Overview

Open-source development framework for LLM applications

Python and JavaScript (TypeScript) packages

Focused on composition and modularity

Key value adds:

1. Modular components
2. Use cases: Common ways to combine components

Components

- Models
  - LLMs: 20+ integrations
  - Chat Models
  - Text Embedding Models: 10+ integrations
- Prompts
  - Prompt Templates
  - Output Parsers: 5+ implementations
    - Retry/fixing logic
  - Example Selectors: 5+ implementations
- Indexes
  - Document Loaders: 50+ implementations
  - Text Splitters: 10+ implementations
  - Vector stores: 10+ integrations
  - Retrievers: 5+ integrations/implementations
- Chains
  - Prompt + LLM + Output parsing
  - Can be used as building blocks for longer chains
  - More application specific chains: 20+ types
- Agents
  - Agent Types: 5+ types
    - Algorithms for getting LLMs to use tools
  - Agent Toolkits: 10+ implementations
    - Agents armed with specific tools for a specific application

Why use prompt templates?

```
prompt = """
Your task is to determine if
the student's solution is
correct or not.
```

```
To solve the problem do the following:
- First, work out your own solution to the problem.
- Then compare your solution to the student's solution
and evaluate if the student's solution is correct or not.
...
Use the following format:
Question:
```
question here
```
Student's solution:
```
student's solution here
```
Actual solution:
```
...
steps to work out the solution and your solution here
```
Is the student's solution the same as actual solution \
just calculated:
```
yes or no
```
Student grade:
```
correct or incorrect
```

Question:
```
{question}
```
Student's solution:
```
{student_solution}
```
Actual solution:
"""
```

Prompts can be long and detailed.

Reuse good prompts when you can!

LangChain also provides prompts for common operations.

LangChain output parsing works with prompt templates

```
EXAMPLES = ["""
Question: What is the elevation range
for the area that the eastern sector
of the Colorado orogeny extends into?

Thought: I need to search Colorado orogeny, find
the area that the eastern sector of the Colorado
orogeny extends into, then find the elevation range
of the area.

Action: Search[Colorado orogeny]

Observation: The Colorado orogeny was an
episode of mountain building (an orogeny) in
Colorado and surrounding areas.

Thought: It does not mention the eastern sector.
So I need to look up eastern sector.
Action: Lookup[eastern sector]

...

Thought: High Plains rise in elevation from
around 1,800 to 7,000 ft, so the answer is 1,800 to
7,000 ft.

Action: Finish[1,800 to 7,000 ft]""",
]
```

LangChain library functions parse the LLM's output assuming that it will use certain keywords.

Example here uses **Thought**, **Action**, **Observation** as keywords for Chain-of-Thought Reasoning. (ReAct)

LangChain for LLM Application Development

Memory

LangChain

DeepLearning.AI

Memory

Large Language Models are ‘stateless’

- Each transaction is independent

Chatbots appear to have memory by providing the full conversation as ‘context’

LangChain provides several kinds of ‘memory’ to store and accumulate the conversation

Memory Types

ConversationBufferMemory

- This memory allows for storing of messages and then extracts the messages in a variable.

ConversationBufferWindowMemory

- This memory keeps a list of the interactions of the conversation over time. It only uses the last K interactions.

ConversationTokenBufferMemory

- This memory keeps a buffer of recent interactions in memory, and uses token length rather than number of interactions to determine when to flush interactions.

ConversationSummaryMemory

- This memory creates a summary of the conversation over time.

Additional Memory Types

Vector data memory

- Stores text (from conversation or elsewhere) in a vector database and retrieves the most relevant blocks of text.

Entity memories

- Using an LLM, it remembers details about specific entities.

You can also use multiple memories at one time.
E.g., Conversation memory + Entity memory to recall individuals.

You can also store the conversation in a conventional database (such as key-value store or SQL)

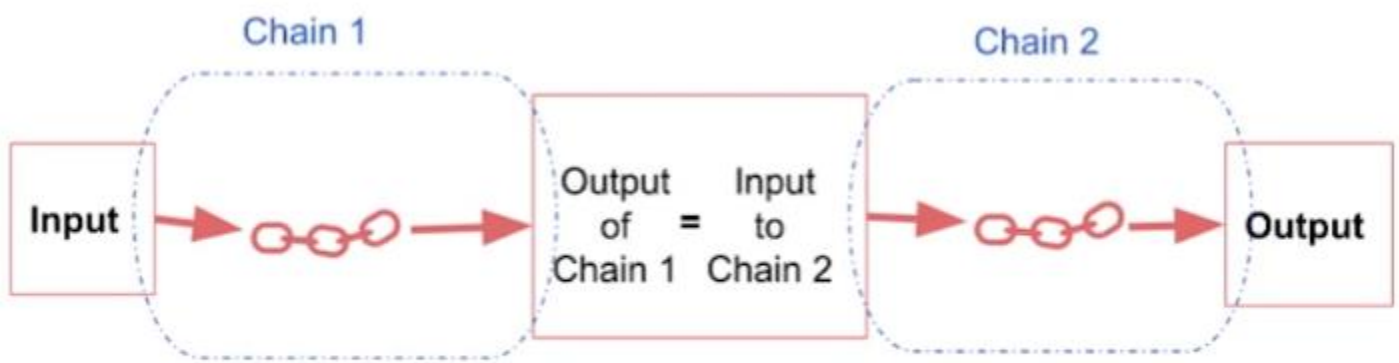
LangChain for LLM Application Development

Chains

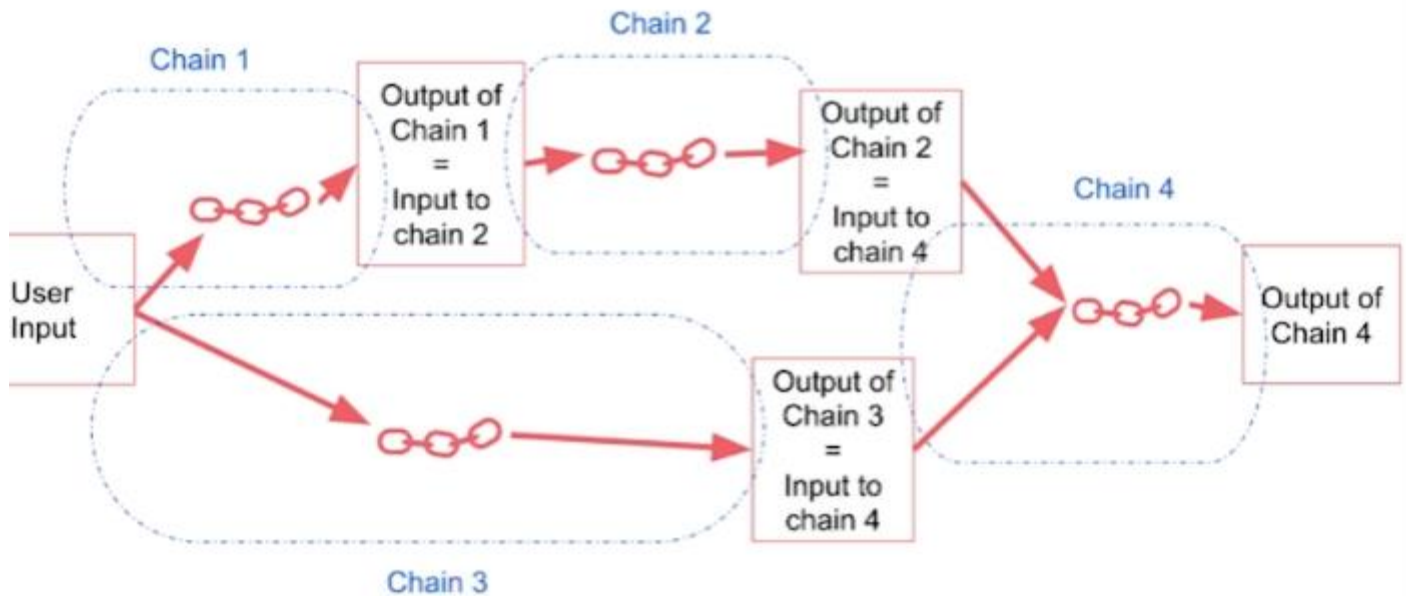
LangChain

DeepLearning.AI

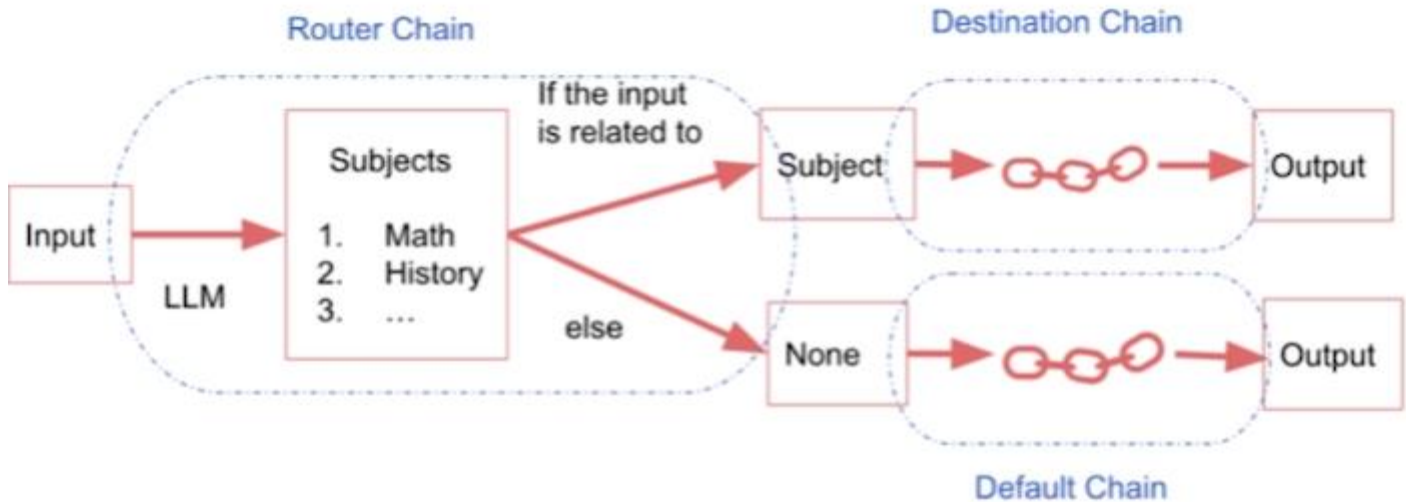
Simple Sequential Chain



Sequential Chain



Router Chain



LangChain for LLM Application Development

Question and Answer Over Documents

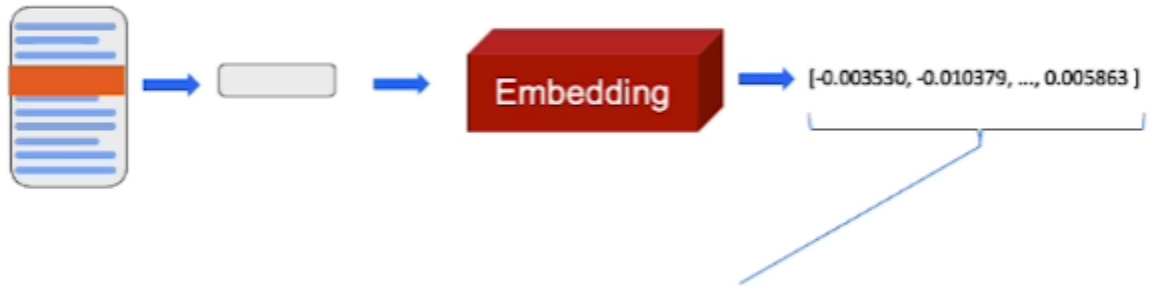
LangChain

DeepLearning.AI

LLM's on Documents

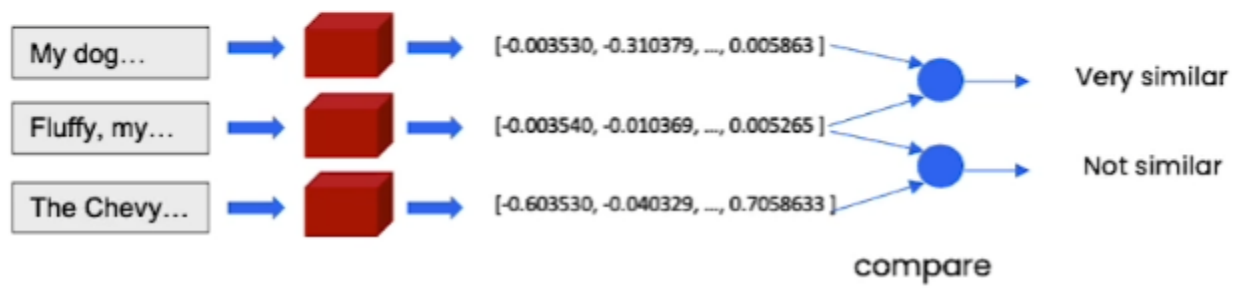
LLM's can only inspect a few thousand words at a time.

Embeddings

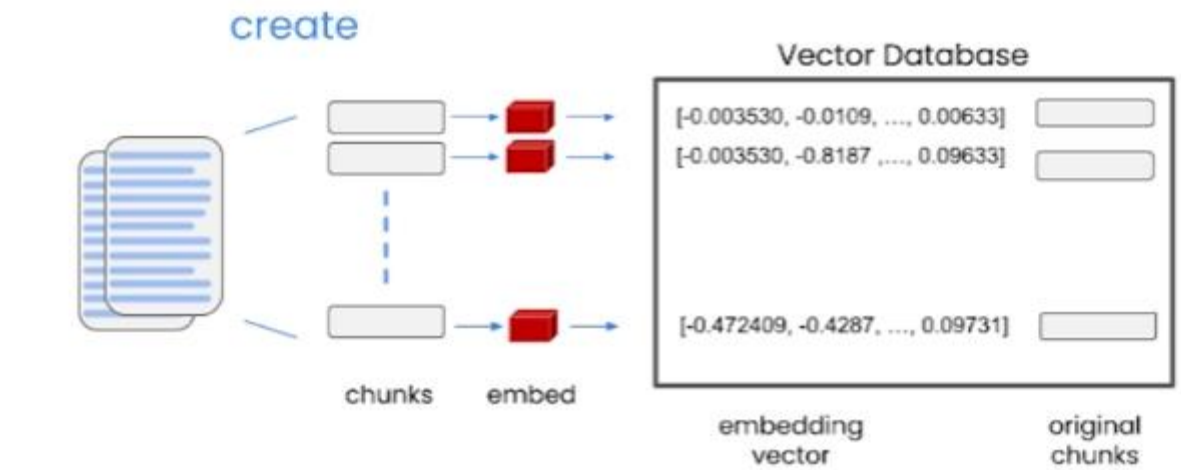


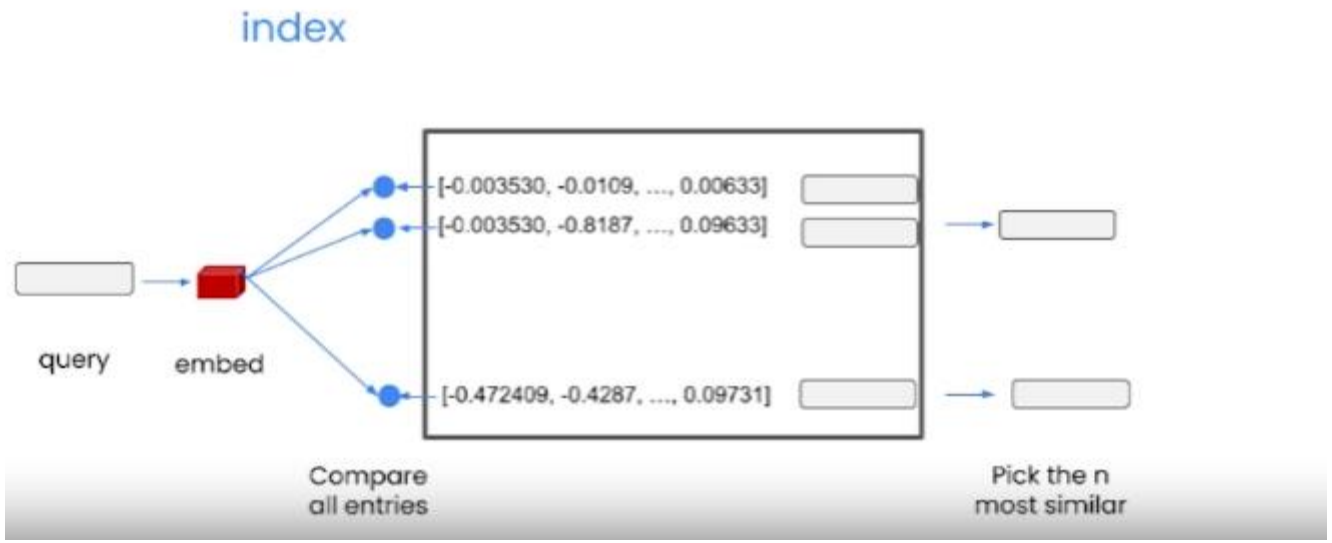
- Embedding vector captures content/meaning
- Text with similar content will have similar vectors

1) My dog Rover likes to chase squirrels.
2) Fluffy, my cat, refuses to eat from a can.
3) The Chevy Bolt accelerates to 60 mph in 6.7 seconds.



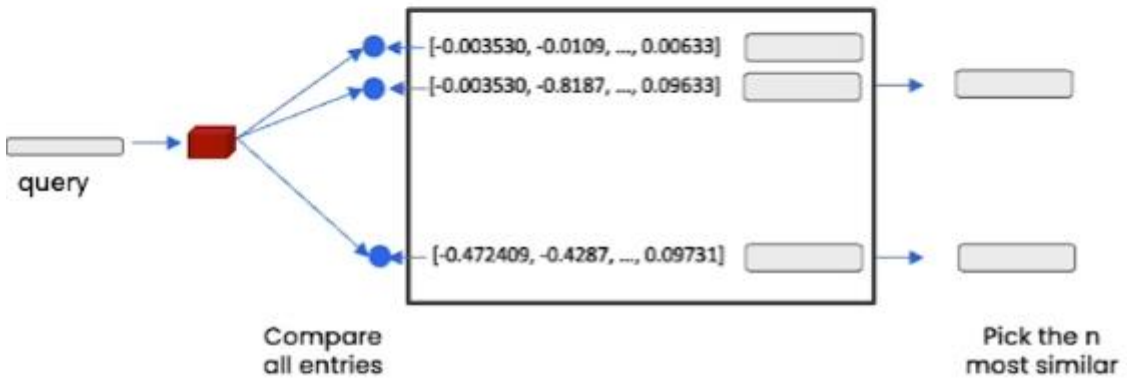
Vector Database





Vector Database

index



Process with llm

The returned values can now fit in the LLM context

Stuff method

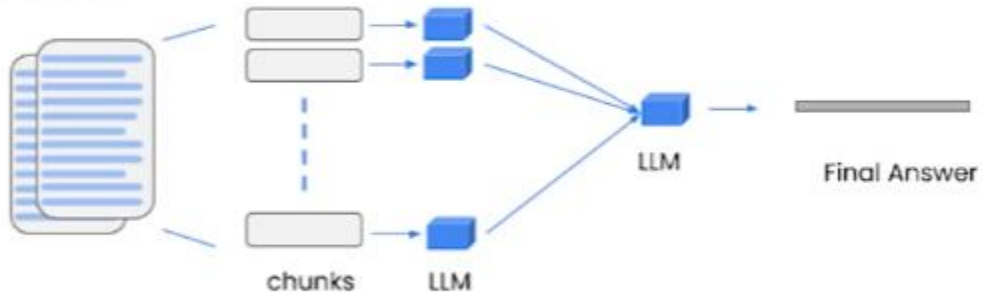
Stuffing is the simplest method. You simply stuff all data into the prompt as context to pass to the language model.

Pros: It makes a single call to the LLM. The LLM has access to all the data at once.

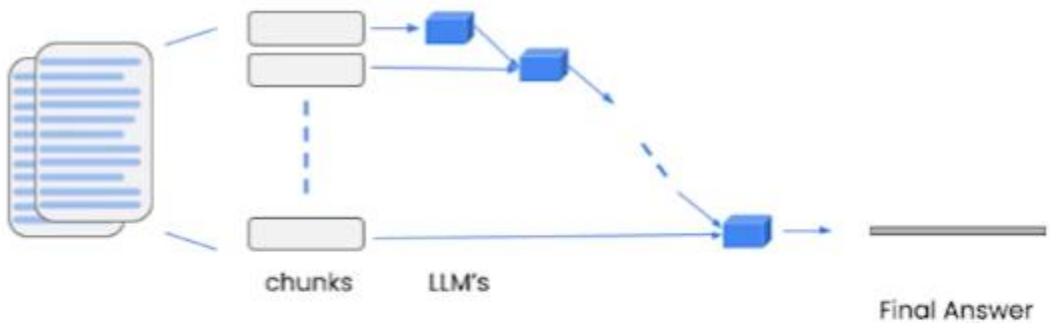
Cons: LLMs have a context length, and for large documents or many documents this will not work as it will result in a prompt larger than the context length.

3 additional methods

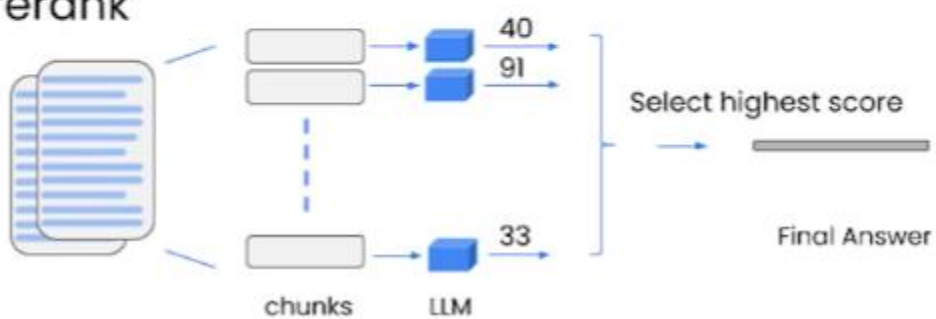
1. Map_reduce



2. Refine



3. Map_rerank



LangChain for LLM Application Development

Evaluating LLM Applications

LangChain

DeepLearning.AI

In [24]:
```
for i, eg in enumerate(examples):
    print(f"Example {i}:")
    print("Question: " + predictions[i]['query'])
    print("Real Answer: " + predictions[i]['answer'])
    print("Predicted Answer: " + predictions[i]['result'])
    print("Predicted Grade: " + graded_outputs[i]['text'])
    print()
```

```
Example 0:
Question: Do the Cozy Comfort Pullover Set have side pockets?
Real Answer: Yes
Predicted Answer: The Cozy Comfort Pullover Set, Stripe does have sid
e pockets.
Predicted Grade: CORRECT

Example 1:
Question: What collection is the Ultra-Lofty 850 Stretch Down Hooded
Jacket from?
Real Answer: The DownTek collection
Predicted Answer: The Ultra-Lofty 850 Stretch Down Hooded Jacket is f
rom the DownTek collection.
Predicted Grade: CORRECT

Example 2:
Question: What is the weight of each pair of Women's Campside Oxford
s?
Real Answer: The approximate weight of each pair of Women's Campside
Oxfords is 1 lb. 1 oz.
Predicted Answer: The weight of each pair of Women's Campside Oxfords
is approximately 1 lb. 1 oz.
Predicted Grade: CORRECT

Example 3:
Question: What are the dimensions of the small and medium Recycled Wa
terhog Dog Mat?
Real Answer: The dimensions of the small Recycled Waterhog Dog Mat ar
e 18" x 28" and the dimensions of the medium Recycled Waterhog Dog Ma
t are 22.5" x 34.5"
```

LangChain for LLM Application Development

Agents

```
    outside this function."""
    return str(date.today())
```

In [11]:
```
agent= initialize_agent(
    tools + [time],
    llm,
    agent=AgentType.CHAT_ZERO_SHOT_REACT_DESCRIPTION,
    handle_parsing_errors=True,
    verbose = True)
```

In [12]:
```
agent.run("whats the date today?")
```

```
> Entering new AgentExecutor chain...
Question: whats the date today?
Thought: I can use the time function to get the date
Action:
```
{
  "action": "time",
  "action_input": ""
}
```

Observation: 2023-05-21
Thought:I need to return the date as the final answer.
Final Answer: Today's date is 2023-05-21.

> Finished chain.
```

Out[12]: "Today's date is 2023-05-21."