Lesson 6: Essay Writer

In [1]:
```
from dotenv import load_dotenv

_ = load_dotenv()
```

In [2]:
```
from langgraph.graph import StateGraph, END
from typing import TypedDict, Annotated, List
import operator
from langgraph.checkpoint.sqlite import SqliteSaver
from langchain_core.messages import AnyMessage, SystemMessage, HumanMessage, AIMessage, ChatM

memory = SqliteSaver.from_conn_string(":memory:")
```

In [3]:
```
class AgentState(TypedDict):
    task: str
    plan: str
    draft: str
    critique: str
    content: List[str]
    revision_number: int
    max_revisions: int
```

In [4]:
```
from langchain_openai import ChatOpenAI
model = ChatOpenAI(model="gpt-3.5-turbo", temperature=0)
```

In [5]:
```
PLAN_PROMPT = """You are an expert writer tasked with writing a high level outline of an essay
Write such an outline for the user provided topic. Give an outline of the essay along with any
or instructions for the sections."""
```

In [6]:
```
WRITER_PROMPT = """You are an essay assistant tasked with writing excellent 5-paragraph essays
Generate the best essay possible for the user's request and the initial outline. \
If the user provides critique, respond with a revised version of your previous attempts. \
Utilize all the information below as needed:

------

{content}"""
```

In [7]:
```
REFLECTION_PROMPT = """You are a teacher grading an essay submission. \
Generate critique and recommendations for the user's submission. \
Provide detailed recommendations, including requests for length, depth, style, etc."""
```

In [8]:
```
RESEARCH_PLAN_PROMPT = """You are a researcher charged with providing information that can \
be used when writing the following essay. Generate a list of search queries that will gather
any relevant information. Only generate 3 queries max."""
```

In [9]:
```
RESEARCH_CRITIQUE_PROMPT = """You are a researcher charged with providing information that ca
be used when making any requested revisions (as outlined below). \
Generate a list of search queries that will gather any relevant information. Only generate 3 
```

In [10]:
```
from langchain_core.pydantic_v1 import BaseModel

class Queries(BaseModel):
    queries: List[str]
```

In [11]:
```python
from tavily import TavilyClient
import os
tavily = TavilyClient(api_key=os.environ["TAVILY_API_KEY"])
```

In [12]:
```python
def plan_node(state: AgentState):
    messages = [
        SystemMessage(content=PLAN_PROMPT),
        HumanMessage(content=state['task'])
    ]
    response = model.invoke(messages)
    return {"plan": response.content}
```

In [13]:
```python
def research_plan_node(state: AgentState):
    queries = model.with_structured_output(Queries).invoke([
        SystemMessage(content=RESEARCH_PLAN_PROMPT),
        HumanMessage(content=state['task'])
    ])
    content = state['content'] or []
    for q in queries.queries:
        response = tavily.search(query=q, max_results=2)
        for r in response['results']:
            content.append(r['content'])
    return {"content": content}
```

In [14]:
```python
def generation_node(state: AgentState):
    content = "\n\n".join(state['content'] or [])
    user_message = HumanMessage(
        content=f"{state['task']}\n\nHere is my plan:\n\n{state['plan']}")
    messages = [
        SystemMessage(
            content=WRITER_PROMPT.format(content=content)
        ),
        user_message
        ]
    response = model.invoke(messages)
    return {
        "draft": response.content,
        "revision_number": state.get("revision_number", 1) + 1
    }
```

In [15]:
```python
def reflection_node(state: AgentState):
    messages = [
        SystemMessage(content=REFLECTION_PROMPT),
        HumanMessage(content=state['draft'])
    ]
    response = model.invoke(messages)
    return {"critique": response.content}
```

In [16]:
```python
def research_critique_node(state: AgentState):
    queries = model.with_structured_output(Queries).invoke([
        SystemMessage(content=RESEARCH_CRITIQUE_PROMPT),
        HumanMessage(content=state['critique'])
    ])
    content = state['content'] or []
    for q in queries.queries:
        response = tavily.search(query=q, max_results=2)
        for r in response['results']:
            content.append(r['content'])
    return {"content": content}
```

In [17]:
```python
def should_continue(state):
    if state["revision_number"] > state["max_revisions"]:
```

```
        return END
    return "reflect"
```

In [18]:
```
builder = StateGraph(AgentState)
```

In [19]:
```
builder.add_node("planner", plan_node)
builder.add_node("generate", generation_node)
builder.add_node("reflect", reflection_node)
builder.add_node("research_plan", research_plan_node)
builder.add_node("research_critique", research_critique_node)
```

In [20]:
```
builder.set_entry_point("planner")
```

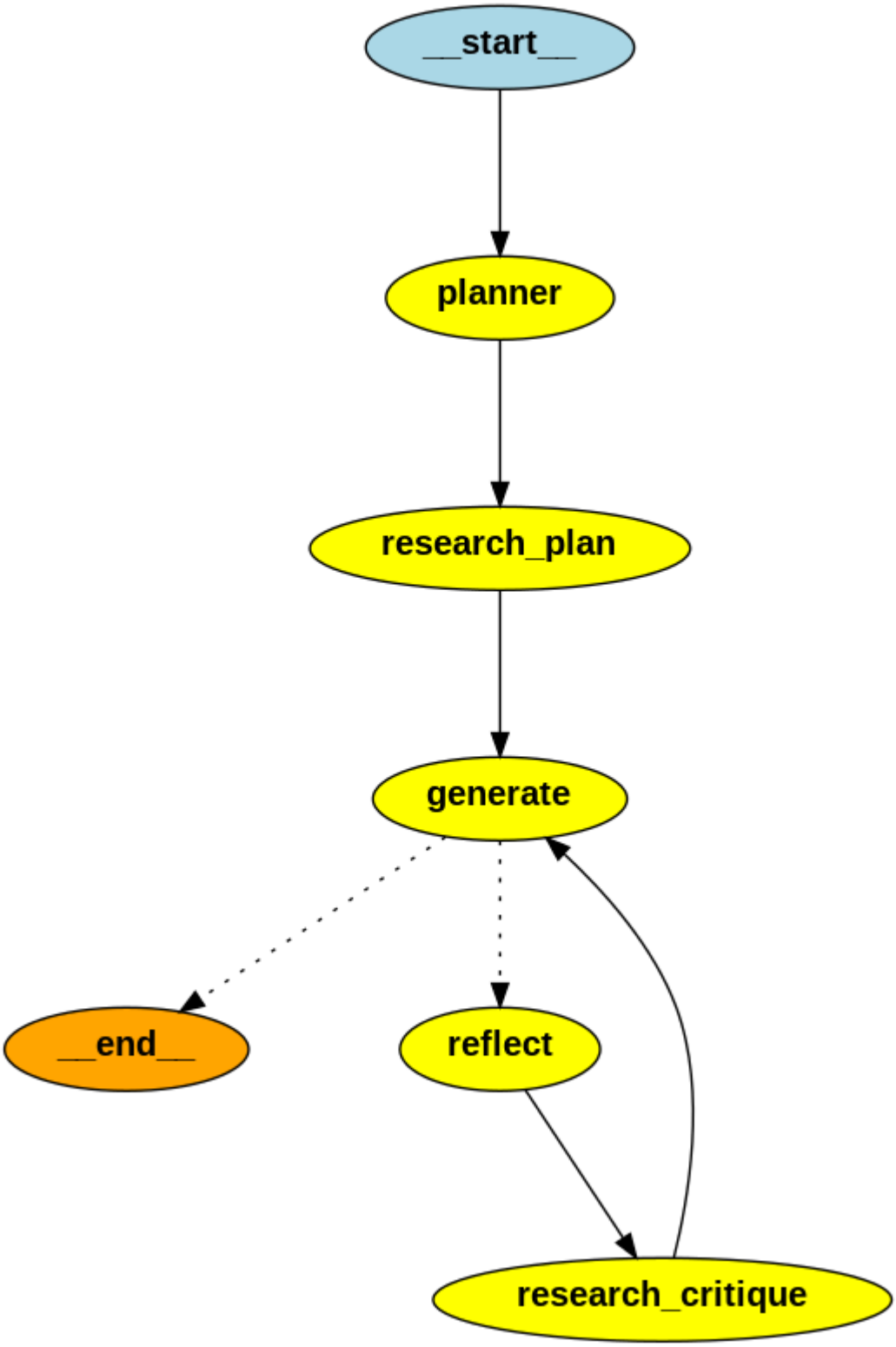
In [21]:
```
builder.add_conditional_edges(
    "generate",
    should_continue,
    {END: END, "reflect": "reflect"}
)
```

In [22]:
```
builder.add_edge("planner", "research_plan")
builder.add_edge("research_plan", "generate")

builder.add_edge("reflect", "research_critique")
builder.add_edge("research_critique", "generate")
```

In [23]:
```
graph = builder.compile(checkpointer=memory)
```

In [24]:
```
from IPython.display import Image

Image(graph.get_graph().draw_png())
```

Out[24]:



In [25]:

```
thread = {"configurable": {"thread_id": "1"}}
for s in graph.stream({
    'task': "what is the difference between langchain and langsmith",
    "max_revisions": 2,
    "revision_number": 1,
}, thread):
    print(s)
```

{'planner': {'plan': 'I. Introduction\n A. Brief overview of Langchain and Langsmith\n B. Thesis statement: Exploring the differences between Langchain and Langsmith\n\nII. Langchain\n A. Definition and purpose\n B. Key features and characteristics\n C. Use cases and applications\n D. Advantages and limitations\n\nIII. Langsmith\n A. Definition and purpose\n B. Key features and characteristics\n C. Use cases and applications\n D. Advantages and limitations\n\nIV. Comparison between Langchain and Langsmith\n A. Technology stack\n B. Scalability and performance\n C. Security and privacy\n D. Adoption and popularity\n E. Future prospects\n\nV. Conclusion\n A. Recap of key differences between Langchain and Langsmith\n B. Implications for the future of blockchain technology\n C. Final thoughts and recommendations\n\nNotes:\n- Ensure to provide clear definitions and examples for both Langchain and Langsmith.\n- Compare and contrast the two technologies in terms of features, use cases, advantages, and limitations.\n- Use relevant data and statistics to support the analysis.\n- Conclude with insights on the potential impact of Langchain and Langsmith on the blockchain industry.'}}
{'research_plan': {'content': ['This report aims to provide a detailed analysis of the difference between LangChain and LangSmith based on the information available from various sources. LangChain: A Language Model Software Tool for Prototyping. LangChain is a language model software tool that is primarily focused on helping developers build prototypes. It provides a platform ...', "In this blog, we'll delve into the differences between LangChain and LangSmith, their pros and cons, and when to use each one. LangChain. LangChain is an open-source Python package that provides a framework for building and deploying LLM applications. It allows developers to create prototypes quickly and easily, making it an ideal choice for ...", 'LangChain and LangSmith are two complementary tools that cater to different stages and requirements of LLM development. LangChain is ideal for early-stage prototyping and small-scale applications, while LangSmith is better suited for large-scale, production-ready applications that require advanced debugging, testing, and monitoring capabilities', 'Langchain vs Langsmith: Unpacking the AI Language Model Showdown\nOverview of Langchain and Langsmith\nLangchain is a versatile open-source framework that enables you to build applications utilizing large language models (LLM) like GPT-3. Check out our free WhatsApp channel to stay educated on LLM developments:\nJoin the Finxter Academy and unlock access to premium courses 👑 to certify your skills in exponential technologies and programming.\n Frequently Asked Questions\nWhether you’re trying to figure out which tool fits your needs or you’re just getting started with language model automation, these FAQs will help shed light on the common curiosities about Langchain and LangSmith.\n The best way to find out is to reach out to them through the LangSmith Walkthrough page or to inquire about access directly through their support channels.\n Here’s how you might start a simple Langchain project in Python:\nTo integrate LangSmith, you could write something like this:\nYou’re not limited to Python, though.', 'LangChain and LangSmith are two complementary tools that cater to different stages and requirements of LLM development. LangChain is ideal for early-stage prototyping and small-scale applications, while LangSmith is better suited for large-scale, production-ready applications that require advanced debugging, testing, and monitoring capabilities.', "Langsmith started charging. Time to compare alternatives. Hey r/Langchain! I've been using Langsmith for a while, and while it's been great, I'm curious about what else is out there. Specifically, I'm on the hunt for something fresh in the realm of LLM observability tools. ... I just think it was the obvious go-to for langchain developers when ..."]}}
{'generate': {'draft': '**Essay:**\n\n**I. Introduction**\n\nLangChain and LangSmith are two prominent tools in the realm of language model software development. While both serve essential functions in this domain, they possess distinct characteristics that cater to different needs. This essay aims to delve into the disparities between LangChain and LangSmith to provide a comprehensive understanding of their unique offerings.\n\n**II. LangChain**\n\nLangChain is an open-source Python package designed to facilitate the creation and deployment of large language model (LLM) applications. It is particularly advantageous for rapid prototyping and the development of small-scale applications. Key features of LangChain include its user-friendly framework, ease of use, and quick prototyping capabilities. Developers often utilize LangChain for experimenting with LLM applications and creating initial prototypes. However, its limitations may arise when scaling up to larger, production-ready projects.\n\n**III. LangSmith**\n\nIn contrast, LangSmith is tailored for large-scale, production-ready applications that demand advanced debugging, testing, and monitoring functionalities. It provides a robust platform for developers to build sophisticated LLM applications with a focus on scalability and performance. While LangSmith may have a steeper learning curve compared to LangChain, it offers comprehensive tools for ensuring the reliability and efficiency of complex language models in real-world applications.\n\n**IV. Comparison between LangChain and LangSmith**\n\n- *Technology Stack*: LangChain primarily relies on Python and offers a straightforward approach to LLM development, whereas LangSmith may incorporate a more diverse technology stack to support its advanced features.\n- *Scalability and Performance*: LangChain excels in early-stage prototyping and small-scale applications, while LangSmith shines in handling large-scale projects that require optimal performance and scalability.\n- *Security and Privacy*: Both tools prioritize security; howeve

r, LangSmith may provide more robust security features to safeguard extensive LLM application
s.\n- *Adoption and Popularity*: LangChain is popular among developers for its ease of use and
quick prototyping capabilities, while LangSmith is favored for its comprehensive debugging and
monitoring tools.\n- *Future Prospects*: The future of LangChain and LangSmith lies in their a
bility to adapt to evolving LLM technologies and meet the increasing demands of developers for
efficient and reliable language model applications.\n\n**V. Conclusion**\n\nIn conclusion, the
disparities between LangChain and LangSmith underscore their complementary roles in the landsc
ape of LLM development. While LangChain caters to early-stage prototyping and experimentation,
LangSmith addresses the needs of developers working on large-scale, production-ready applicati
ons. Understanding the unique features, advantages, and limitations of each tool is crucial fo
r developers to make informed decisions based on their project requirements. As the blockchain
industry continues to evolve, the distinct contributions of LangChain and LangSmith are poised
to shape the future of language model software development significantly.', 'revision_number':
2}}
{'reflect': {'critique': '**Critique:**\n\nThe essay provides a clear overview of LangChain an
d LangSmith, highlighting their key features and differences effectively. The introduction set
s the stage well by outlining the purpose of the essay, and the subsequent sections delve into
the specifics of each tool concisely. The comparison between LangChain and LangSmith is struct
ured logically, covering essential aspects such as technology stack, scalability, security, ad
option, and future prospects.\n\n**Recommendations:**\n\n1. **Depth and Analysis:** While the
essay provides a good overview of LangChain and LangSmith, consider delving deeper into specif
ic examples or case studies where each tool has been successfully utilized. Providing real-wor
ld scenarios can help readers better understand the practical applications and benefits of the
se tools.\n\n2. **Expansion on Limitations:** It would be beneficial to elaborate further on t
he limitations of LangChain and LangSmith. Exploring scenarios where each tool may not be the
best fit or where developers might encounter challenges can provide a more comprehensive under
standing of their capabilities.\n\n3. **Case Studies or Use Cases:** Including case studies or
examples of projects that have utilized LangChain and LangSmith can add depth to the essay. An
alyzing how these tools have been applied in different contexts and the outcomes achieved can
enhance the discussion.\n\n4. **Future Trends:** While the essay briefly touches on the future
prospects of LangChain and LangSmith, consider expanding on this aspect. Discuss potential tre
nds in language model software development, emerging technologies that could impact these tool
s, and how they might evolve to meet future demands.\n\n5. **Conclusion:** Strengthen the conc
lusion by summarizing the key points discussed in the essay and reiterating the significance o
f understanding the unique features of LangChain and LangSmith. Consider emphasizing the impor
tance of choosing the right tool based on project requirements and the evolving landscape of l
anguage model software development.\n\n6. **Length and Detail:** The essay could benefit from
additional details and explanations to provide a more comprehensive analysis of LangChain and
LangSmith. Consider expanding on the features, functionalities, and user experiences of each t
ool to offer a more in-depth comparison.\n\nOverall, the essay presents a solid foundation for
comparing LangChain and LangSmith. By incorporating more examples, case studies, and in-depth
analysis, you can enhance the depth and clarity of the discussion, providing readers with a ri
cher understanding of these language model software tools.'}}

{'research_critique': {'content': ['This report aims to provide a detailed analysis of the difference between LangChain and LangSmith based on the information available from various sources. LangChain: A Language Model Software Tool for Prototyping. LangChain is a language model software tool that is primarily focused on helping developers build prototypes. It provides a platform ...', "In this blog, we'll delve into the differences between LangChain and LangSmith, their pros and cons, and when to use each one. LangChain. LangChain is an open-source Python package that provides a framework for building and deploying LLM applications. It allows developers to create prototypes quickly and easily, making it an ideal choice for ...", 'LangChain and LangSmith are two complementary tools that cater to different stages and requirements of LLM development. LangChain is ideal for early-stage prototyping and small-scale applications, while LangSmith is better suited for large-scale, production-ready applications that require advanced debugging, testing, and monitoring capabilities', 'Langchain vs Langsmith: Unpacking the AI Language Model Showdown\nOverview of Langchain and Langsmith\nLangchain is a versatile open-source framework that enables you to build applications utilizing large language models (LLM) like GPT-3. Check out our free WhatsApp channel to stay educated on LLM developments:\nJoin the Finxter Academy and unlock access to premium courses 👑 to certify your skills in exponential technologies and programming.\n Frequently Asked Questions\nWhether you’re trying to figure out which tool fits your needs or you’re just getting started with language model automation, these FAQs will help shed light on the common curiosities about Langchain and LangSmith.\n The best way to find out is to reach out to them through the LangSmith Walkthrough page or to inquire about access directly through their support channels.\n Here’s how you might start a simple Langchain project in Python:\nTo integrate LangSmith, you could write something like this:\n You’re not limited to Python, though.', 'LangChain and LangSmith are two complementary tools that cater to different stages and requirements of LLM development. LangChain is ideal for early-stage prototyping and small-scale applications, while LangSmith is better suited for large-scale, production-ready applications that require advanced debugging, testing, and monitoring capabilities.', "Langsmith started charging. Time to compare alternatives. Hey r/Langchain! I've been using Langsmith for a while, and while it's been great, I'm curious about what else is out there. Specifically, I'm on the hunt for something fresh in the realm of LLM observability tools. ... I just think it was the obvious go-to for langchain developers when ...", "Rakuten Group builds with LangChain and LangSmith to deliver premium products for its business clients\nLangChain Partners with CommandBar on their Copilot User Assistant\nLangChain partners with Elastic to launch the Elastic AI Assistant\nAlly Financial Collaborates with LangChain to Deliver Coding Module to Mask PII\nLLMs accelerate Adyen's support team through\nsmart-ticket routing and support agent copilot\nMorningstar Intelligence Engine puts personalized investment insights at analysts' fingertips\nRobocorp’s code generation assistant makes building Python automation easy for developers\nLangChain Expands Collaboration with Microsoft\nHear from our happy customers\nLangSmith helps teams of all sizes, across all industries, from ambitious\nstartups to established enterprises.\n We could have built evaluation, testing and monitoring tools in house, but with LangSmith it took us 10x less time to get a 1000x better tool.”\nReady to start shipping\nreliable GenAI apps faster?\nLangChain and LangSmith are critical parts of the reference\narchitecture to get you from prototype to production. We couldn’t have achieved \xa0the product experience delivered to our customers without LangChain, and we couldn’t have done it at the same pace without LangSmith.”\n“As soon as we heard about LangSmith, we moved our entire development stack onto it. We couldn’t have achieved \xa0the product experience delivered to our customers without LangChain, and we couldn’t have done it at the same pace without LangSmith.”\n“As soon as we heard about LangSmith, we moved our entire development stack onto it. We could have built evaluation, testing and monitoring tools in house, but with LangSmith it took us 10x less time to get a 1000x better tool.”\n“LangSmith helped us improve the accuracy and performance of Retool’s fine-tuned models.", "By visualizing the exact sequence of events in complex chains that retrieve context from various sources, LangSmith provided insights into the inputs and outputs of LLM calls and ensured that the conversational aspects of Mortgage AI, ContentAI, and PolicyAI were logical, precise, and user-friendly.\n He further added that the company intends to continue working with LangChain to further its AI capabilities and become a truly AI-powered mortgage company that delivers the best experience for both its employees and clients.\n Meet InstaAI\nLeading the transformation, InstaMortgage joined forces with Aptford to create InstaAI, an AI platform reshaping the mortgage landscape with three core modules powered by Langchain & LangSmith:\nWith InstaAI, Maya's role underwent a radical change. InstaMagic of LangChain & LangSmith\nLangChain's LCEL has been instrumental in developing InstaAI's three core modules, simplifying the construction of dynamic prompts across various data sources, most of which are more than 500 pages long and include tabular data. Here's how it works:\nLCEL has simplified the developer experience to a couple of lines of Python for each of the steps above.\n", 'Open-Source: LangChain is open-source, which means it is free to use and has an active community contributing to its development. Cons. Limited Scalability: LangChain is not designed for large-scale production environments, making it less suitable for complex, high-traffic applications. Debugging Challenges: LangChain can be difficult to debug ...', 'In

the ever-evolving landscape of AI and ML, context-aware reasoning applications will continue t
o be at the forefront of innovation, and LangChain is playing a crucial role in this exciting
journey.\n Building Context-Aware Reasoning Applications with LangChain and LangSmith\nThis bl
og post is part of the Ray Summit 2023 highlights series where we provide a summary of the mos
t exciting talk from our recent LLM developer conference.\n LangChain, with its innovative too
ls like LangSmith and its commitment to tackling the complexities of data engineering, prompt
engineering, debugging, evaluation, and collaboration, is playing a significant role in advanc
ing this field.\n Harrison discusses the challenges and solutions in the development of contex
t-aware reasoning applications, offering a deep dive into the LangChain ecosystem, including L
angSmith. Parent Document Retriever\nThe Parent Document Retriever is designed to bridge the g
ap between semantic search over small chunks of data and providing the language model with mor
e extensive context.', "To view or add a comment, sign in\nMore articles by this author\nThe R
esurgence of Tech: A McKinsey Report on Top Trends\nUnveiling Corporate Risks Through Generati
ve #AI: A Comprehensive Analysis\nNavigating the Future: A Comparative Analysis of the UK's Pr
o-Innovation Approach to AI Regulation\nTransforming Public Service Delivery with Product Thin
king: A Vision for Decision Makers\nThe Importance of Skill Development\nArabization in NLP\nO
pportunities and Challenges Presented by The Innovative Characteristics of ChatGPT\nChatGPT Pr
eamble\nAdoption as an opportunity and risk - The case of artificial intelligence\n Join now\n
Insights from the community\nOthers also viewed\nThe Rise of Large Language Models: How AI is
Changing the Way We Communicate\nHow LLMs Influence Software Engineering and Development\nThe
Rise of Artificial Intelligence in Marketing: The role of artificial intelligence tools and te
chniques in business and the global economy\nSign in\nStay updated on your professional world
\nBy clicking Continue, you agree to LinkedInâ€™s User Agreement, Privacy Policy, and Cookie P
olicy.\n The Transformative Impact of Large Language Models on Software Engineering\nRadouane
Monhem\nOver the years, the landscape of software engineering has changed dramatically, with n
ew methodologies, tools, and frameworks constantly emerging to streamline the development proc
ess. Large Language Models (LLMs) like GPT-4 have the potential to play a pivotal role in the
foreseeable future of software engineering as we move further into the era of artificial intel
ligence.", "Or are LLMs creating more hype than functionality for software development, and, a
t the same time, plunging everyone into a world where it is hard to distinguish the perfectly
formed, yet sometimes fake and incorrect, code generated by artificial intelligence (AI) progr
ams from verified and well-tested systems?\nLLMs and Their Potential Impact on the Future of S
oftware Engineering\nThis blog post, which builds on ideas introduced in the IEEE paper Applic
ation of Large Language Models to Software Engineering Tasks: Opportunities, Risks, and Implic
ations by Ipek Ozkaya, focuses on opportunities and cautions for LLMs in software development,
the implications of incorporating LLMs into software-reliant systems, and the areas where more
research and innovations are needed to advance their use in software engineering. BibTeX Code
\n@misc{ozkaya_2023,author={Ozkaya, Ipek and Carleton, Anita and Robert, John and Schmidt, Dou
glas},title={Application of Large Language Models (LLMs) in Software Engineering: Overblown Hy
pe or Disruptive Change?},month={Oct},year={2023},howpublished={Carnegie Mellon University, So
ftware Engineering Institute's Insights (blog)},url={https://doi.org/10.58012/6n1p-pw64},note=
{Accessed: 2023-Nov-22}}\nApplication of Large Language Models (LLMs) in Software Engineering:
Overblown Hype or Disruptive Change?\nSoftware Engineering Research and Development\nHas the d
ay finally arrived when large language models (LLMs) turn us all into better software engineer
s? Here are some research areas related to software engineering where we expect to see signifi
cant focus and progress in the near future:\nThe Way Forward in LLM Innovation for Software En
gineering\nAfter the two winters of AI in the late 1970s and early 1990s, we have entered not
only a period of AI blossoms, but also exponential growth in funding, use, and alarm about AI.
While the excitement around LLMs continues, the jury is still out on whether AI-augmented soft
ware development powered by generative AI tools and other automated methods and tools will ach
ieve the following ambitious objectives:\nEven if a fraction of the above is accomplished, it
will influence the flow of activities in the SDLC, likely enabling and accelerating the shift-
left actions in software engineering. The reaction of the software engineering community to th
e accelerated advances that LLMs have demonstrated since the final quarter of 2022 has ranged
from snake oil to no help for programmers to the end of programming and computer science educa
tion as we know it to revolutionizing the software development process."]}}

```
{'generate': {'draft': "**Title: Unveiling the Contrasts: LangChain vs. LangSmith in the Realm
of Language Model Development**\n\nI. Introduction\nLangChain and LangSmith are two prominent
tools in the field of language model development, each offering unique features and capabiliti
es. This essay aims to delve into the distinctions between LangChain and LangSmith to provide
a comprehensive understanding of their functionalities and applications.\n\nII. LangChain\nLan
gChain is an open-source Python package designed for rapid prototyping of language model appli
cations. It enables developers to create prototypes swiftly and efficiently, making it an exce
llent choice for early-stage projects. With its versatility and ease of use, LangChain is idea
l for small-scale applications that require quick development cycles.\n\nIII. LangSmith\nIn co
ntrast, LangSmith is tailored for large-scale, production-ready applications that demand advan
ced debugging, testing, and monitoring functionalities. It provides robust tools for ensuring
the scalability and performance of language model applications, making it a preferred option f
or complex and high-traffic environments.\n\nIV. Comparison between LangChain and LangSmith\n
A. Technology Stack:\nLangChain utilizes Python as its primary programming language, offering
flexibility and simplicity in development. On the other hand, LangSmith may support a broader
range of languages and frameworks to cater to diverse development needs.\n\nB. Scalability and
Performance:\nWhile LangChain excels in rapid prototyping and small-scale applications, LangSm
ith shines in scalability and performance optimization for large-scale deployments. LangSmit
h's advanced capabilities make it suitable for handling complex and high-traffic scenarios eff
ectively.\n\nC. Security and Privacy:\nBoth LangChain and LangSmith prioritize security and pr
ivacy in language model development. However, LangSmith may offer more robust security feature
s and compliance measures to meet the stringent requirements of enterprise-grade application
s.\n\nD. Adoption and Popularity:\nLangChain, being open-source and user-friendly, may have a
broader community of developers contributing to its growth. In comparison, LangSmith's focus o
n enterprise solutions may attract a niche audience looking for advanced features and suppor
t.\n\nE. Future Prospects:\nThe future of language model development is promising with tools l
ike LangChain and LangSmith leading the innovation. While LangChain may continue to evolve wit
h new features and enhancements for rapid prototyping, LangSmith is poised to expand its capab
ilities for addressing the evolving needs of large-scale applications.\n\nV. Conclusion\nIn co
nclusion, the distinct characteristics of LangChain and LangSmith cater to different stages an
d requirements of language model development. Understanding the nuances between these tools is
crucial for developers to choose the right platform based on their project needs and goals. As
the landscape of language model development evolves, both LangChain and LangSmith are set to p
lay pivotal roles in shaping the future of this dynamic industry.", 'revision_number': 3}}
```

In [ ]:

Essay Writer Interface

In [26]:
```
import warnings
warnings.filterwarnings("ignore")

from helper import ewriter, writer_gui
```

In [27]:
```
MultiAgent = ewriter()
app = writer_gui(MultiAgent.graph)
app.launch()
```

```
Running on local URL:  http://0.0.0.0:8080
Running on public URL: https://75f2ca334f9d66aefa.gradio.live

This share link expires in 72 hours. For free permanent hosting and GPU upgrades, run `gradio
deploy` from Terminal to deploy to Spaces (https://huggingface.co/spaces)
```

Agent | Plan | Research Content | Draft | Critique | StateSnapShots

Essay Topic

Pizza Shop

Generate Essay

Continue Essay

last node | next node | Thread | Draft Rev | count

Manage Agent

Live Agent Output

Use via API · Built with Gradio

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: