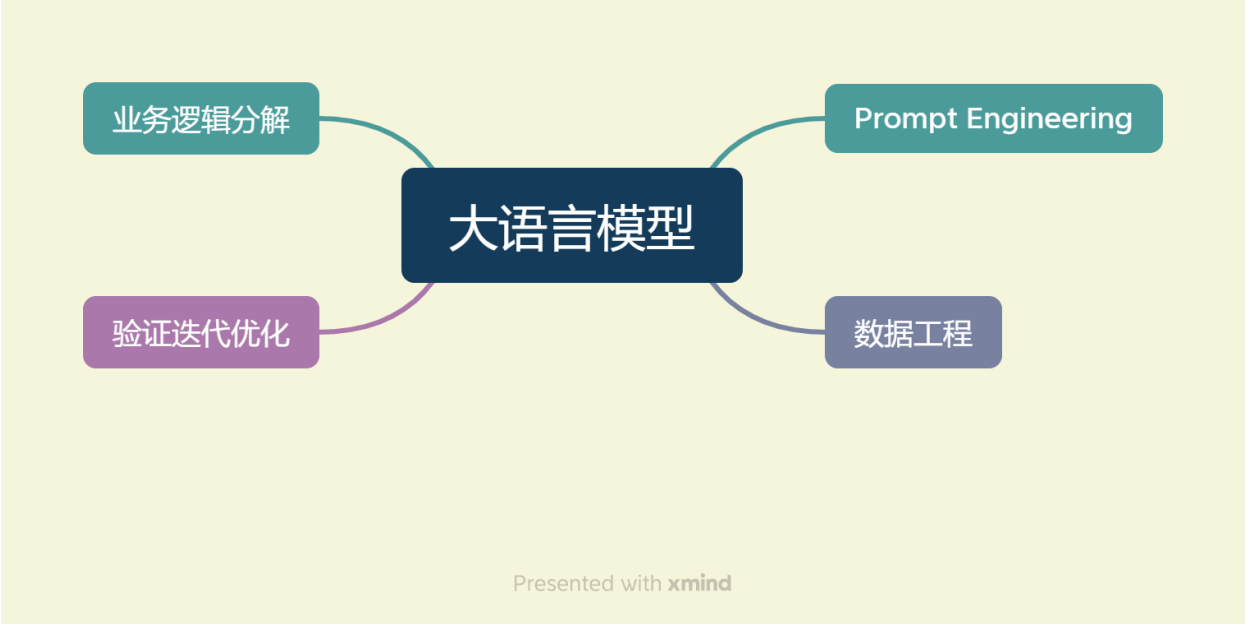
3316 words | 9 minutes

The overall process of developing LLM applications

1. What is large model development?

We call **the development of applications that use big language models as the core functions, use the powerful understanding and generation capabilities of big language models, and combine special data or business logic to provide unique functions** as big model development . Although the core technology of developing big model-related applications is on big language models, they are generally achieved by calling APIs or open source models to achieve core understanding and generation, and by prompt engineering to achieve control of big language models. Therefore, although big models are the culmination of deep learning, big model development is more of an **engineering problem** .

In the development of large models, we generally do not make major changes to the model, but **use the large model as a calling tool, and use prompt engineering, data engineering, business logic decomposition and other means to give full play to the capabilities of the large model and adapt to application tasks** , rather than focusing on optimizing the model itself. Therefore, as beginners in the development of large models, we do not need to deeply study the internal principles of the large model, but rather need to master the practical skills of using the large model.



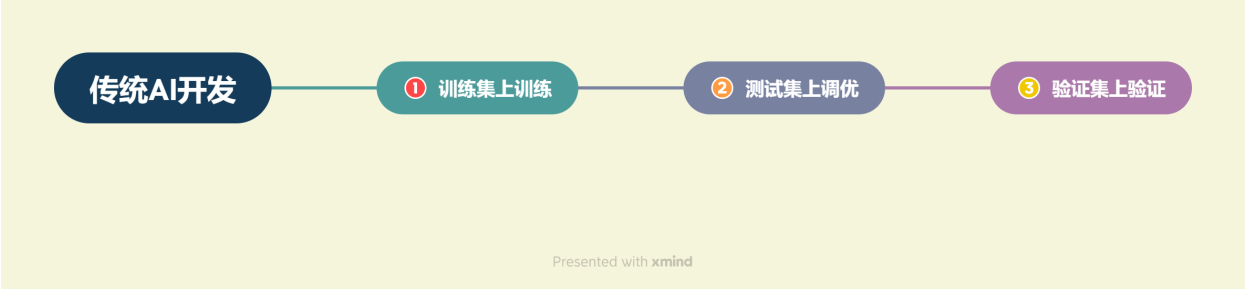
Large Model Development Elements

At the same time, the overall idea of large model development, which focuses on calling and utilizing large models , is quite different from that of traditional AI development . The two core capabilities of large language models are: 指令遵循 providing 文本生成 a simple alternative to complex business logic.

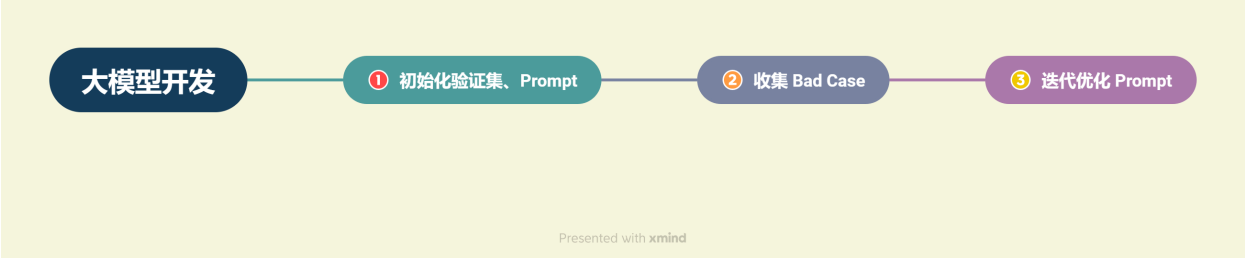
- 传统的 AI 开发 : First, you need to disassemble the very complex business logic one by one, construct training data and verification data for each sub-business, train and optimize the model for each sub-business, and finally form a complete model chain to solve the entire business logic.
- 大模型开发 ： Use Prompt Engineering to replace sub-model training and tuning, implement business logic through Prompt link combination, use a general large model + several business prompts to solve the task, thereby transforming traditional model training and tuning into a simpler, easier and lower-cost Prompt design and tuning.

At the same time, in terms of **evaluation ideas** , there are qualitative differences between large model development and traditional AI development.

- 传统 AI 开发 : You need to first construct a training set, a test set, and a validation set, and then evaluate the performance by training the model on the training set, fine-tuning the model on the test set, and finally verifying the model effect on the validation set.
- 大模型开发 ： The process is more flexible and agile. Construct a small batch validation set based on actual business needs, and design a reasonable prompt to meet the validation set effect. Then, continuously collect the bad cases of the current prompt from the business logic, add the bad cases to the validation set, optimize the prompt in a targeted manner, and finally achieve a better generalization effect.



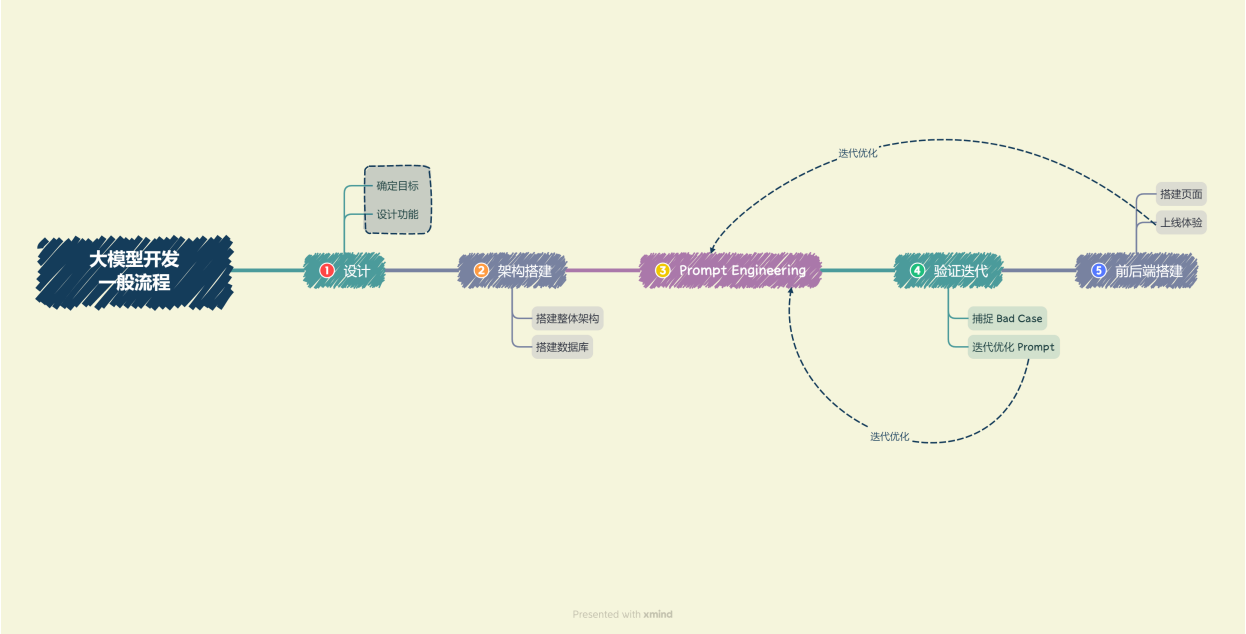
Traditional AI Evaluation



In this chapter, we will briefly describe the general process of large model development, and combine it with the actual needs of the project to gradually analyze the work and steps to complete the project development.

2. General process of large model development

Combined with the above analysis, we can generally decompose large model development into the following processes:



1. **Determine the goal** . Before development, we first need to determine the development goal, that is, the application scenario, target population, and core value of the application to be developed. For individual developers or small development teams, they should generally set a minimum goal first, starting with building an MVP (minimum viable product), and gradually improve and optimize.

2. **Design functions** . After determining the development goals, it is necessary to design the functions that the application will provide, as well as the general implementation logic of each function. Although we have simplified the decomposition of business logic by using large models, clearer and deeper understanding of business logic can often lead to better Prompt effects. Similarly, for individual developers or small development teams, the first thing to do is to determine the core functions of the application, and then extend the design of upstream and downstream functions of the core functions; for example, if we want to create a personal knowledge base assistant, then the core function is to answer questions based on the content of the personal knowledge base, then the upstream function of users

uploading knowledge bases and the downstream function of users manually correcting model answers are sub-functions that we must also design and implement.

3. **Build the overall architecture** . At present, most large model applications use the architecture of specific database + prompt + general large model. We need to build the overall architecture of the project for the functions we designed, and realize the whole process from user input to application output. Generally speaking, we recommend development based on the LangChain framework. LangChain provides the implementation of Chain, Tool and other architectures. We can customize it based on LangChain to realize the overall architecture connection from user input to database to large model and final output.

4. **Build a database** . Personalized large model applications need to be supported by a personalized database. Since large model applications require vector semantic retrieval, vector databases such as Chroma are generally used. In this step, we need to collect data and preprocess it, and then vectorize and store it in the database. Data preprocessing generally includes conversion from multiple formats to plain text, such as PDF, MarkDown, HTML, audio and video, as well as cleaning of erroneous data, abnormal data, and dirty data. After preprocessing, it is necessary to slice and vectorize to build a personalized database.

5. **Prompt Engineering** . High-quality prompts have a great impact on the capabilities of large models. We need to gradually and iteratively build high-quality prompt engineering to improve application performance. In this step, we should first clarify the general principles and techniques of prompt design, build a small validation set from actual business, and design a prompt based on the small validation set that meets basic requirements and has basic capabilities.

6. **Verification iteration** . Verification iteration is an extremely important step in large model development. It generally refers to improving system effects and dealing with boundary conditions by constantly discovering bad cases and improving prompt engineering in a targeted manner. After completing the initial prompt design in the previous step, we should conduct actual business tests, explore boundary conditions, find bad cases, and analyze the problems of prompts in a targeted manner, so as to continuously iterate and optimize until a relatively stable prompt version that can basically achieve the goal is reached.

7. **Build the front-end and back-end** . After completing Prompt Engineering and its iterative optimization, we have completed the core functions of the application and can give full play to the powerful capabilities of the large language model. Next, we need to build the front-end and back-end, design the product page, and make our application go online as a product. Front-end and back-end development is a very classic and mature field, so I will not go into details here. We use Gradio and Streamlit to help individual developers quickly build visual pages to realize the Demo online.

8. **Experience optimization** . After completing the front-end and back-end construction, the application can be put online for experience. Next, it is necessary to conduct long-term user experience tracking, record bad cases and user negative feedback, and then make targeted optimizations.

3. Brief analysis of the process of building an LLM project (taking the knowledge base assistant as an example)

Below we will combine this practical project with the overall process introduction above to briefly analyze **the knowledge base assistant project** development process:

Step 1: Project planning and demand analysis

1. Project goal : Question-answering assistant based on personal knowledge base

2. Core Functions

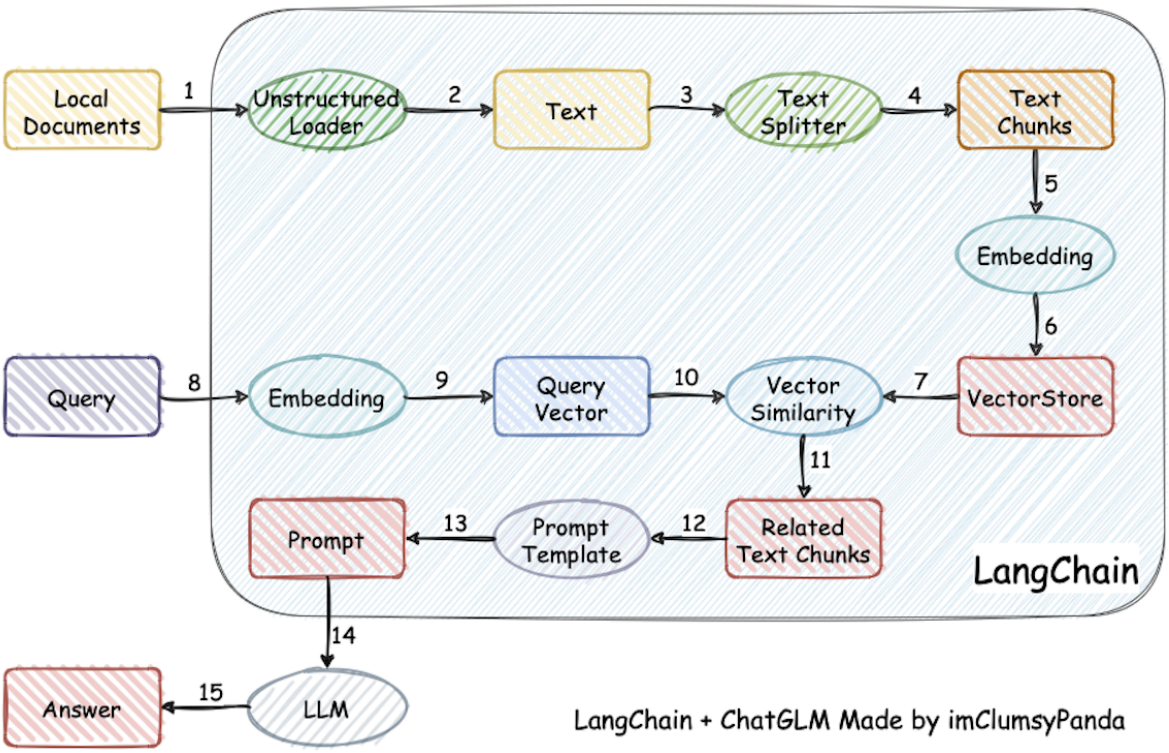
1. Vectorize the crawled and summarized MarkDown files and user-uploaded documents, and create a knowledge base;
2. Select a knowledge base and retrieve the knowledge fragments asked by the user;
3. Provide knowledge snippets and questions to get answers from the big model;
4. Streaming reply;
5. Historical conversation records

3. Determine technical architecture and tools

1. **Framework** : LangChain
2. **Embedding models** : GPT, Zhipu, **M3E**
3. **Database** : Chroma
4. **Large models** : GPT, iFlytek Spark, Wenxin Yiyan, GLM, etc.
5. **Front and back end** : Gradio and Streamlit

Step 2: Data preparation and vector knowledge base construction

The implementation principle of this project is shown in the figure below (image source): load local document -> read text -> text segmentation -> text vectorization -> question vectorization -> match the top k most similar to the question vector in the text vector -> add the matched text as context and question to the prompt -> submit to LLM to generate an answer.



1. Collect and organize documents provided by users

Common document formats used by users include PDF, TXT, MD, etc. First, we can use LangChain's document loader module to easily load documents provided by users, or use some mature Python packages to read them.

Due to the limitation of using tokens in current large models, we need to segment the read text and divide longer text into smaller texts. At this time, a piece of text is a unit of knowledge.

2. Vectorize document words

`文本嵌入(Embeddings)技术` The segmented documents are vectorized so that semantically similar text segments have close vector representations. Then, they are stored in the vector database to complete the `索引(index)` creation of .

By using the vector database to index each document fragment, fast retrieval can be achieved.

3. Import the vectorized documents into the Chroma knowledge base and create a knowledge base index

Langchain integrates with over 30 different vector databases. The Chroma database is lightweight and the data is stored in memory, which makes it very easy to launch and start using.

The user's knowledge base content is stored in the vector database through Embedding, and then every time the user asks a question, it will also go through Embedding. The vector correlation algorithm (such as the cosine algorithm) is used to find the most matching knowledge base fragments. These knowledge base fragments are used as context and submitted to the LLM as a prompt together with the user's question for answer.

Step 3: Large model integration and API connection

1. Integrate large models such as GPT, Spark, Wenxin, GLM, etc., and configure API connections.
2. Write code to interact with the big model API to get answers to your questions.

Step 4: Core Function Implementation

1. Build Prompt Engineering to implement large-model answering capabilities and generate answers based on user questions and knowledge base content.
2. Implement streaming replies, allowing users to have multiple rounds of conversations.
3. Add the historical conversation record function to save the interaction history between the user and the assistant.

Step 5: Iterative optimization of core functions

1. Conduct verification and evaluation and collect Bad Cases.
2. Iterate and optimize the core function implementation according to Bad Case.

Step 6: Front-end and user interface development

1. Use Gradio and Streamlit to build the front-end interface.
2. Realize the function of users uploading documents and creating knowledge base.
3. Design the user interface, including question input, knowledge base selection, history display, etc.

Step 7: Deployment, testing and launch

1. Deploy the Q&A Assistant to a server or cloud platform and ensure that it is accessible on the Internet.
2. Conduct production environment testing to ensure system stability.

3. Go online and release to users.

Step 8: Maintenance and Continuous Improvement

1. Monitor system performance and user feedback, and handle issues in a timely manner.
2. The knowledge base is updated regularly with new documents and information.
3. Collect user needs and make system improvements and function expansions.

The entire process will ensure that the project can proceed smoothly from planning, development, testing to launch and maintenance, providing users with high-quality question-and-answer assistants based on personal knowledge bases.

> **Now that we have a preliminary understanding of the general process of large model development, we will introduce the entire development environment to ensure that everyone can carry out project development smoothly.**
>
> - If you are an old hand, you can skip the subsequent content of this chapter and go directly to the second part.
> - The next two chapters mainly introduce the construction of two development environments for students who do not have a suitable development environment. You can read them as needed.
>   - Chapter 5 mainly introduces `阿里云服务器的基本使用` ,, `通过 SSH 远程连接服务器` and `jupyter notebook 的使用` .
>   - Chapter 6 mainly introduces `GitHub CodeSpace` the use of GitHub and how to `GitHub CodeSpace` build a development environment in GitHub. (First, make sure you have a network environment that can smoothly access GitHub. Otherwise, it is recommended to use Alibaba Cloud.)
> - If you already have a suitable development machine, you can jump directly to `7.环境配置` the chapter and start configuring the development environment.

Previous chapter
3. Introduction to LangChain

5. Alibaba Cloud Server Usage Guide