DL Short Course Notes

Anri Lombard

2023

Table of Contents

LangChain for LLM Application Development 1
Introduction 2
Models, Prompts, and Parsers 2
Memory 2
Memory Types 2
Chains 3
Question and Answer 3

LangChain for LLM Application Development

Introduction

The creator of LangChain is Harrison Chase, a former Harvard student.

LangChain is:

1. an open-source framework for building LLM applications with either Python or JavaScript
2. focussed on composition and modularity

It makes the process of making modular components for specific use cases easier.

Models, Prompts, and Parsers

Models refer to the LMM models that are used to generate text.

Prompts refer to the inputs that are passed into the model.

- Prompt templates allow for the reuse of good prompts to do specific tasks. Langchain has a set of template prompts that can be used.

Parsers refer to the output from the model that needs to be parsed into a format that can be used downstream.

- Prompt templates also support output parsing.
- Output is initially given as a long string, but can be parsed into a dictionary with LangChain.

Memory

Store past conversations in memory so that it has a more conversational flow.

- The way it stores memory is with ConversationBufferMemory.
- The LLM itself is stateless (it does not remember anything from the past). Wrapper code gives the full conversation so far as context to the LLM so it could generate a response. As a conversation gets longer the context gets longer and becomes expensive to provide. LangChain solves this with ConversationBufferMemory, which specifies in a parameter how many conversation steps should be remembered.

Memory Types

1. **ConversationBufferMemory**: allows for storing of messages and then extracts the message in a variable.

2. **ConversationBufferWindowMemory**: keeps list of interactions of the conversation over time and only uses last k interactions.
3. **ConversationTokenBufferMemory**: keeps buffer of recent interactions in memory and uses token length rather than number of interactions to determine when to flush interactions.
4. **ConversationSummaryMemory**: creates summary of the conversation over time.

Chains

Combines LLM with prompt.

- chain.run formats the prompt and feeds it to the LLM, then returns the result of the chain.
- The simple sequential chain is where there are multiple chains with a single input and single output each.
- For multiple inputs and/or outputs
- A router chain routes input to the correct chain

Question and Answer

LLMs can only inspect a few thousand words at a time. This is why we need to use embeddings and vector stores.

A **Vector store** is a database of vectors that can be queried for similar vectors.

An **Embedding** is a vector representation of a piece of text which allows us to compare it to other pieces of text.

A **Vector database** is a database of vectors that can be queried for similar vectors.

The **Stuff Method / Stuffing** is a method to simply stuff all data into a prompt as context to pass to the language model.