinhard.com/2023/12/11/deep-dive-into-co-pilots-understanding-architecture-llms-and-advanced-concepts/#what-is-orchestration)
16. [What are the most common frameworks?](https://jannikreinhard.com/2023/12/11/deep-dive-into-co-pilots-understanding-architecture-llms-and-advanced-concepts/#what-are-the-most-common-frameworks)
    1. [LangChain](https://jannikreinhard.com/2023/12/11/deep-dive-into-co-pilots-understanding-architecture-llms-and-advanced-concepts/#langchain)
    2. [Semantic Kernel](https://jannikreinhard.com/2023/12/11/deep-dive-into-co-pilots-understanding-architecture-llms-and-advanced-concepts/#semantic-kernel)
    3. [Comparison](https://jannikreinhard.com/2023/12/11/deep-dive-into-co-pilots-understanding-architecture-llms-and-advanced-concepts/#comparison)
17. [Conclusion](https://jannikreinhard.com/2023/12/11/deep-dive-into-co-pilots-understanding-architecture-llms-and-advanced-concepts/#conclusion)

What is an co pilot?

Lets make it easy and ask Chat GPT:

*Microsoft Copilot is an AI-powered tool designed to assist developers in writing and maintaining code. It is integrated into various Microsoft products, including Visual Studio and GitHub, where it’s known as GitHub Copilot. This tool leverages machine learning models, trained on a vast corpus of source code, to provide suggestions for code completion, debugging, and even writing whole blocks of code.*

This is not completely correct. But this is also important to understand. Not all content you got from GPT is always correct. You should always verify and also make double checks bevor you share something. Let me define what it is. For me an CoPilot redefine the way how we interact with the Internet, with our PC as well with Tools and content. It can for example support you to research things in the Internet generate presentations or building automations and code.

An CoPilot will talk to you like an real Person and is integrated in Tools directly where you need them. Sometimes they can also take actions, like the Windows CoPilot who can start Apps or Timers for you and helps to work interact with the OperatingSystem. An CoPilot has an deep knowledge and is grounded with necessary informations.

Mo make this more clear for you how big the hype is here some facts. It took 16 years for mobile phones to reach the level of 100 mio. users, 30 month for Instagram and 78 month for google translator. For ChatGPT it was 2,5 month.

One of the first CoPilots in the field was the GitHub CoPilot who supported in writing code.



What is an large language model?

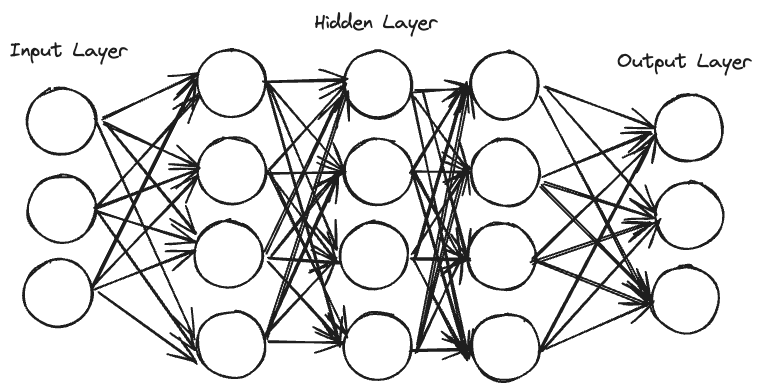
An large language model is an transformer model which is developed based on the transformer architecture. So now you know what it is. Lets go to the next chapter….. Okay lets break it down to make it more clear.

**Breaking Down the Basics**

Firstly, it’s important to understand what a transformer model is. The transformer model, introduced in a groundbreaking paper titled “Attention Is All You Need” in 2017, revolutionised the field of natural language processing (NLP). This model relies heavily on a mechanism known as ‘attention’, allowing it to weigh the significance of different words in a sentence. This approach differs with previous models that processed words in a linear or sequential manner, often missing the broader context of a sentence or paragraph.

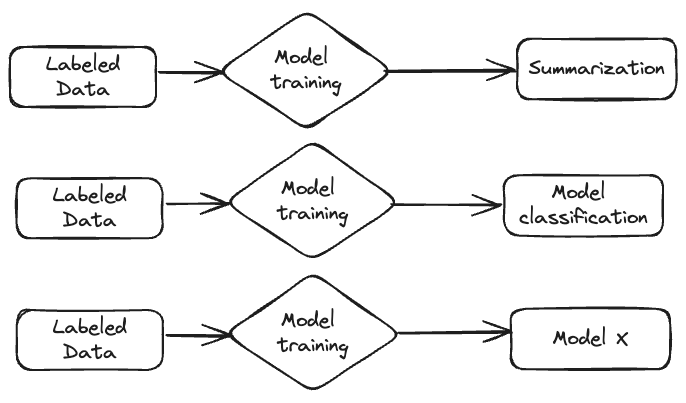
Large language models, as the name suggests, are massive in scale. They are trained on vast amounts of data, sometimes encompassing a significant portion of the publicly available date from the internet or other sources. This extensive training enables these models to generate, understand, and respond to human language in a way that is remarkably coherent and contextually relevant.

An LLM is in the end a transformer model neural network which has an input layer, multiple hidden layers and an output layers. The size of LLMs will be defined with the count of parameters. GPT2 has 1.5B, GPT3 has 175B parameters and GPT4 has 1.4T parameters. Here you can also see the tremendous speed in the development and evaluation. The training needs a lot and a lot means really a lot of power especially GPU power.

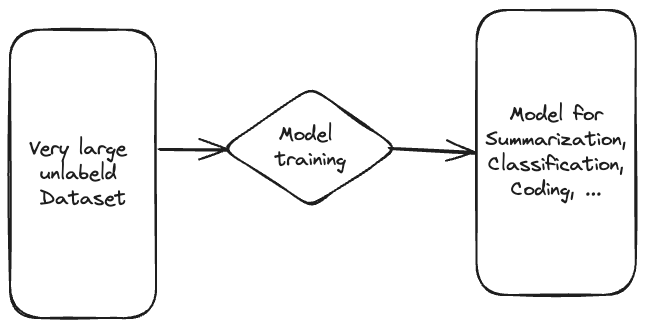


What is the different between LLM to classical NLP models?

Classical NLP models such as text classification and detection models must be individually developed and trained. They performed rally good for single task where they where optimised.



This is totally different for the LLMs they was trained with very large data sets and with this they have a very large and wide understanding, can handle different media types like picture, text and sound and can with this used for a lot of different use cases without adapting them or make individual training.



What is grounding

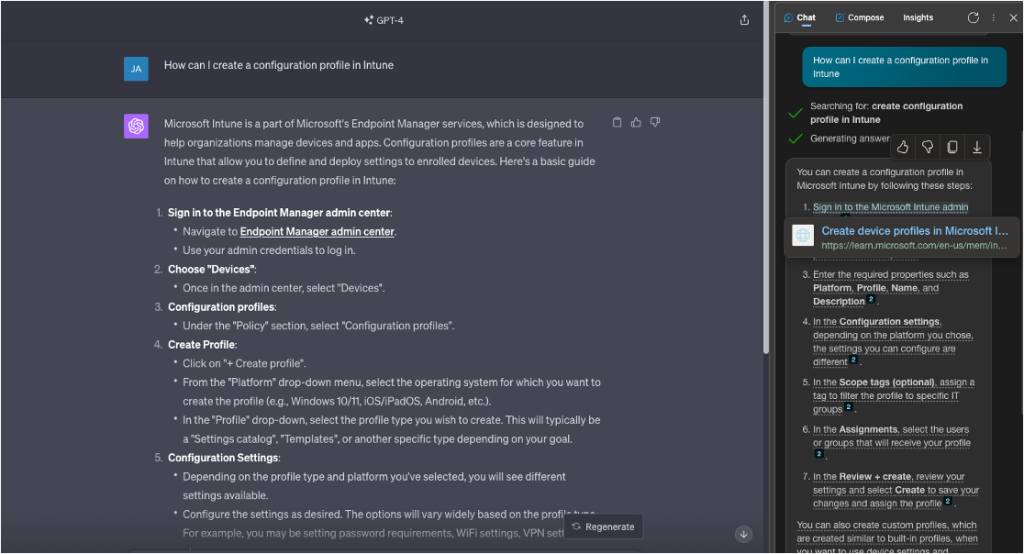
It is very important that you not rely on the domain knowledge of an model. Why? It is outdated, not always correct, you can have hallucination, it does not know you internal processes and guidelines. Enough arguments. We talking here about an Large Language Model which is optimised to understand and speech a humen like language but is is not an all what you want to know repository. But how can you answer questions otherwise? The answer is grounding.

Let me explain what it is.

Grounding in the context of a Large Language Model (LLM) like GPT refers to the model’s ability to connect and apply its vast knowledge and language understanding to real-world contexts and specific situations. In this context you often hear Retrieval Augmented Generation (RAG). This involves integrating external data sources or user-provided information to generate responses that are relevant, accurate, and appropriately contextualised. Essentially, grounding helps the LLM to understand and respond to queries not just based on its training data, but also considering the current context or specific details provided during the interaction.

Here is an very good example. I ask chat GPT: “How can I create a configuration profile in Intune”. The answer I got starts with Navigate to “Endpoint Manager admin center”. As you know Microsoft loves to rename his products. The Name is Intune Admin Center and not longer Endpoint Manager.

I ask the same question in the Bing chat and got an Answer with he Microsoft Intune admin center and also an link to the source where this information came form. Bing works on the way that is first run a web search to ground the model and then it answer the question.



What is prompt engineering?

Before we go into the prompt engineering lets first clarify what is an prompt. A prompt is a short instruction or description given to an LLM or other system to initiate a specific action or response. It’s usually a text input that guides the AI on what to do or create, like asking a question or describing an image to be generated. With the new GPT-4v it is also possible to input an image.

So now to prompt engineering. Prompt engineering is in the end an skill for crafting prompts to effectively communicate with the LLM systems. This involves understanding how an AI interprets and responds to different wordings, and optimizing the prompt to achieve the desired outcome. It’s especially important in contexts where precise or nuanced responses are needed, and can involve experimenting with various phrasings, structures, and keywords to guide the AI towards producing more accurate, relevant, or creative results.

I can talk hours and weeks about this topic. I will create an dedicated blog to explain prompt engineering more in detail and give you an getting start guide.

What is fine tuning?

Fine-tuning is the the process of taking a pre-trained model and further training it on a specific, often smaller, dataset. This specialised training allows the model to adapt its knowledge and capabilities to a particular domain or task. An typical dataset for this is e.g. question and answer. Fine tuning is mostly expensive and needs also many resources to do this.

Fine tuning vs. grounding

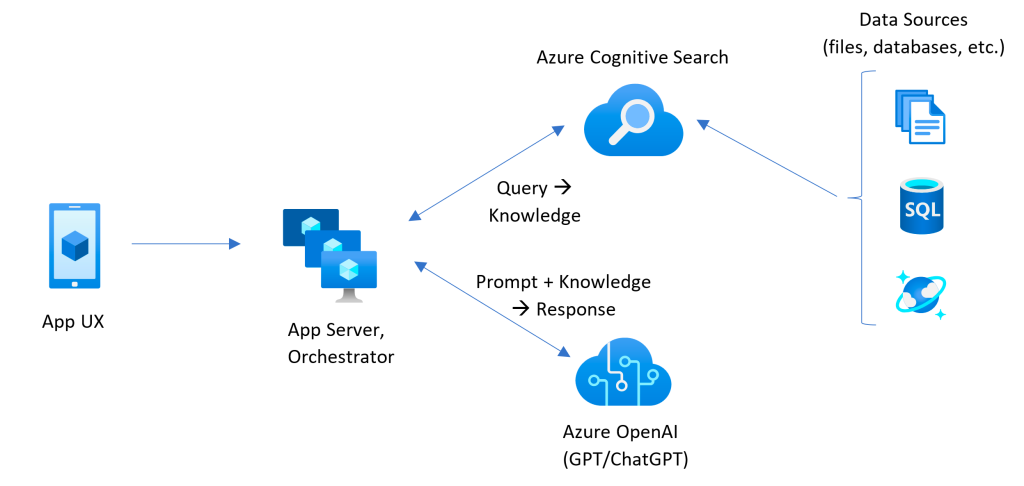
There is an clear hierarchy how you bring context into an large language model. I would say 90% of the use cases can be fulfilled only with grounding and prompt engineering of the prompts. If this for some reassones (I will specify soon) not work than you can fine tune the model. If this also not work than you have the possibility to train an on model. With this steps the complexity, price, maintenance and many other factors will be increase. For this my clear advice is always try to solve your use case without fine-tuning. Also all the co pilots working with basic fondation models.

| **Criteria** | **LLM Fine-Tuning** | **Prompt Grounding** |
| --- | --- | --- |
| **Definition** | Fine-tuning involves training an LLM on a specific dataset to adapt its responses. | Prompt grounding is adding context into the prompt. |
| **Data Requirement** | High. Requires a dataset relevant to the specific task or domain for training. | Depends. It relies on the existing knowledge and capabilities of the LLM but with context/data in the prompt. |
| **Flexibility** | Once fine-tuned, the model is specialized in the trained domain. If the data change a new fine tuning is needed. | Highly flexible; the approach can be modified for different tasks or domains simply by changing the prompt. |
| **Cost and Resources** | Resource-intensive as it involves additional training, which requires computational power and time. | Cost-effective and less resource-intensive since it doesn’t require additional model training. |
| **Customization Level** | Allows for deep customization as the model learns specific patterns, styles, or information from the training data. | Limited customization compared to fine-tuning; the model’s response is only as good as the prompt’s design. |
| **Time to Implement** | Can be time-consuming due to the need for training and model optimization. | Quick to implement, as it only requires crafting a suitable prompt. |
| **Scalability** | Scalable but may require retraining or additional fine-tuning for different domains or tasks. | Highly scalable as different prompts can be easily created and applied for various tasks. |
| **Use Cases** | Ideal for specialized tasks where in-depth knowledge or a specific response style is essential. | Best suited for general-purpose tasks or when needing to quickly adapt to different requirements without additional training. |

How can I ground the model with internal knowledge

There are multiple ways how you can do this. The easies way is to copy text or the relevant content from an document or something else into your prompt. This will work for single queries but as soon as you build an full application on top of this you have to find an better way.

One whey how you can do this is to play you documents and content into an blob storage and using Azure Ai Search to index your documents. When an User ask a question in your Application the the first thing is that the question will send to the AzureAI search which try to find the relevant content to answer the question. After wards this content will be send together with an prompt and the question to the GPT Service. There is an very nice reference architecture from Microsoft:



When you want to setup this solution without coding you can do this really easy via the OpenAi Studio.

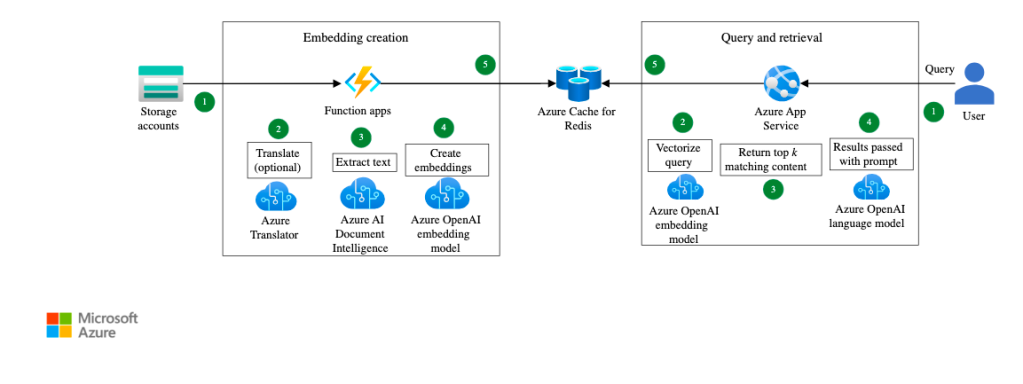
The only thing what you have to to is to write an system prompt to instruct the model and create a connection to you AzureAi Search. You can also provision this Service via this wizard. Then you will find an deploy button at the Top which deploys also the service for you to an AppService. Thats it. Isn’t it easy to build your own CoPilot?

How does an more professional concept look like?

In this concept you create a embedding of your content via a embedding model. Here you can also pre transform your data like translate, standardised, getting information from docs and so on. The this data will be stored in an vector data base. This is now a simplified description how to store the knowledge.

When the user has a question now this question will be embedded and a vectorise query will be run on the vector db. The best fitting content will be gathered and send as grounding together with the question to the large language model to answer the question as precise as possible.

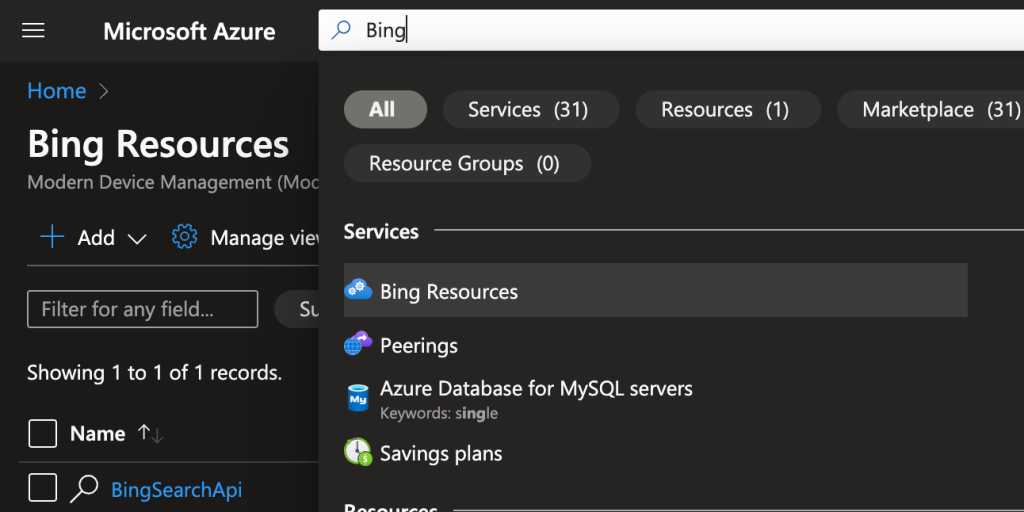
Here you can find an example architecture form microsoft how this could work. This design use a azure function and an azure app service together with different azure ai services (former cognitive services).



[Source](https://learn.microsoft.com/en-us/azure/architecture/ai-ml/openai/idea/search-and-query-using-openai-service)

How can I also integrate a web search to my App?

The answer is very easy. Use Bing! In azure you can find an Service called “Bing Resources”. This service provides you an API for bing. Here you can send you question/prompt/search term to bing and you get an result from near live content from the internet.



How can I build an solution together with Intune

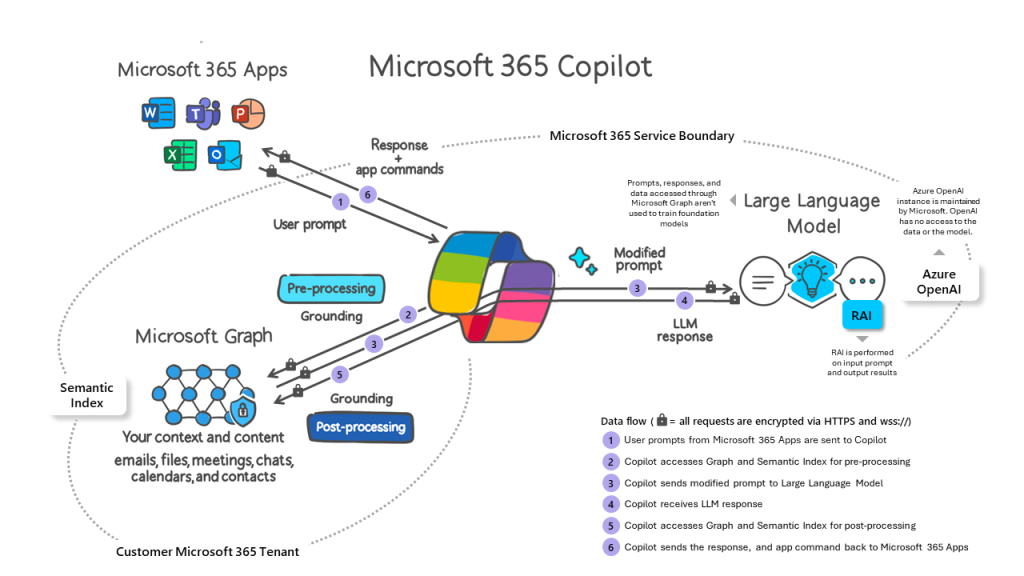
Check out my blog post where I develop a tool (GPT Device Troubleshooter) which combines Intune and GPT.

<https://jannikreinhard.com/2023/09/10/the-magic-of-the-gpt-intune-device-troubleshooter/>

How does the architecture form the M365 CoPilot looks like?

graph, graph and graph! It is the source of everything. When you run an prompt in one of the office product it goes to the M365 chat engine which detects your intent and make an query to graph in the context of the user to ground the prompt. Afterwards it build an modified prompt with templates and send it to the OpenAi Service. This service answers your question afterwards there is also an post processing to attach for example also raw data to the answer like a file or a list of data. This bundle will be send back to the user.

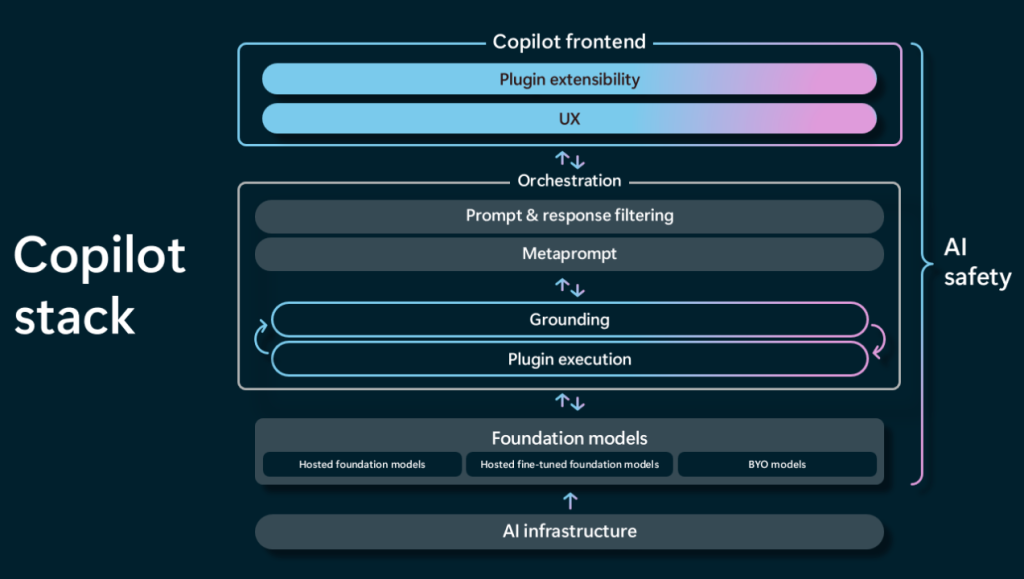
Here you can see the official picture from microsoft



This was an very simplified explanation how the thinks work. You know that I like deep dives. Tbh. the whole thing is a blackbox but lets me if you want me to show you also an architecture how this could look like and which you can use when you want to build such an comprehensive bot.

Let me explain the CoPilot stack

The Copilot contains multiple layers. The first layer is the azure infrastructure and the foundational model as such. On top of this there is a AI orchestration. We will talk more about this in the next chapter what it is in detail but in short it is the hard if the CoPilot. On top of it it is the frontend with the possibility to extend the copilot with additional Add-Ons/Skills/Plugins name it as you want. This add ons has also an component within the engine.



What is orchestration?

In summary, orchestration is about seamlessly integrating various components and services, managing tasks, making informed decisions, interacting effectively with users, and continually learning and adapting to provide the best possible assistance.

The main aspects are:

1. **Integration of Services**: The AI co-pilot may need to integrate various services and data sources. For instance, in a driving context, like integration to graph, bing or in general all APIs and Databases.
2. **Task Management**: Orchestration involves managing and prioritising tasks. Tasks could be getting data, sending prompts or taking action.
3. **Decision Making**: An AI co-pilot would use orchestration to make decisions based on the data it gathers and processes. It could be e.g. to decide which action it should propose to an user.
4. **User Interaction**: Efficient orchestration also includes how the system interacts with the user, ensuring that communication is clear, timely, and relevant.

Mostly a LLM framework is used to do the orchestration

What are the most common frameworks?

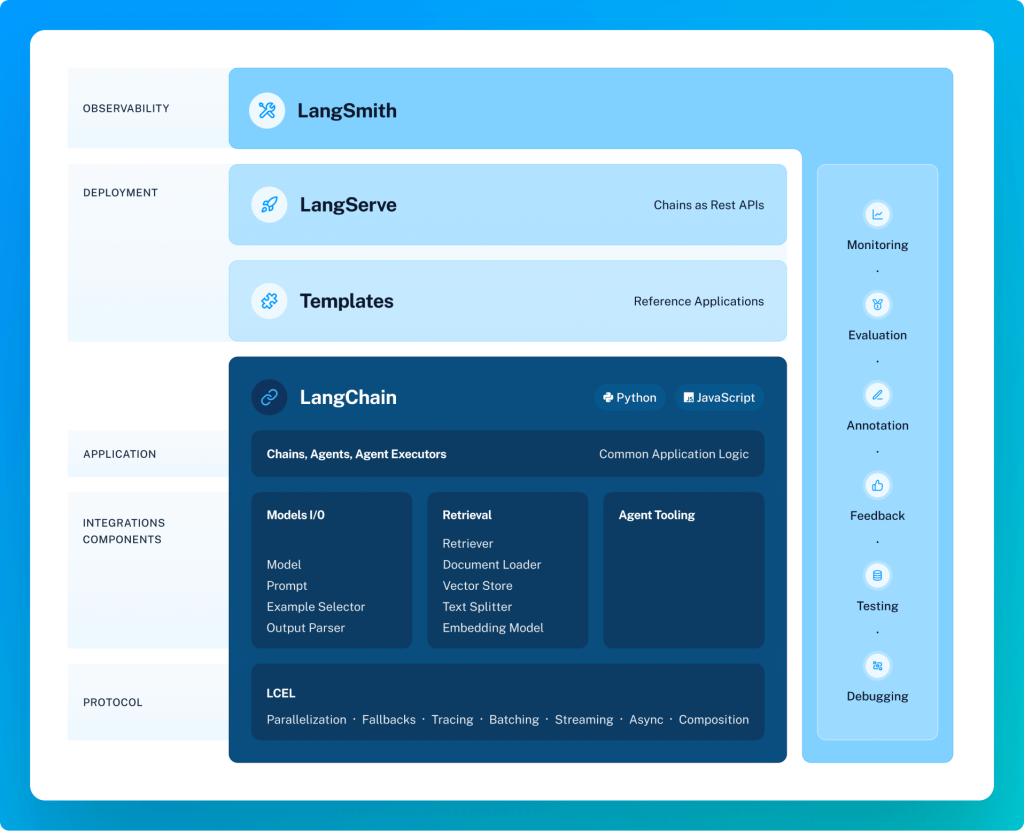
There many frameworks but I would say the most common ones are LangChain and Semantic Kernel. Both are OpenSource.

LangChain

Lang chain is a community project on Github which is helpful for creating chatbots, summarizing documents, or analyzing code. Here’s how it works:

1. **Model I/O:** Connects with language models.
2. **Retrieval:** Gathers specific data needed for the task.
3. **Chains:** Links different steps or actions together.
4. **Agents:** Decides which tools to use based on what you want to do.
5. **Memory:** Remembers what has happened in previous interactions.
6. **Response:** Gives back the final result.

LangChain is flexible and supports different programming languages like Python and JavaScript. It’s like having a smart assistant that can use different tools and remember past conversations to help you build complex language-based applications.

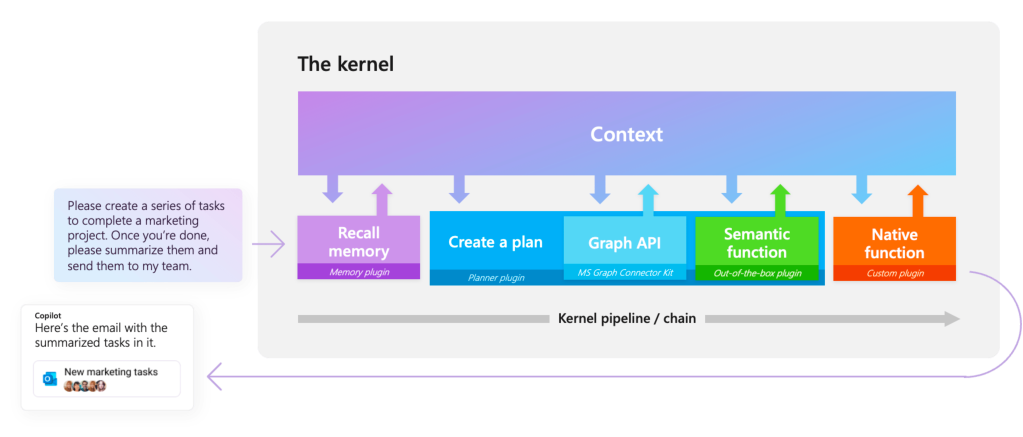


Semantic Kernel

This is also an open source project driven and used by microsoft. It’s a simpler tool that helps you integrate LLM with common programming languages (like C# and Python). Here’s how it operates:

1. **Kernel:** Manages and organizes your tasks.
2. **Memories:** Stores and recalls information.
3. **Planner:** Chooses the right tools (or plugins) for the job.
4. **Connectors:** Links to external data sources.
5. **Plugins:** Adds custom functions in C# or Python.
6. **Response:** Delivers the outcome.

Semantic Kernel is lightweight and focuses on seamlessly integrating different AI services into your existing applications. It’s like adding a smart layer to your apps, allowing them to remember past interactions and use various AI tools effectively.



Comparison

* **Flexibility:** LangChain offers more flexibility with more out-of-the-box tools and integrations, especially for Python and JavaScript developers. Semantic Kernel is more streamlined, focusing on integrating AI services into existing applications, particularly for C# and Python.
* **Use Cases:** LangChain is great for more complex tasks that need remembering past interactions and linking various steps. Semantic Kernel is ideal for simpler, more direct integrations of AI into existing apps.
* **Community:** LangChain has a strong community and support, especially since it’s exist longer and has been used in various applications. Semantic Kernel, being relatively newer, is growing its community and support, especially among developers looking to integrate AI into enterprise applications.

This is only a short comparison but there are also much more aspects you should consider. The best is if you try out both.

Conclusion

Hope this brings some more light into the dark. Please let me know if you want to have some more deep dives for a specific topic and I am happy to blog about this.