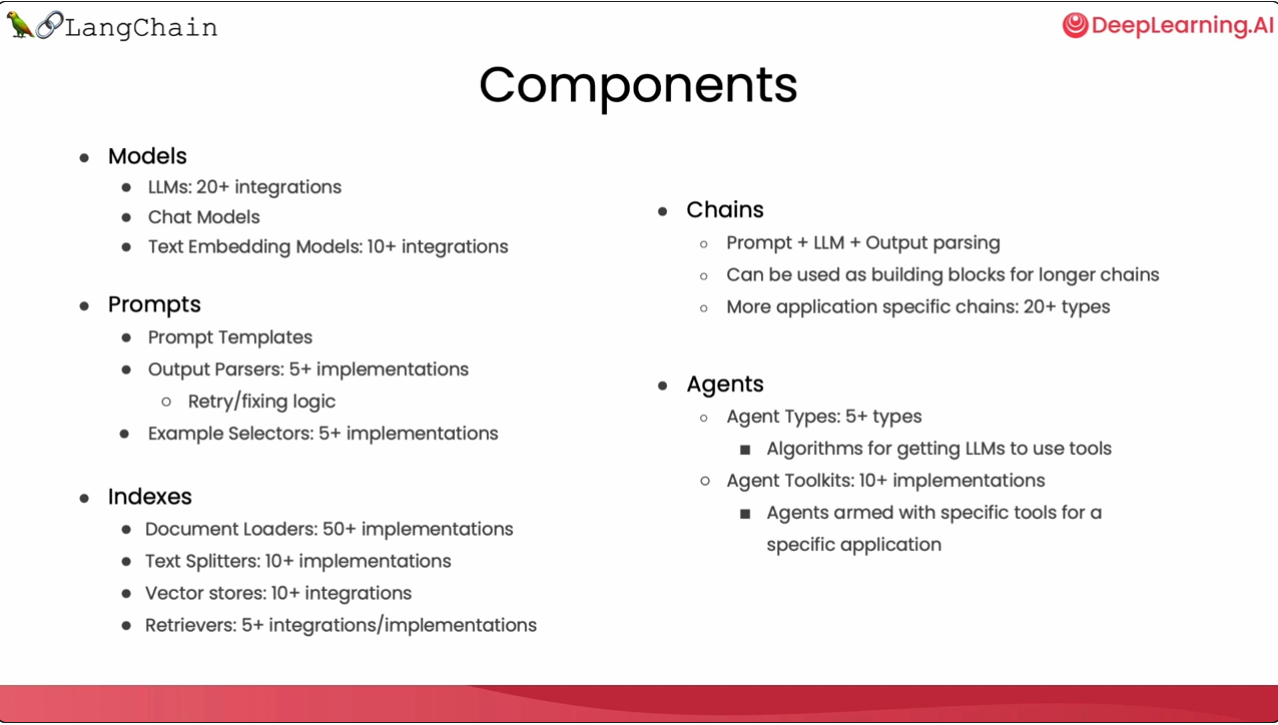


**Langchain:**

LangChain is a framework designed to simplify the development of applications that integrate large language models (LLMs) like those provided by OpenAI, Hugging Face, and other AI platforms. It provides a modular and flexible structure to build, deploy, and manage complex language model-driven workflows and applications. LangChain abstracts away much of the complexity involved in integrating and orchestrating various components such as document loaders, vector stores, language models, prompts, and chains.



**Core Components:**

1. **Document Loaders:**

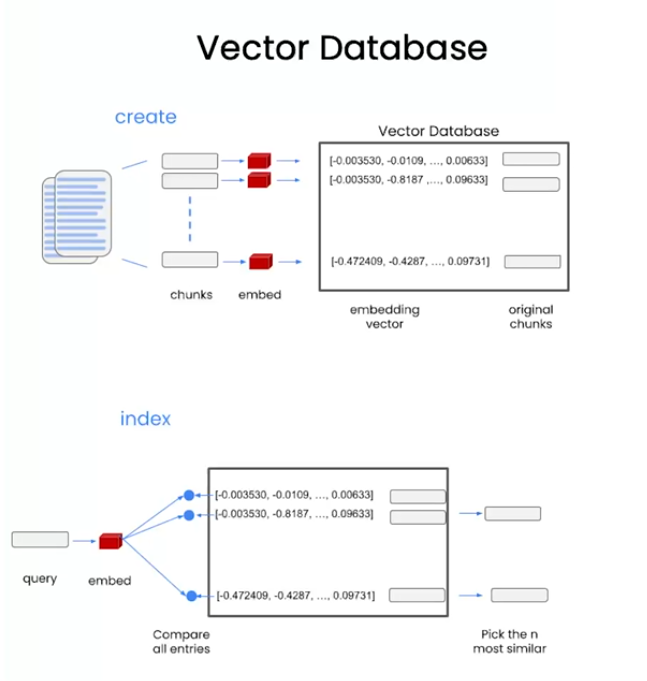
Classes like PyPDFLoader and PyPDFDirectoryLoader help in loading and reading documents from various sources (e.g., PDF files).

1. **Text Splitters:**

Classes such as RecursiveCharacterTextSplitter are used to split large documents into smaller chunks for easier processing.

1. **Vector Stores:**

Stores like FAISS are used to store and retrieve embeddings for documents, facilitating efficient similarity searches.

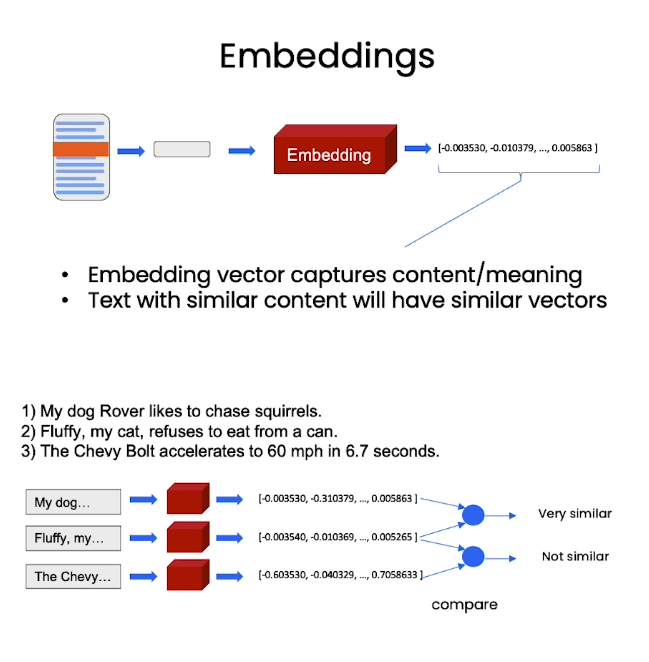


1. **Embeddings:**

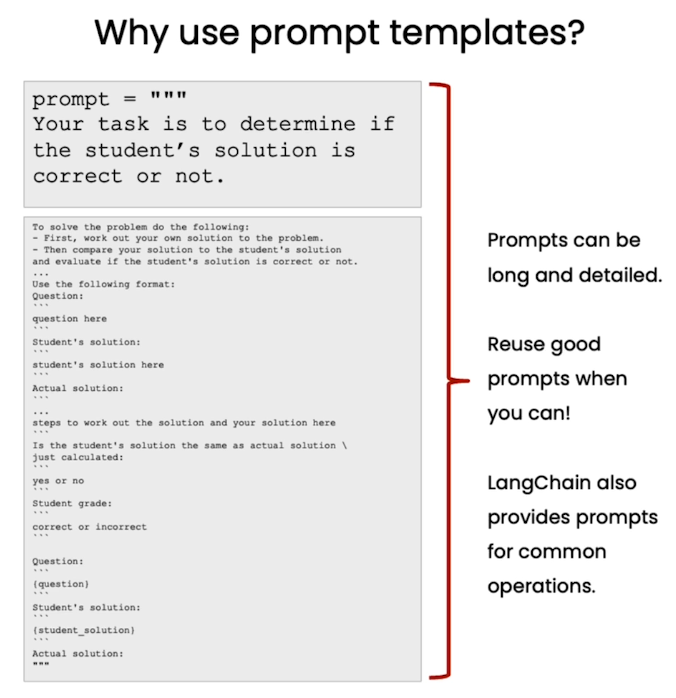
An embedding is a numerical representation of data, often used in the context of natural language processing (NLP) and machine learning. Embeddings translate complex data like text, images, or other forms of raw data into vectors of real numbers in a high-dimensional space. These vectors capture the semantics or meaning of the data in a form that can be used for various machine learning tasks.

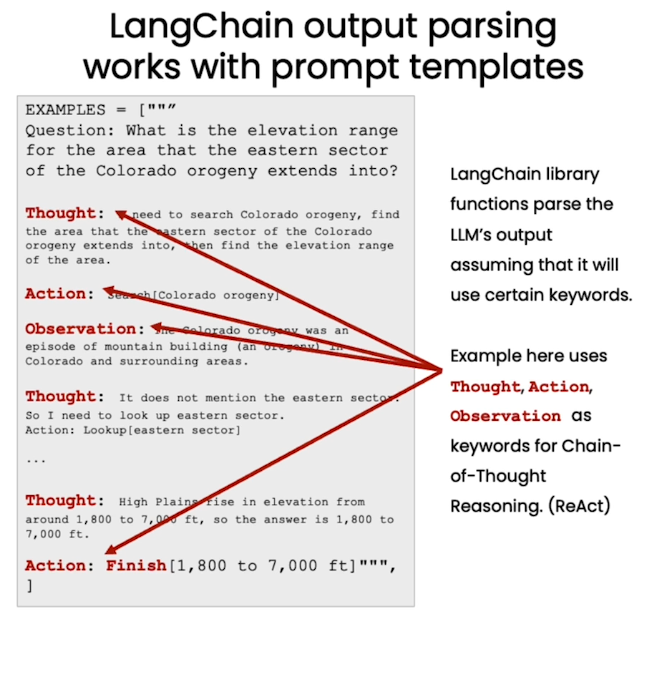
**Step-by-Step Process:**

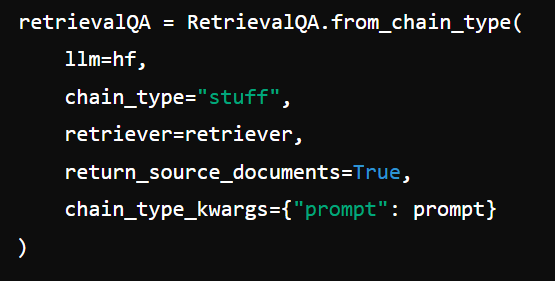
* **Text to Embedding Conversion:** Initially, the text data (documents, sentences, etc.) is converted into numerical embeddings using a pre-trained model (e.g., BERT, sentence-transformers). These embeddings capture the semantic meaning of the text. Each text piece is represented by a vector in a high-dimensional space.
* **Storing Embeddings:** The generated embeddings are stored in a vector store or index. Alongside the embeddings, metadata is usually stored, which includes the original text or references to where the original text can be found.
* **Similarity Search:** When a query is made, it is also converted into an embedding using the same model. The query embedding is then compared with the stored embeddings to find the most similar ones. This is often done using methods like cosine similarity or Euclidean distance.
* **Retrieving Original Text:** The vector store returns the identifiers of the most similar embeddings. Using these identifiers, the system retrieves the original text associated with these embeddings from the metadata.

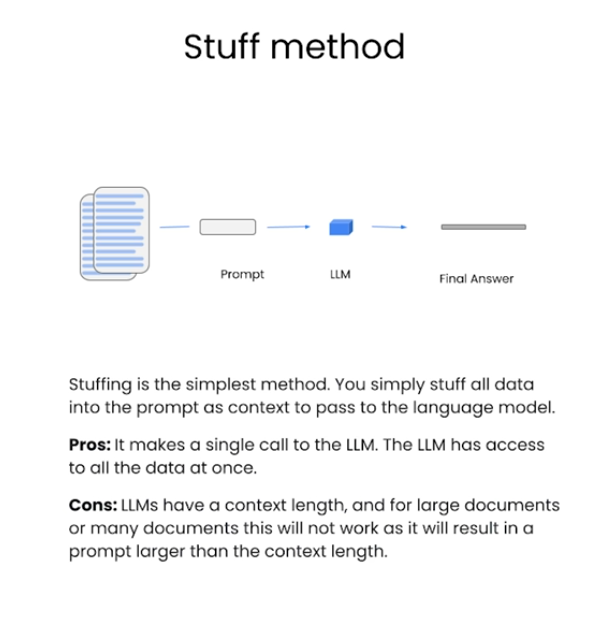


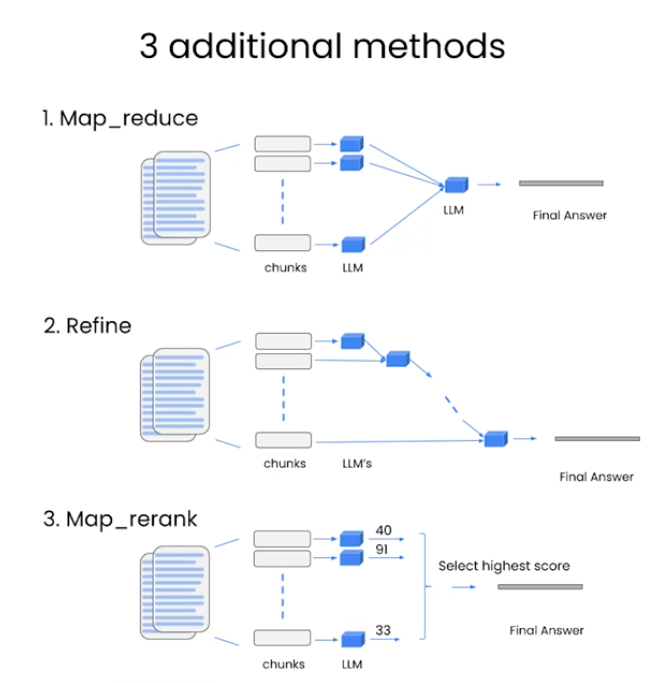
1. **Prompts:**
   * PromptTemplate helps in defining structured prompts for the LLMs to follow, improving the quality and relevance of generated responses.

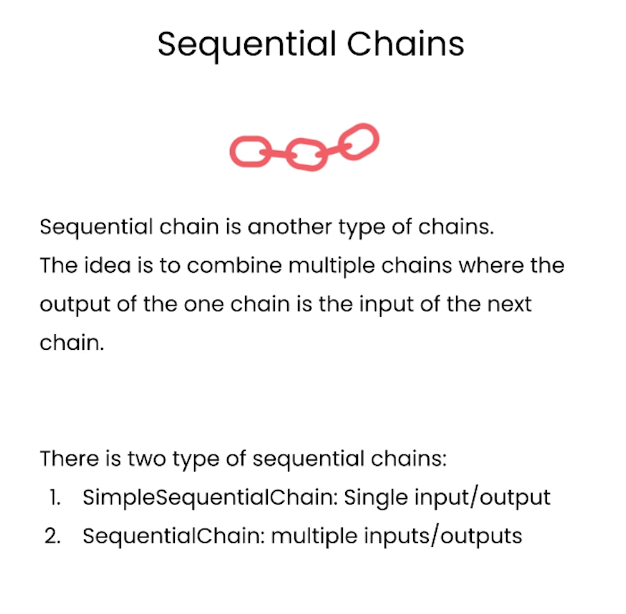




1. **Chains:**
   * RetrievalQA and other chains combine multiple components (e.g., retrievers, LLMs) to perform complex tasks like question answering, text generation, summarization, etc.
   * 
   * This line of code creates the chain, combining the language model (hf), the retriever (retriever), and the prompt template (prompt). The chain\_type="stuff" specifies the type of chain being used, in this case, a retrieval-based QA chain.







1. **LLMs (Language Models):**
   * HuggingFaceHub and other interfaces integrate various language models, allowing you to leverage their capabilities within the LangChain framework.

