LangChain simplifies building applications with language models through reusable components and pre-built chains. It makes models data-aware and agentic for more dynamic interactions. The modular architecture supports rapid development and customization.

### LLMs

An interface for OpenAI GPT-3.5-turbo LLM

```python
from langchain.llms import OpenAI
llm = OpenAI(temperature=0.9)
text = "What do you know about KDnuggets?"
llm(text)
```

>>> KDnuggets is one of the most popular data science websites which focusses....

### Agents and Tools

Tool refers to a function that performs a specific task, such as Google Search, database lookup, or Python REPL. Agents use LLMs to choose a sequence of actions to execute.

```python
from langchain.agents import load_tools
from langchain.agents import initialize_agent
tools = load_tools(
    ['wikipedia', 'llm-math', 'serp', 'duckduckgo'], llm=llm)
agent = initialize_agent(tools, llm, agent="zero-shot-react-description")
result = agent.run("Can you tell me the distance between Earth and the moon? And could you please convert it into miles? Thank you")
```

>>> Action Input: Earth-moon distance
>>> Action Input: Calculator
>>> Action Input: Conversion from meters to miles
>>> Final Answer: The distance between Earth and the Moon is approximately 384,400 miles.

### Chains

Combining LLMs and prompt template can enhance multi-step workflows.

```python
from langchain.chains import ConversationChain
from langchain.memory import ConversationBufferMemory
conversation = ConversationChain(
    llm=llm, memory=ConversationBufferMemory(),
    predict_action_input="Do you have any questions?"
)
result = conversation.predict(input="How can one overcome anxiety?")
```

>>> To be mindful of the present, it can be helpful to pra....

### Document Loaders

By combining language models with your own text data, you can answer personalized queries. You can load CSV, Markdown, PDF, and more.

```python
from langchain.document_loaders import TextLoader
raw_document = TextLoader("/work/data/Gregory.txt").load()
```

### Memory

LangChain facilitates prompt management and optimization through the use of prompt templates.

```python
from langchain import PromptTemplate
prompt = PromptTemplate(  
    template="What is your favorite place to travel?
    Could you provide some information on the impact of global warming?"
)
result = prompt.format(question="What is your favorite place to travel?
Could you provide some information on the impact of global warming?")
```

>>> Global warming is no laughing matter but that doesn....

### An Interface for HugginFace LLM

```python
from langchain import HuggingFaceHub
llm = HuggingFaceHub(repo_id="togethercomputer/LLaMA-2-7B-32K", model_kwargs={"temperature":0.9, "max_length":64})
text = "How old is KDnuggets?"
llm(text)
```

>>> KDnuggets was founded in 1997, making it 23 years old.

### Document Loaders

A retriever is an interface that returns documents based on an unstructured query. When combined with LLM, it generates a natural response instead of simply displaying the text from the document.

```python
from langchain.chains import RetrievalQA
from langchain.text_splitter import CharacterTextSplitter
from langchain.embeddings.openai import OpenAIEmbeddings
from langchain.vectorstores import FAISS

# Text Splitter
text_splitter = CharacterTextSplitter(chunk_size=1000, chunk_overlap=0)
docs = text_splitter.split_documents(raw_document)

# Vector Store
db = FAISS.from_documents(docs, OpenAIEmbeddings())

# Similarity Search
query = "When was Gregory born?"
docs = db.similarity_search(query)
print(docs[0].page_content)
```

>>> Gregory I. Piatetsky-Shapiro was born on July 7, 1958.

### Memory

LangChain simplifies persistent state management in chain or agent calls with a standard interface.

```python
from langchain.chains import ConversationChain
conversation = ConversationChain(llm=llm, verbose=True,
memory=ConversationBufferMemory())
result = conversation.predict(input="How can one overcome anxiety?")
```

>>> To be mindful of the present, it can be helpful to pra....

### Agents and Tools

Tool refers to a function that performs a specific task, such as Google Search, database lookup, or Python REPL. Agents use LLMs to choose a sequence of actions to execute.

```python
from langchain.agents import load_tools
from langchain.agents import initialize_agent
tools = load_tools(['wikipedia', 'llm-math', 'serp', 'duckduckgo'], llm=llm)
agent = initialize_agent(tools, llm, agent="zero-shot-react-description")
result = agent.run("Can you tell me the distance between Earth and the moon? And could you please convert it into miles? Thank you")
```

>>> Action Input: Earth-moon distance
>>> Action Input: Calculator
>>> Action Input: Conversion from meters to miles
>>> Final Answer: The distance between Earth and the Moon is approximately 384,400 miles.

### Document Loaders

By combining language models with your own text data, you can answer personalized queries. You can load CSV, Markdown, PDF, and more.

```python
from langchain.document_loaders import TextLoader
raw_document = TextLoader("/work/data/Gregory.txt").load()
```

### Document Loaders

A retriever is an interface that returns documents based on an unstructured query. When combined with LLM, it generates a natural response instead of simply displaying the text from the document.

```python
from langchain.chains import RetrievalQA
from langchain.text_splitter import CharacterTextSplitter
from langchain.embeddings.openai import OpenAIEmbeddings
from langchain.vectorstores import FAISS

# Text Splitter
text_splitter = CharacterTextSplitter(chunk_size=1000, chunk_overlap=0)
docs = text_splitter.split_documents(raw_document)

# Vector Store
db = FAISS.from_documents(docs, OpenAIEmbeddings())

# Similarity Search
query = "When was Gregory born?"
docs = db.similarity_search(query)
print(docs[0].page_content)
```

>>> Gregory I. Piatetsky-Shapiro was born on July 7, 1958.

### Document Loaders

By combining language models with your own text data, you can answer personalized queries. You can load CSV, Markdown, PDF, and more.

```python
from langchain.document_loaders import TextLoader
raw_document = TextLoader("/work/data/Gregory.txt").load()
```

### Memory

LangChain simplifies persistent state management in chain or agent calls with a standard interface.

```python
from langchain.chains import ConversationChain
conversation = ConversationChain(llm=llm, verbose=True,
memory=ConversationBufferMemory())
result = conversation.predict(input="How can one overcome anxiety?")
```

>>> To be mindful of the present, it can be helpful to pra....

### Agents and Tools

Tool refers to a function that performs a specific task, such as Google Search, database lookup, or Python REPL. Agents use LLMs to choose a sequence of actions to execute.

```python
from langchain.agents import load_tools
from langchain.agents import initialize_agent
tools = load_tools(['wikipedia', 'llm-math', 'serp', 'duckduckgo'], llm=llm)
agent = initialize_agent(tools, llm, agent="zero-shot-react-description")
result = agent.run("Can you tell me the distance between Earth and the moon? And could you please convert it into miles? Thank you")
```

>>> Action Input: Earth-moon distance
>>> Action Input: Calculator
>>> Action Input: Conversion from meters to miles
>>> Final Answer: The distance between Earth and the Moon is approximately 384,400 miles.

### Document Loaders

By combining language models with your own text data, you can answer personalized queries. You can load CSV, Markdown, PDF, and more.

```python
from langchain.document_loaders import TextLoader
raw_document = TextLoader("/work/data/Gregory.txt").load()
```

### Memory

LangChain simplifies persistent state management in chain or agent calls with a standard interface.

```python
from langchain.chains import ConversationChain
conversation = ConversationChain(llm=llm, verbose=True,
memory=ConversationBufferMemory())
result = conversation.predict(input="How can one overcome anxiety?")
```

>>> To be mindful of the present, it can be helpful to pra....

### Agents and Tools

Tool refers to a function that performs a specific task, such as Google Search, database lookup, or Python REPL. Agents use LLMs to choose a sequence of actions to execute.

```python
from langchain.agents import load_tools
from langchain.agents import initialize_agent
tools = load_tools(['wikipedia', 'llm-math', 'serp', 'duckduckgo'], llm=llm)
agent = initialize_agent(tools, llm, agent="zero-shot-react-description")
result = agent.run("Can you tell me the distance between Earth and the moon? And could you please convert it into miles? Thank you")
```

>>> Action Input: Earth-moon distance
>>> Action Input: Calculator
>>> Action Input: Conversion from meters to miles
>>> Final Answer: The distance between Earth and the Moon is approximately 384,400 miles.

### Document Loaders

By combining language models with your own text data, you can answer personalized queries. You can load CSV, Markdown, PDF, and more.

```python
from langchain.document_loaders import TextLoader
raw_document = TextLoader("/work/data/Gregory.txt").load()
```

### Memory

LangChain simplifies persistent state management in chain or agent calls with a standard interface.

```python
from langchain.chains import ConversationChain
conversation = ConversationChain(llm=llm, verbose=True,
memory=ConversationBufferMemory())
result = conversation.predict(input="How can one overcome anxiety?")
```

>>> To be mindful of the present, it can be helpful to pra....

### Agents and Tools

Tool refers to a function that performs a specific task, such as Google Search, database lookup, or Python REPL. Agents use LLMs to choose a sequence of actions to execute.

```python
from langchain.agents import load_tools
from langchain.agents import initialize_agent
tools = load_tools(['wikipedia', 'llm-math', 'serp', 'duckduckgo'], llm=llm)
agent = initialize_agent(tools, llm, agent="zero-shot-react-description")
result = agent.run("Can you tell me the distance between Earth and the moon? And could you please convert it into miles? Thank you")
```

>>> Action Input: Earth-moon distance
>>> Action Input: Calculator
>>> Action Input: Conversion from meters to miles
>>> Final Answer: The distance between Earth and the Moon is approximately 384,400 miles.

### Document Loaders

By combining language models with your own text data, you can answer personalized queries. You can load CSV, Markdown, PDF, and more.

```python
from langchain.document_loaders import TextLoader
raw_document = TextLoader("/work/data/Gregory.txt").load()
```