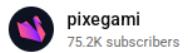


Python RAG Tutorial (with Local LLMs): AI For Your PDFs



pixegami
75.2K subscribers

Subscribe

15K | | Share | Ask | Download | ...

566,164 views Apr 17, 2024

Learn how to build a RAG (Retrieval Augmented Generation) app in Python that can let you query/chat with your PDFs using generative AI.

This project contains some more advanced topics, like how to run RAG apps locally (with Ollama), how to update a vector DB with new items, how to use RAG with PDFs (or any other files), and how to test the quality of AI generated responses.

🔗 Links

- 🔗 GitHub: <https://github.com/pixegami/rag-tutor...>
- 🔗 Basic RAG Tutorial: RAG + Langchain Python Project: Easy AI/Ch...
- 🔗 PyTest Video: How To Write Unit Tests in Python • Pytest...

🔗 Resources

- 🔗 Document loaders: <https://python.langchain.com/docs/mod...>
- 🔗 PDF Loader: <https://python.langchain.com/docs/mod...>
- 🔗 Ollama: <https://ollama.com>

📘 Chapters

- 00:00 Introduction
- 01:06 RAG Recap
- 03:22 Loading PDF Data
- 05:08 Generate Embeddings
- 07:16 How To Store and Update Data
- 10:46 Updating Database
- 11:45 Running RAG Locally
- 15:12 Unit Testing AI Output
- 20:29 Wrapping Up

The screenshot shows the GitHub repository page for 'rag-tutorial-v2'. It includes the repository name, a 'Code' tab, a file list, and sections for 'About', 'Releases', and 'Packages'.

About
An Improved Langchain RAG Tutorial (v2) with local LLMs, database updates, and testing.

Code

| File | Description | Last Commit |
|---------------------------|--|-------------|
| data | add project | last year |
| .gitignore | add project | last year |
| README.md | first commit | last year |
| get_embedding_function.py | add project | last year |
| populate_database.py | add project | last year |
| query_data.py | add project | last year |
| requirements.txt | Update requirements.txt to include boto3 | last year |
| test_rag.py | add project | last year |

Releases
No releases published

Packages



```
~$ python query_data.py "How do I build a hotel in Monopoly?"  
  
Response: To build a hotel, you must first have all  
houses built on every property of a completed color group.  
Then, you can buy a hotel from the Bank and erect it on any property  
of that color group.
```

```
Sources: ['data/monopoly.pdf:0:0', 'data/monopoly.pdf:5:1']
```

```
Expected Response: $1500  
Actual Response: A player starts with $1,500 in Monopoly.  
---  
(Answer with 'true' or 'false') Does the actual response match the expected response?  
  
Response: true. the expected response is a description of an amount of money, and the actual response describes that a player starts with $1,500 in monopoly. both responses convey the same information.  
.Response: The longest continuous train gets a bonus of 10 points.  
Sources: ['data/ticket_to_ride.pdf:3:3', 'data/ticket_to_ride.pdf:3:1', 'data/ticket_to_ride.pdf:0:1', 'data/ticket_to_ride.pdf:3:2', 'data/ticket_to_ride.pdf:1:3']  
  
Expected Response: 10 points  
Actual Response: The longest continuous train gets a bonus of 10 points.  
---  
(Answer with 'true' or 'false') Does the actual response match the expected response?  
  
Response: true. the expected response and the actual response both award 10 points for the longest continuous train.  
.===== 2 passed in 22.70s =====  
  
advanced-rag on ✘ main [!] via ✪ v3.10 [base] on px-beta-au:<ap-southeast-2> took 23s → |
```



RETRIEVAL AUGMENTED GENERATION

```
advanced-rag on ✘ main [!] via ✪ v3.10 [base] on px-beta-au:<ap-southeast-2> →  
python query_data.py "How do I build a hotel in Monopoly?"  
  
Response: To build a hotel in Monopoly, you need to have four houses on each property of a complete color-group first. Then, you can buy a hotel from the Bank and erect it on any property of that color-group. You return the four houses from that property to the Bank and pay the price for the hotel as shown on the Title Deed card. Only one hotel may be erected on any one property.  
Sources: ['data/monopoly.pdf:0:0', 'data/monopoly.pdf:5:1', 'data/monopoly.pdf:5:2', 'data/monopoly.pdf:1:1', 'data/monopoly.pdf:1:0']
```

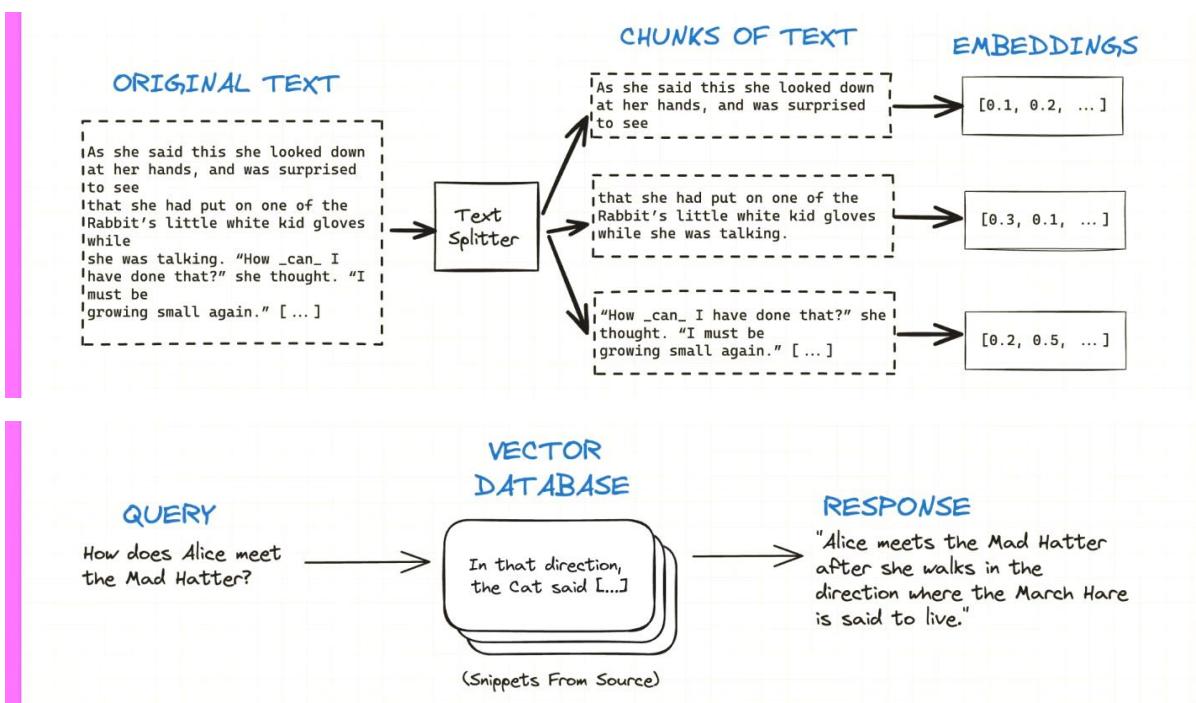
```
advanced-rag on ✘ main [!] via ✪ v3.10 [base] on px-beta-au:<ap-southeast-2> took 14s →
```

llama serve -H local-llm-llama (llama)

```

{"function":"print_timings","level":"INFO","line":272,"msg":"prompt eval time = 325.35 ms / 57 tokens (5.71 ms per token, 175.20 tokens per second)", "n_prompt_tokens_processed":57, "n_tokens_second":175.19648131698577, "slot_id":0, "t_prompt_processing":325.349, "t_token":5.707877192982456, "task_id":82, "tid": "0x177b97000", "timestamp":1712975483}
{"function":"print_timings","level":"INFO","line":286,"msg":"generation eval time = 695.14 ms / 22 runs (31.60 ms per token, 31.65 tokens per second)", "n_decoded":22, "n_tokens_second":31.648301061656646, "slot_id":0, "t_token":31.597272727272728, "t_token_generation":695.14, "task_id":82, "tid": "0x177b97000", "timestamp":1712975483}
{"function":"print_timings","level":"INFO","line":295,"msg":"total time = 1020.49 ms", "slot_id":0, "t_prompt_processing":325.349, "t_token_generation":695.14, "t_total":1020.489, "task_id":82, "tid": "0x177b97000", "timestamp":1712975483}
{"function":"update_slots","level":"INFO","line":1642,"msg":"slot released", "n_cache_tokens":83, "n_ctx":2048, "n_past":82, "n_system_tokens":0, "slot_id":0, "task_id":82, "tid": "0x177b97000", "timestamp":1712975483, "truncated":false}
[GIN] 2024/04/13 - 12:31:23 | 200 | 1.022267625s | 127.0.0.1 || POST /api/generate
{"function":"launch_slot_with_data","level":"INFO","line":829,"msg":"slot is processing task", "slot_id":0, "task_id":107, "tid": "0x177b97000", "timestamp":1712975747}
{"function":"update_slots","level":"INFO","line":1810,"msg":"slot progression", "n_past":4, "n_past_se":0, "n_prompt_tokens_processed":1142, "slot_id":0, "task_id":107, "tid": "0x177b97000", "timestamp":1712975747}
{"function":"update_slots","level":"INFO","line":1834,"msg":"kv cache rm [p0, end]", "p0":4, "slot_id":0, "task_id":107, "tid": "0x177b97000", "timestamp":1712975747}
{"function":"print_timings","level":"INFO","line":272,"msg":"prompt eval time = 8174.72 ms / 1142 tokens (7.16 ms per token, 139.70 tokens per second)", "n_prompt_tokens_processed":1142, "n_tokens_second":139.69895730998036, "slot_id":0, "t_prompt_processing":8174.721, "t_token":7.158249562171628, "task_id":107, "tid": "0x177b97000", "timestamp":1712975758}
{"function":"print_timings","level":"INFO","line":286,"msg":"generation eval time = 3281.48 ms / 89 runs (36.87 ms per token, 27.12 tokens per second)", "n_decoded":89, "n_tokens_second":27.121908407182126, "slot_id":0, "t_token":36.870561797752806, "t_token_generation":3281.48, "task_id":107, "tid": "0x177b97000", "timestamp":1712975758}
{"function":"print_timings","level":"INFO","line":295,"msg":"total time = 11456.20 ms", "slot_id":0, "t_prompt_processing":8174.721, "t_token_generation":3281.48, "t_total":11456.201, "task_id":107, "tid": "0x177b97000", "timestamp":1712975758}
{"function":"update_slots","level":"INFO","line":1642,"msg":"slot released", "n_cache_tokens":1235, "n_ctx":2048, "n_past":1234, "n_system_tokens":0, "slot_id":0, "task_id":107, "tid": "0x177b97000", "timestamp":1712975758, "truncated":false}
[GIN] 2024/04/13 - 12:35:58 | 200 | 11.469743666s | 127.0.0.1 || POST /api/generate"

```



👉 Running RAG Locally

👉 Updating Your Database

👉 Testing RAG Output

github.com/pixegami/rag-tutorial-v2/blob/main/query_data.py

Files

- main
- data
- .gitignore
- README.md
- get_embedding_function.py
- populate_database.py
- query_data.py
- requirements.txt
- test_rag.py

rag-tutorial-v2 / query_data.py

Code Blame 53 lines (37 loc) + 1.46 KB

```
10 PROMPT_TEMPLATE = """  
14  
15 ----  
16 Answer the question based on the above context: {question}  
17 """  
18  
21 def main():  
22     # Create CLI.  
23     parser = argparse.ArgumentParser()  
24     parser.add_argument("query_text", type=str, help="The query text.")  
25     args = parser.parse_args()  
26     query_text = args.query_text  
27     query_rag(query_text)  
28  
29  
30 def query_rag(query_text: str):  
31     # Prepare the DB.  
32     embedding_function = get_embedding_function()  
33     db = Chroma(persist_directory=CHROMA_PATH, embedding_function=embedding_function)  
34  
35     # Search the DB.  
36     results = db.similarity_search_with_score(query_text, k=5)  
37  
38     context_text = "\n\n----\n\n".join([doc.page_content for doc, _score in results])  
39     prompt_template = ChatPromptTemplate.from_template(PROMPT_TEMPLATE)  
40     prompt = prompt_template.format(context=context_text, question=query_text)  
41     # print(prompt)  
42  
43     model = Ollama(model="mistral")  
44     response_text = model.invoke(prompt)  
45  
46     sources = [doc.metadata.get("id", None) for doc, _score in results]  
47     formatted_response = f"Response: {response_text}\nSources: {sources}"  
48     print(formatted_response)  
49     return response_text  
50  
51 if __name__ == "__main__":  
52     main()
```

pip install langchain # LLM Library
pip install chromadb # Vector storage
pip install pypdf # Loading PDFs
pip install pytest # Unit Testing

LOADING THE DATA

MONOPOLY
Property Trading Game from Parker Brothers®

AGES 8+
2 to 8 Players

Contents: Gameboard, 3 dice, tokens, 32 houses, 12 hotels, Chance and Community Chest cards, Title Deed cards, play money and a Banker's tray.

Now there's a faster way to play MONOPOLY. Choose to play by the classic rules for buying, renting and selling properties or use the Speed Die to get into the action faster. If you've never played the classic MONOPOLY game, refer to the Classic Rules beginning on the next page. If you already know how to play and want to use the Speed Die, just read the section below for the additional Speed Die rules.

SPEED DIE RULES

Learning how to play with the Speed Die is as fast as playing with it.

- When starting the game, hand out an extra \$1,000 to each player (two \$500s should work). The game moves fast and you'll need the extra cash to buy and build.
- Do not use the Speed Die until you've landed on or passed over GO for the first time. Once you collect that first \$200 salary, you'll use the Speed Die for the rest of the game. This means that some players will start using the die before others.
- Once you start using the Speed Die, roll it along with the two white dice on your turn. Then do the following depending on what you rolled:
 - 1, 2, or 3: Add this number to the roll of the two white dice. You'll zoom around the board.

A few minor details:

- Only the white dice are used when determining if you rolled doubles. Do not look at the Speed Die.
- If you roll a three-of-a-kind (all of the dice show the same number), you can move anywhere you want on the board!
- If you get sent to jail during your move (either by landing on the "Go to Jail" space or by rolling doubles three times in a row) then your turn is over and you do not get to use the Speed Die for that turn.
- Use the white dice ONLY when rolling to get out of jail.
- Use the sum of all three dice when determining how much to pay on an utility. Note: The Bus and Mr. Monopoly are valued at 0.

CLASSIC MONOPOLY RULES

OBJECT: The object of the game is to become the wealthiest player through buying, renting and selling property.

PREPARATION: Place the board on a table and put the Chance and Community Chest cards face down on their allotted spaces on the board. Each player chooses one token to represent him/herself while traveling around the board.

Each player is given \$1,500 divided as follows: 2 each of \$500s, \$100s and \$50s; 6 \$20s; 5 each of \$10s, \$5s and \$1s.

All remaining money and other equipment go to the Bank. Stack the Bank's money on edge in the compartments in the plastic Banker's tray.

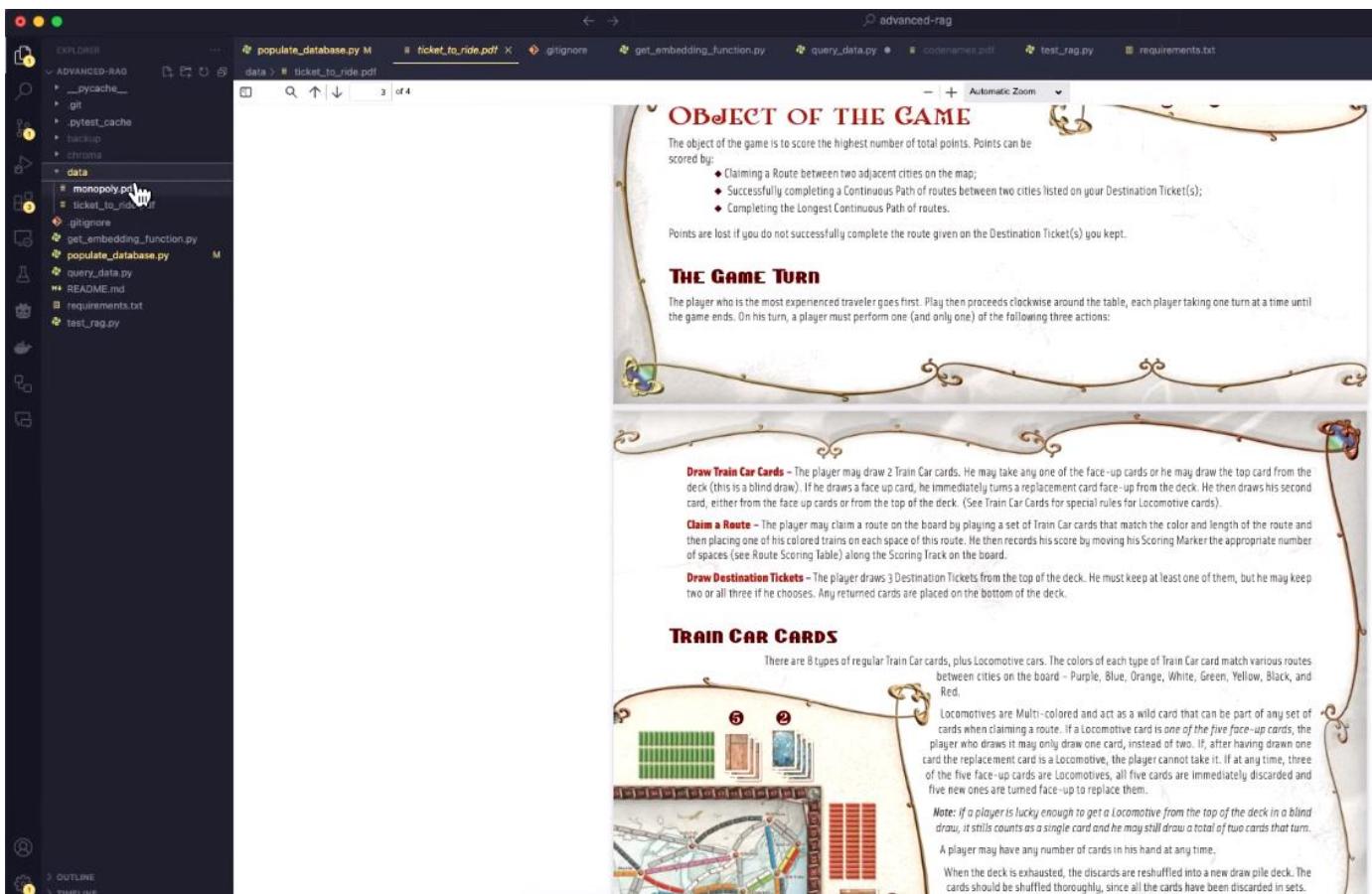
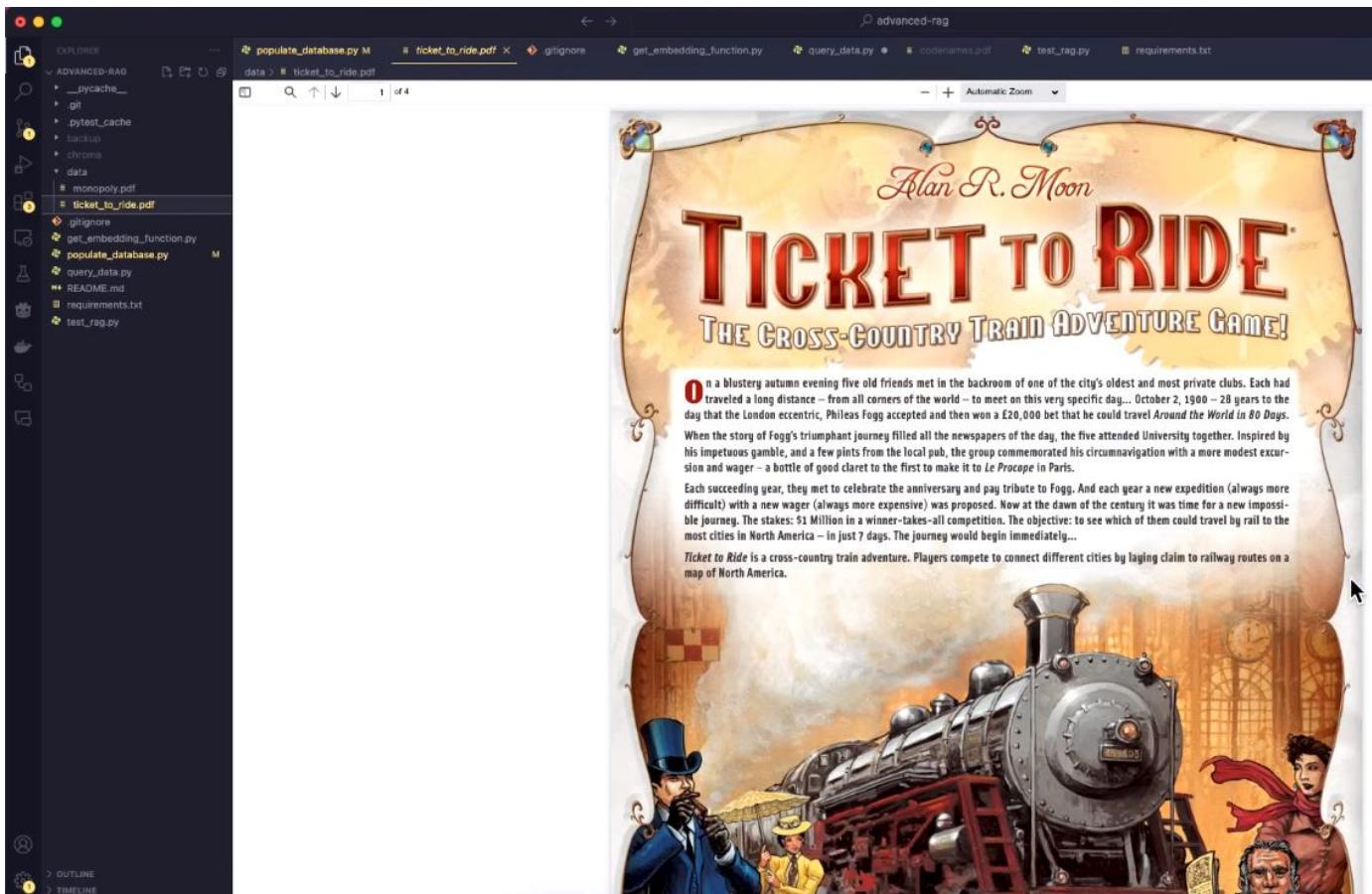
BANKER: Select as Banker a player who will also make a good Auctioneer. A Banker who plays in the game must keep his/her personal funds separate from those of the Bank. When more than five persons play, the Banker may elect to act only as Banker and Auctioneer.

THE BANK Besides the Bank's money, the Bank holds the Title Deed cards and houses and hotels prior to purchase and use by the players. The Bank pays salaries and bonuses. It sells and auctions properties and hands out their proper Title Deed cards; it sells houses and hotels to the players and loans money when required on mortgages.

The Bank collects all taxes, fines, loans and interest, and the price of all properties which it sells and auctions.

The Bank never "goes broke." If the Bank runs out of money, the Banker may issue as much more as needed by writing on any ordinary paper.

THE PLAY: Starting with the Banker, each player in turn throws the dice.



```

from langchain.document_loaders.pdf import PyPDFDirectoryLoader

def load_documents():
    document_loader = PyPDFDirectoryLoader(DATA_PATH)
    return document_loader.load()

```

This is the code to load the documents from the data folder in our code

Document loaders

INFO
Head to [Integrations](#) for documentation on built-in document loader integrations with 3rd-party tools.

Use document loaders to load data from a source as Document's. A Document is a piece of text and associated metadata. For example, there are document loaders for loading a simple .txt file, for loading the text contents of any web page, or even for loading a transcript of a YouTube video.

Document loaders provide a "load" method for loading data as documents from a configured source. They optionally implement a "lazy load" as well for lazily loading data into memory.

Get started

The simplest loader reads in a file as text and places it all into one document.

```

from langchain_community.document_loaders import TextLoader
loader = TextLoader("./index.md")
loader.load()

```

[
 Document(page_content='---\n---\n# Document loaders\n\nUse document loaders to
]

You can also load other types of documents by using the LangChain loaders here or integrations below

Document loaders

- acroom**
acroom is a dev-first knowledge base with tasks
- AirbyteLoader**
Airbyte is a data integration
- Airbyte CDK (Deprecated)**
Note: AirbyteCDKLoader is deprecated. Please use
- Airbyte Gong (Deprecated)**
Note: This connector-specific loader is deprecated. Please use
- Airbyte Hubspot (Deprecated)**
Note: AirbyteHubspotLoader is deprecated. Please use
- Airbyte JSON (Deprecated)**
Note: AirbyteJSONLoader is deprecated. Please use

```
● ● ●  
documents = load_documents()  
print(documents[0])
```

```
● ● ●  
page_content='If you roll a three-of-a-kind [...]'  
metadata={'source': 'data/monopoly.pdf', 'page': 0}
```

SPLIT THE DOCUMENTS

```
● ● ●  
from langchain_text_splitters import RecursiveCharacterTextSplitter  
from langchain.schema.document import Document  
  
def split_documents(documents: list[Document]):  
    text_splitter = RecursiveCharacterTextSplitter(  
        chunk_size=800,  
        chunk_overlap=80,  
        length_function=len,  
        is_separator_regex=False,  
    )  
    return text_splitter.split_documents(documents)
```

```
● ● ●  
documents = load_documents()  
chunks = split_documents(documents)  
print(chunks[0])
```

EMBEDDING FUNCTION

```
● ● ●  
from langchain_community.embeddings.bedrock import BedrockEmbeddings  
  
def get_embedding_function():  
    embeddings = BedrockEmbeddings(  
        credentials_profile_name="default", region_name="us-east-1"  
    )  
    return embeddings
```

Use an embedding function because we will need it in 2 separate places, when you create the database itself and when we want to query the database. LangChain comes with many different embedding functions like the **BedrockEmbeddings()** above

Chrome File Edit View History Bookmarks Profiles Tab Window Help

Video analytics - YouTube Subs | Embedding models | Ollama | Document loaders | Embedding models | +

python.langchain.com/docs/integrations/text_embedding/

LangChain Components Integrations Guides API Reference More ▾

Providers

- Anthropic
- AWS
- Google
- Hugging Face
- Microsoft
- OpenAI
- More

Components

- Chat models
- LLMs
- Embedding models
- AI21 Labs
- Aleph Alpha
- Anyscale
- AwaDB
- Azure OpenAI
- Baichuan Text Embeddings
- Baidu Qianfan
- Bedrock
- BGE on Hugging Face
- Bookend AI
- Clarifai
- Cloudflare Workers AI
- Cohere

FastEmbed by Qdrant
FastEmbed from

FireworksEmbeddings
This notebook explains how to use Fireworks Embeddings, which is

GigaChat
This notebook shows how to use LangChain with GigaChat.

Google Generative AI Embeddings
Connect to Google's generative AI embeddings service using the

Google Vertex AI PaLM
Vertex AI PaLM

GPT4All
GPT4All is a free-to-use, locally

Gradient
Gradient allows to create Embeddings as well fine tune and get

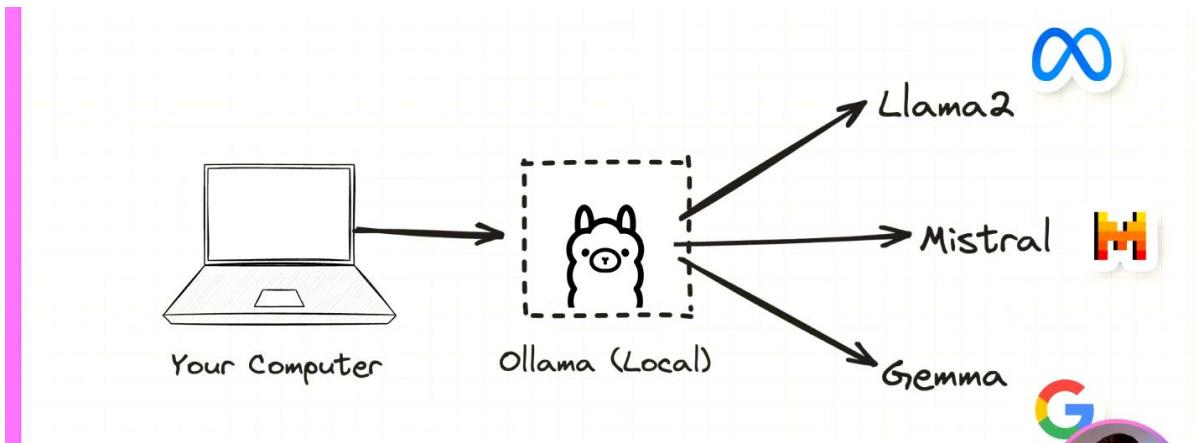
Hugging Face
Let's load the Hugging Face Embedding class.

Infinity
Infinity allows to create Embeddings using a MIT-licensed Embed...

Instruct Embeddings on Hugging Face
Hugging Face

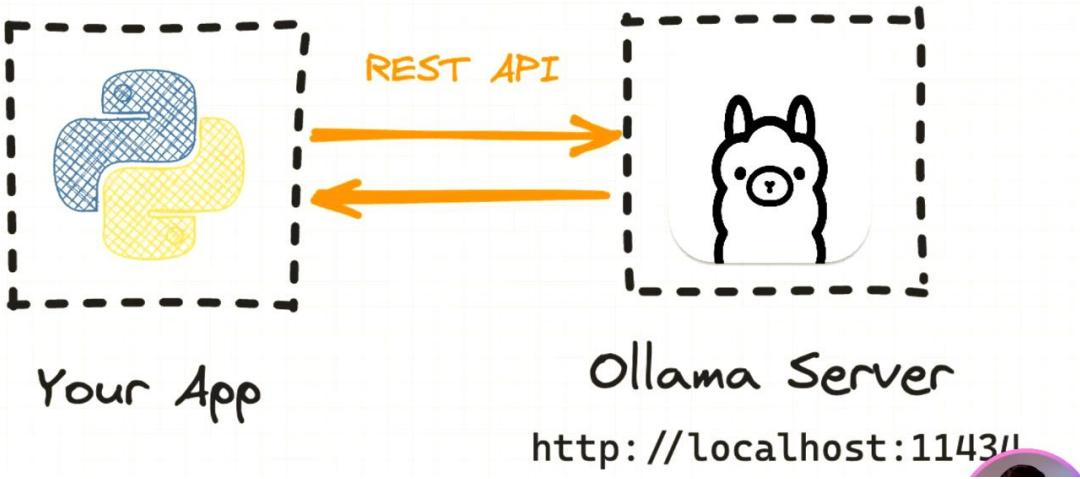
```
from langchain_community.embeddings.ollama import OllamaEmbeddings

def get_embedding_function():
    embeddings = OllamaEmbeddings(model="nomic-embed-text")
    return embeddings
```



```
ollama pull llama2
ollama pull mistral
```

```
ollama serve
```



← → C https://ollama.com/search?c=embedding&q=nomic

Cloud Embedding Vision Tools Thinking Popular ▾

nomic-embed-text

A high-performing open embedding model with a large token context window.

embedding

43.9M Pulls 3 Tags Updated 1 year ago

qwen3-embedding

Building upon the foundational models of the Qwen3 series, Qwen3 Embedding provides a comprehensive range of text embeddings models in various sizes

embedding 0.6b 4b 8b

91.4K Pulls 12 Tags Updated 1 month ago

bge-large

Embedding model from BAAI mapping texts to vectors.

embedding 335m

136.8K Pulls 3 Tags Updated 1 year ago

granite-embedding

The IBM Granite Embedding 30M and 278M models are text-only dense biencoder embedding models, with 30M available in English only and 278M serving multilingual use cases.

embedding 30m 278m

107.5K Pulls 6 Tags Updated 10 months ago

jina/jina-embeddings-v2-base-de

Text embedding model (base) for English and German input of size up to 8192 tokens

embedding

172.7K Pulls 1 Tag Updated 1 year ago

CREATING THE DATABASE

```
from get_embedding_function import get_embedding_function
from langchain.vectorstores.chroma import Chroma

def add_to_chroma(chunks: list[Document]):
    db = Chroma(
        persist_directory=CHROMA_PATH, embedding_function=get_embedding_function()
    )
    db.add_documents(new_chunks, ids=new_chunk_ids)
    db.persist()
```

Once we have the documents splitted into chunks, we can use the embedding function to build a vector database with it

```
page_content='If you roll a three-of-a-kind [...]'
metadata={'source': 'data/monopoly.pdf', 'page': 0}
```

To update the ChromaDB database, we need to tag every item in it with an id value

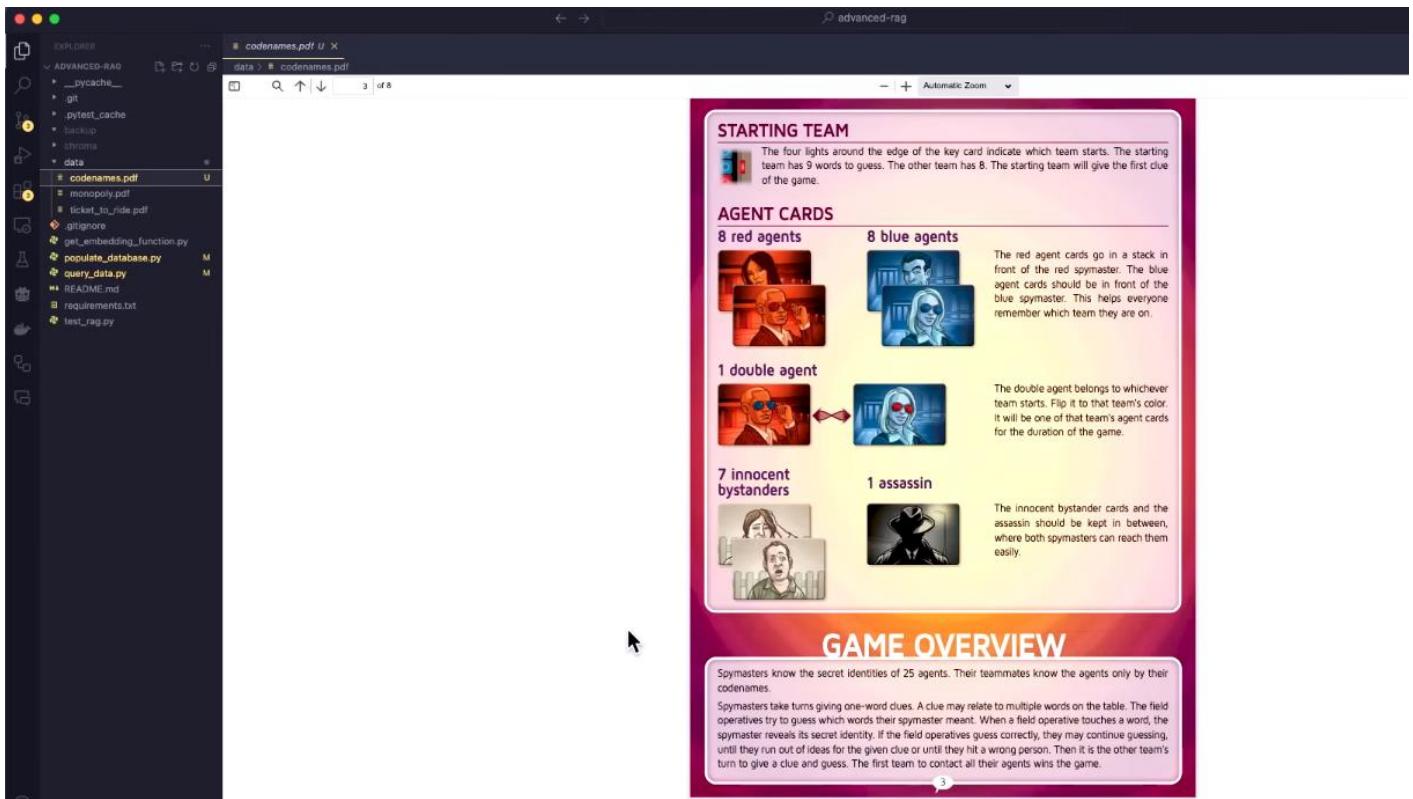
```
for chunk in chunks:
    source = chunk.metadata.get("source")
    page = chunk.metadata.get("page")
    current_page_id = f"{source}:{page}"
```

```
last_page_id = None
current_chunk_index = 0

for chunk in chunks:
    [...]
    # If the page ID is the same as the last one, increment the index.
    if current_page_id == last_page_id:
        current_chunk_index += 1
    else:
        current_chunk_index = 0
```

```
data/monopoly.pdf:0:0
data/monopoly.pdf:0:1
data/monopoly.pdf:0:2
data/monopoly.pdf:1:0
data/monopoly.pdf:1:1
```

```
# Add it to the chunk meta-data.
chunk.metadata["id"] = chunk_id
```



We want the document chunking to only work for new documents added to the data folder, that is why we are using the **document:page:chunk** naming strategy above

```
advanced-rag on ⬢ main [!] via ⓘ v3.10 [base] on px-beta-au:<ap-southeast-2> →
python populate_database.py
Number of existing documents in DB: 41
➡ Adding new documents: 27

advanced-rag on ⬢ main [!?] via ⓘ v3.10 [base] on px-beta-au:<ap-southeast-2> took 13s →
```

We can see that it detected 41 already existing vector chunks inside the database, and we have 27 new document chunks to add from the newly added pdf file dropped in the **data** folder

```
advanced-rag on ⬢ main [!?] via ⓘ v3.10 [base] on px-beta-au:<ap-southeast-2> took 13s →
python populate_database.py
Number of existing documents in DB: 68
 No new documents to add

advanced-rag on ⬢ main [!?] via ⓘ v3.10 [base] on px-beta-au:<ap-southeast-2> took 2s →
```

When we run the same embedding function again against the **data** folder, we can see that it recognized that all the documents chunk already exist in the database and do not need to be added again



But if we modify a pdf file content, we need a way to detect that and replace all the document chunks in the database

UPDATING YOUR DATABASE

```
from langchain.vectorstores.chroma import Chroma

db = Chroma(
    persist_directory=CHROMA_PATH,
    embedding_function=get_embedding_function()
)
```

Now that all chunks have a unique id, we can add them to the vector database

```
# Add or Update the documents.
existing_items = db.get(include=[]) # IDs are always included by default
existing_ids = set(existing_items["ids"])
print(f"Number of existing documents in DB: {len(existing_ids)}")
```

```
# Only add documents that don't exist in the DB.
new_chunks = []
for chunk in chunks_with_ids:
    if chunk.metadata["id"] not in existing_ids:
        new_chunks.append(chunk)
```

```
new_chunk_ids = [chunk.metadata["id"] for chunk in new_chunks]
db.add_documents(new_chunks, ids=new_chunk_ids)
db.persist()
```

RUNNING RAG LOCALLY

```
from langchain_community.embeddings.ollama import OllamaEmbeddings

def get_embedding_function():
    embeddings = OllamaEmbeddings(model="nomic-embed-text")
    return embeddings
```

```
from langchain_community.embeddings.bedrock import BedrockEmbeddings

def get_embedding_function():
    embeddings = BedrockEmbeddings(
        credentials_profile_name="default", region_name="us-east-1"
    )
    return embeddings
```

Local LLMs and local embedding models are good if we can validate them

```
def query_rag(query_text: str):
    ...
```

```
embedding_function = get_embedding_function()
db = Chroma(
    persist_directory=CHROMA_PATH,
    embedding_function=embedding_function
)
```

```
PROMPT_TEMPLATE = """
Answer the question based only on the following context:
{context}

---
Answer the question based on the above context: {question}
"""
```

```
results = db.similarity_search_with_score(query_text, k=5)
```

```
context_text = "\n\n---\n".join([doc.page_content for doc, _score in results])
prompt_template = ChatPromptTemplate.from_template(PROMPT_TEMPLATE)
prompt = prompt_template.format(context=context_text, question=query_text)
```

```
advanced-rag on ⚡ main [!?] via 🐄 v3.10 [base] on px-beta-au:<ap-southeast-2> →
python query_data.py "How many clues can I give in Codenames?"
```

Human:
Answer the question based only on the following context:

"This may be a bit of a stretch..." You are playing Codenames. It's always a bit of a stretch. Your clue cannot be any of the codenames visible on the table. On later turns, some codenames will be covered up, so a clue that is not legal now might be legal later.

MAKING CONTACT

When the spymaster gives a clue, his or her field operatives try to figure out what it means. They can debate it amongst themselves, but the spymaster must keep a straight face. The operatives indicate their official guess when one of them touches one of the codenames on the table.

- If the field operative touches a card belonging to his or her team, the spymaster covers the word with an agent card in that color. The operatives get another guess (but not another clue).

and MARBLE.Acronyms and Abbreviations

Technically, CIA is not one word. But it is a great clue. You can decide to allow common abbreviations like UK, lol, and PhD. And words like laser, radar, and sonar are always allowed, even though they originated as acronyms.

```
advanced-rag on ⚡ main [!?] via 🐄 v3.10 [base] on px-beta-au:<ap-southeast-2> took 15s →
```

```
and MARBLE.Acronyms and Abbreviations
```

Technically, CIA is not one word. But it is a great clue. You can decide to allow common abbreviations like UK, lol, and PhD. And words like laser, radar, and sonar are always allowed, even though they originated as acronyms.

Homonyms

Some people prefer to allow a more liberal use of homonyms. You can allow knight to be a clue for night-related things if that makes the game more fun for you.

Rhymes

Rhymes are always valid when they refer to meanings. Snail is a valid clue for MAIL because this rhyme is a common phrase. Snail is also a valid clue for WHALE because they are both animals. Snail is not a valid clue for SCALE because their main association is through the sound of the words. (If someone in your group

We playtested various rules. Some groups like the rules one way. Some like the rules another way. You should experiment to find out what your group likes.

```
advanced-rag on ⚡ main [!?] via 🐄 v3.10 [base] on px-beta-au:<ap-southeast-2> took 15s →
```

```
---  
We playtested various rules. Some groups like the rules one way. Some like the rules another way.  
You should experiment to find out what your group likes.  
Some clues are invalid because they violate the  
spirit of the game.  
Your clue must be about the meaning of the  
words. You can't use your clue to talk about  
the letters in a word or its position on the table.  
Gland is not a valid clue for ENGLAND. You  
can't tie BUG, BED, and BOW together with  
a c lue like b: 3 n or with a c lue like three: 3 .  
However ...  
Letters and numbers are valid clues, as long  
as they refer to meanings. You can use X: 1 as  
a clue for RAY. You can use eight: 3 as a clue  
for BALL, FIGURE, and OCTOPUS.  
The number you say after your clue can't be  
used as a clue. Citrus: 8 is not a valid clue for  
---  
You can decide to allow any compound words.  
However, in no case should a player be allowed  
advanced-rag on / main [!?] via v3.10 [base] on px-beta-au:<ap-southeast-2> took 15s →
```

```
You can decide to allow any compound words.  
However, in no case should a player be allowed  
to invent compound words. Lunar squid is not  
a valid clue for MOON and OCTOPUS.  
Proper Names  
Proper names are always valid clues if they follow the other rules.  
George is a valid clue, but  
you might want to specify whether you mean  
George Washington or George W. Bush . Your  
group can agree to count proper names as one word. This would also allow titles such as  
The  
Three Musketeers . Even if you don't allow multi-  
word proper names, you might want to make an exception for place names like  
New York .  
Spymasters should not be allowed to make up  
names, not even names that turn out to be real.  
Lily X. Rockne is not a valid clue for ROSE, RAY,  
and MARBLE.Acronyms and Abbreviations  
---  
Teams take turns. The starting team is indicated by the 4 lights on the edges of the key card.  
GIVING A CLUE  
If you are the spymaster, you are trying to think of a one-word clue that relates to some of the words  
your team is trying to guess. When you think you have a good clue, you say it. You also say one  
advanced-rag on / main [!?] via v3.10 [base] on px-beta-au:<ap-southeast-2> took 15s →
```

```
● ○ □ -/P/auto-edit -/P/auto-edit -/I/advanced-rag + Update Warp Q ↻ ⌂
Lilly X. Rockne is not a valid clue for ROSE, RAY,
and MARBLE.Acronyms and Abbreviations

---
4Teams take turns. The starting team is indicated by the 4 lights on the edges of the key card.
GIVING A CLUE
If you are the spymaster, you are trying to think of a one-word clue that relates to some of the words your team is trying to guess. When you think you have a good clue, you say it. You also say one number, which tells your teammates how many codenames are related to your clue.
Example: Two of your words are NUT and BARK. Both of these grow on trees, so you say tree: 2 .
You are allowed to give a clue for only one word ( cashew: 1 ) but it's fun to try for two or more. Getting four words with one clue is a big accomplishment.
One Word
Your clue must be only one word. You are not allowed to give extra hints. For example, don't say,
---

Answer the question based on the above context: How many clues can I give in Codenames?

Response: In Codenames, you can give only one clue per turn, and the clue should be a single word. However, the number after the clue indicates how many codenames are related to that word.
Sources: ['data/codenames.pdf:3:1', 'data/codenames.pdf:6:2', 'data/codenames.pdf:5:0', 'data/codenames.pdf:6:1', 'data/codenames.pdf:3:0']

advanced-rag on ⚙ main [!?] via ⚙ v3.10 [base] on px-beta-au:<ap-southeast-2> took 15s →
```

```
● ○ ●
from langchain_community.llms.ollama import Ollama
model = Ollama(model="mistral")
response_text = model.invoke(prompt)
print(response_text)

sources = [doc.metadata.get("id", None) for doc, _score in results]
```

```
advanced-rag on ⚙ main [!?] via ⚙ v3.10 [base] on px-beta-au:<ap-southeast-2> →
python query_data.py "How do I get out of jail in Monopoly?"
```

```
● ○ □ -/P/auto-edit -/P/auto-edit -/I/advanced-rag + Update Warp Q ↻ ⌂
py query_data.py How do I get out of jail in Monopoly?

Human:
Answer the question based only on the following context:

"JAIL": You land in Jail when. ..(1) your token lands on the space marked "Go to Jail"; (2) you draw a card marked "Go to Jail"; or (3) you throw doubles three times in succession.
When you are sent to Jail you cannot collect your $200 salary in that move since, regardless of where your token is on the board, you must move it directly into Jail. Your turn ends when you are sent to Jail.
If you are not "sent" to Jail but in the ordinary course of play land on that space, you are "Just Visiting," you incur no penalty, and you move ahead in the usual manner on your next turn.
You get out of Jail by... .(1) throwing doubles on any of your next three turns; if you succeed in doing this you immediately move forward the number of spaces shown by your doubles throw; even though you

---
the number of spaces shown by your doubles throw; even though you had thrown doubles, you do not take another turn; (2) using the "Get Out of Jail Free" card if you have it; (3) purchasing the "Get Out of Jail
```

```
● ○ □ ~P/auto-edit ~P/auto-edit ~T/advanced-rag + Update Warp Q ↻ ⌂
that space, you are "Just Visiting," you incur no penalty, and you move
ahead in the usual manner on your next turn.
You get out of Jail by.. .(I) throwing doubles on any of your next
three turns; if you succeed in doing this you immediately move forward
the number of spaces shown by your doubles throw; even though you

---
the number of spaces shown by your doubles throw; even though you
had thrown doubles, you do not take another turn; (2) using the "Get
Out of Jail Free" card, if you have it; (3) purchasing the "Get Out of Jail
Free" card from another player and playing it; (4) paying a fine of $50
before you roll the dice on either of your next two turns.
If you do not throw doubles by your third turn, you must pay the $50
fine. You then get out of Jail and immediately move forward the number
of spaces shown by your throw.
Even though you are in Jail, you may buy and sell property, buy and
sell houses and hotels and collect rents.

---
NO, there are no more properties in the bank - Advance to the
NOCT property on which you will owe another player money.
A few minor details:
Only the white dice are used when determining if you rolled doubles.

advanced-rag on ✎ main [!?] via ✏ v3.10 [base] on px-beta-au:<ap-southeast-2> took 16s →
```

```
● ○ □ ~P/auto-edit ~P/auto-edit ~T/advanced-rag + Update Warp Q ↻ ⌂
According to the space your token reaches, you may be entitled to
buy real estate or other properties - or obliged to pay rent, pay taxes,
draw a Chance or Community Chest card, "Go to Jail," etc.
If you throw doubles, you move your token as usual, the sum of the
two dice, and are subject to any privileges or penalties pertaining to
the space on which you land. Retaining the dice, throw again and move
your token as before. If you throw doubles three times in succession,
move your token immediately to the space marked "In Jail" (see JAIL).

---
Answer the question based on the above context: How do I get out of jail in Monopoly?

Response: To get out of jail in Monopoly, you have several options:

1. Throw doubles with your next three turns using only the white dice. If you succeed, immediately move forward the number
   of spaces shown by your doubles throw.
2. Use a "Get Out of Jail Free" card if you have it.
3. Purchase the "Get Out of Jail Free" card from another player and play it.
4. Pay a fine of $50 before rolling the dice on either of your next two turns. If you do not throw doubles by your third t
urn, you must pay the fine and get out of jail.

While in jail, you may buy and sell property, buy and sell houses and hotels, and collect rents.
Sources: ['data/monopoly.pdf:4:1', 'data/monopoly.pdf:4:2', 'data/monopoly.pdf:1:1', 'data/monopoly.pdf:4:0', 'data/monopo
ly.pdf:2:2']

advanced-rag on ✎ main [!?] via ✏ v3.10 [base] on px-beta-au:<ap-southeast-2> took 16s →
```

It worked!

buy real estate or other properties - or obliged to pay rent, pay taxes, draw a Chance or Community Chest card, "Go to Jail," etc.

If you throw doubles, you move your token as usual, the sum of the two dice, and are subject to any privileges or penalties pertaining to the space on which you land. Retaining the dice, throw again and move your token as before. If you throw doubles three times in succession, move your token immediately to the space marked "In Jail" (see JAIL).

Answer the question based on the above context: How do I get out of jail in Monopoly?

Response: To get out of jail in Monopoly, you have several options:

1. Throw doubles with your next three turns using only the white dice. If you succeed, immediately move forward the number of spaces shown by your doubles throw.
2. Use a "Get Out of Jail Free" card if you have it.
3. Purchase the "Get Out of Jail Free" card from another player and play it.
4. Pay a fine of \$50 before rolling the dice on either of your next two turns. If you do not throw doubles by your third turn, you must pay the fine and get out of jail.

While in jail, you may buy and sell property, buy and sell houses and hotels, and collect rents.

Sources: ['data/monopoly.pdf:4:1', 'data/monopoly.pdf:4:2', 'data/monopoly.pdf:1:1', 'data/monopoly.pdf:4:0', 'data/monopoly.pdf:2:2']

advanced-rag on ⚡ main [!] via 🐀 v3.10 [base] on px-beta-au:<ap-southeast-2> took 16s →

```
oillama serve ~/local-llm-oillama (oillama) M1 ~/(advanced-rag (-fish)) M2
{"function":"print_timings","level":"INFO","line":272,"msg":"prompt eval time = 5932.34 ms / 1219 tokens (4.87 ms per token, 205.48 tokens per second)", "n_prompt_tokens_processed":1219, "n_tokens_second":205.4837008777643, "slot_id":0, "t_prompt_processing":5932.344, "t_token":4.866566037735849, "task_id":0, "tid":0x17679b000, "timestamp":1712978703}
{"function":"print_timings","level":"INFO","line":286,"msg":"generation eval time = 1584.61 ms / 43 runs (36.85 ms per token, 27.14 tokens per second)", "n_decoded":43, "n_tokens_second":27.13598029044334, "slot_id":0, "t_token":36.851441860465115, "t_token_generation":1584.612, "task_id":0, "tid":0x17679b000, "timestamp":1712978703}
{"function":"print_timings","level":"INFO","line":295,"msg":" total time = 7516.96 ms", "slot_id":0, "t_prompt_processing":5932.344, "t_token_generation":1584.612, "t_total":7516.956, "task_id":0, "tid":0x17679b000, "timestamp":1712978703}
{"function":"update_slots","level":"INFO","line":1642,"msg":"slot released", "n_cache_tokens":1262, "n_ctx":2048, "n_past":1261, "n_system_tokens":0, "slot_id":0, "task_id":0, "tid":0x17679b000, "timestamp":1712978703, "truncated":false}
[GIN] 2024/04/13 - 13:25:03 | [ ] 12.992054083s | 127.0.0.1 | POST /api/generate
{"function":"launch_slot_with_data","level":"INFO","line":829,"msg":"slot is processing task", "slot_id":0, "task_id":46, "tid":0x17679b000, "timestamp":17129787979}
{"function":"update_slots","ga_i":0,"level":"INFO","line":1810,"msg":"slot progression", "n_past":23, "n_past_se":0, "n_prompt_tokens_processed":1101, "slot_id":0, "task_id":46, "tid":0x17679b000, "timestamp":17129787979}
{"function":"update_slots","level":"INFO","line":1834,"msg":"kv cache rm [p0, end)", "p0":23, "slot_id":0, "task_id":46, "tid":0x17679b000, "timestamp":17129787979}
{"function":"print_timings","level":"INFO","line":272,"msg":"prompt eval time = 7794.46 ms / 1101 tokens (7.08 ms per token, 141.25 tokens per second)", "n_prompt_tokens_processed":1101, "n_tokens_second":141.2542453251387, "slot_id":0, "t_prompt_processing":7794.456, "t_token":7.079433242506812, "task_id":46, "tid":0x17679b000, "timestamp":1712978993}
{"function":"print_timings","level":"INFO","line":286,"msg":"generation eval time = 6037.39 ms / 163 runs (37.04 ms per token, 27.00 tokens per second)", "n_decoded":163, "n_tokens_second":26.998425975202196, "slot_id":0, "t_token":37.03919631901841, "t_token_generation":6037.389, "task_id":46, "tid":0x17679b000, "timestamp":1712978993}
{"function":"print_timings","level":"INFO","line":295,"msg":" total time = 13831.85 ms", "slot_id":0, "t_prompt_processing":7794.456, "t_token_generation":6037.389, "t_total":13831.845000000001, "task_id":46, "tid":0x17679b000, "timestamp":1712978993}
{"function":"update_slots","level":"INFO","line":1642,"msg":"slot released", "n_cache_tokens":1287, "n_ctx":2048, "n_past":1286, "n_system_tokens":0, "slot_id":0, "task_id":46, "tid":0x17679b000, "timestamp":1712978993, "truncated":false}
[GIN] 2024/04/13 - 13:29:53 | [ ] 13.847065709s | 127.0.0.1 | POST /api/generate
```

TESTING RAG OUTPUT

QUALITY OF ANSWERS

- 👉 Source Material
- 👉 Text Splitting Strategy
- 👉 LLM Model and Prompt



Question: How much total money does a player start with in Monopoly?
Expected Answer: \$1500

```
actual_answer = query_rag(question)
assert actual_answer == expected_answer
```

Error:
Expected "\$1500" but got "1,500 Dollars".

The problem here is that we can't do a strict output string comparison because there are many possible LLM responses

```
EVAL_PROMPT = """
Expected Response: {expected_response}
Actual Response: {actual_response}
---
(Answer with 'true' or 'false') Does the actual response match the expected response?
""
```

```
response_text = query_rag(question)
prompt = EVAL_PROMPT.format(
    expected_response=expected_response,
    actual_response=response_text
)

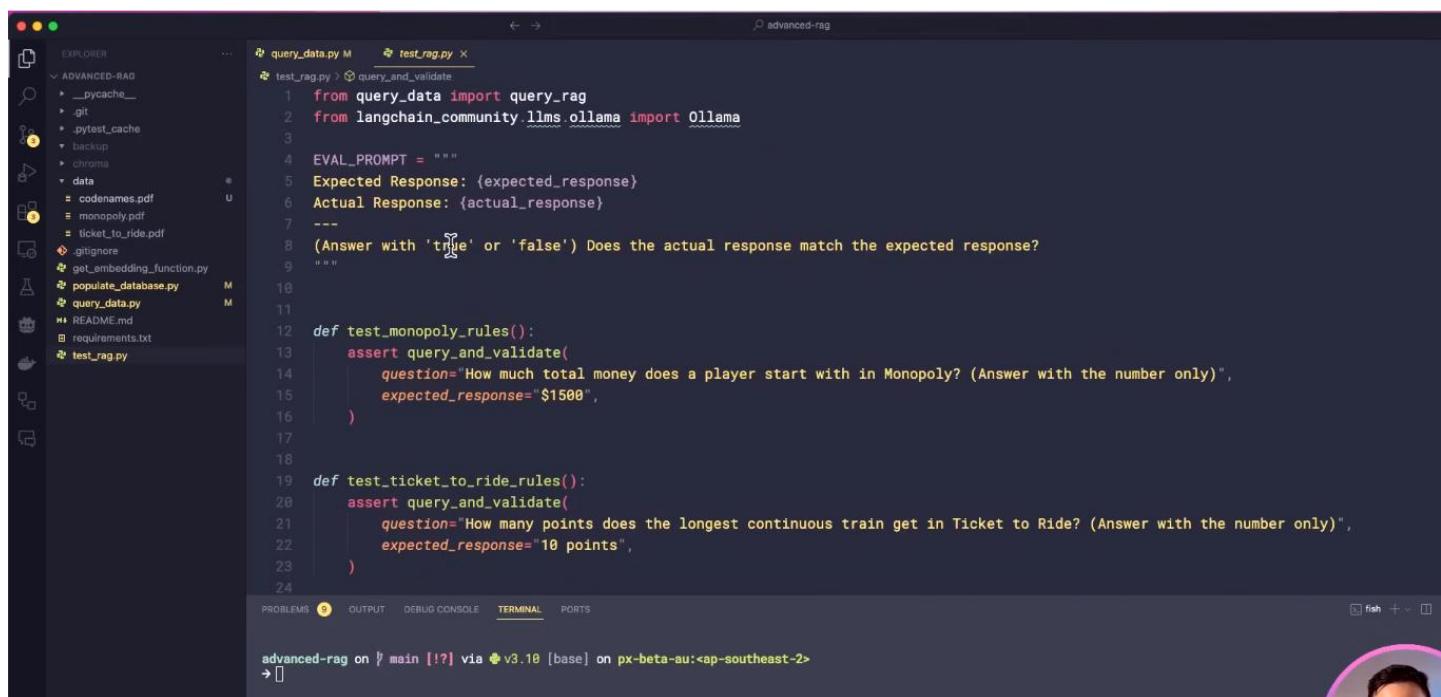
model = Ollama(model="mistral")
evaluation_results_str = model.invoke(prompt)
```

```
final_result = evaluation_results_str.strip().lower()
if "true" in final_result:
    return True
elif "false" in final_result:
    return False
else:
    raise ValueError("Cannot determine if true or false")
```

```
def query_and_validate(question: str, expected_response: str):
    ...
```

We can wrap this into a helper function that will return true or false as above. Then we can write unit tests as below

```
def test_monopoly_rules():
    assert query_and_validate(
        question="How much total money does a player start with in Monopoly?",
        expected_response="$1500",
    )
```



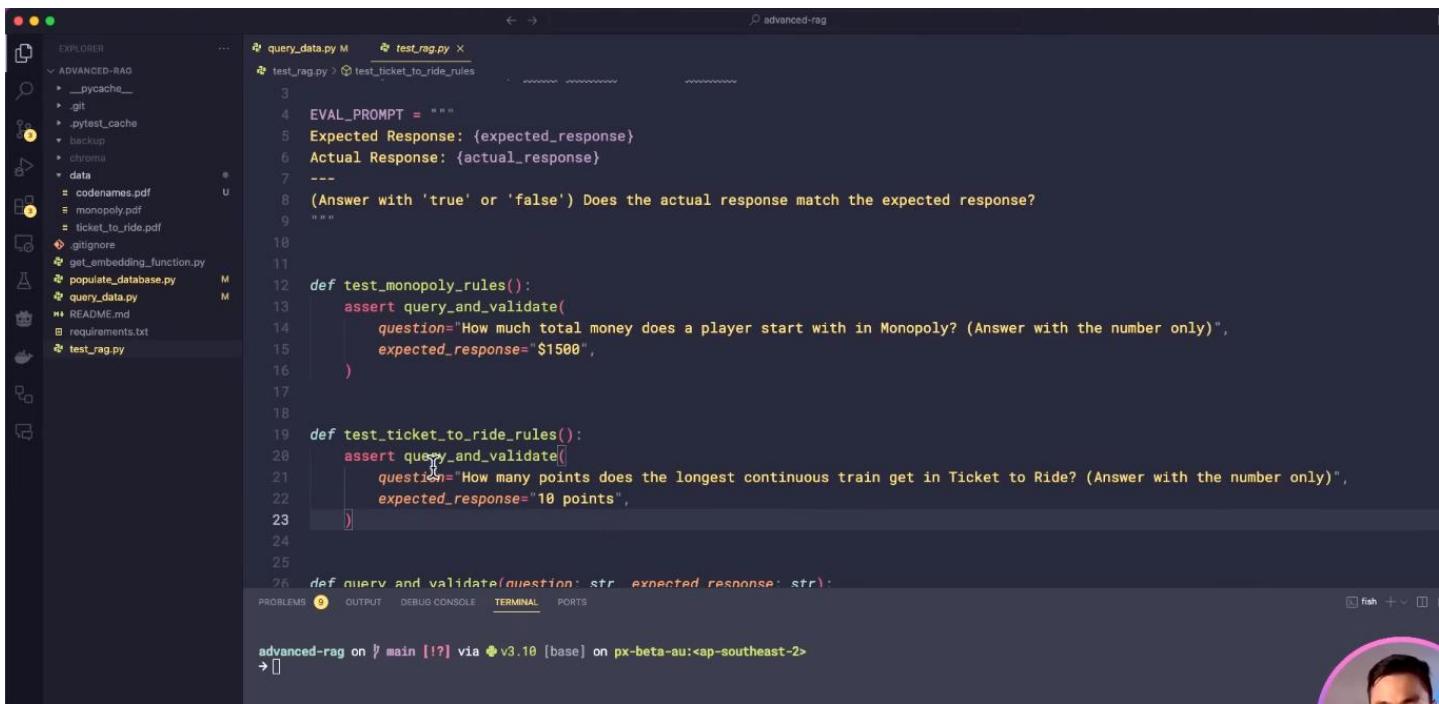
The screenshot shows a dark-themed code editor interface with several files listed in the Explorer sidebar on the left. The main editor area displays Python test code for a system named 'advanced-rag'. The code includes imports for `query_data` and `langchain_community.llms.ollama`, defines an `EXPECTED_RESPONSE` constant, and contains three test functions: `test_monopoly_rules`, `test_ticket_to_ride_rules`, and `test_rag`. The `test_monopoly_rules` function asserts that the response to a Monopoly-related question matches the expected response of '\$1500'. The `test_ticket_to_ride_rules` function asserts that the response to a Ticket to Ride-related question matches the expected response of '10 points'. The code uses triple quotes for multi-line strings and includes docstrings for the test functions.

```
from query_data import query_rag
from langchain_community.llms.ollama import Ollama

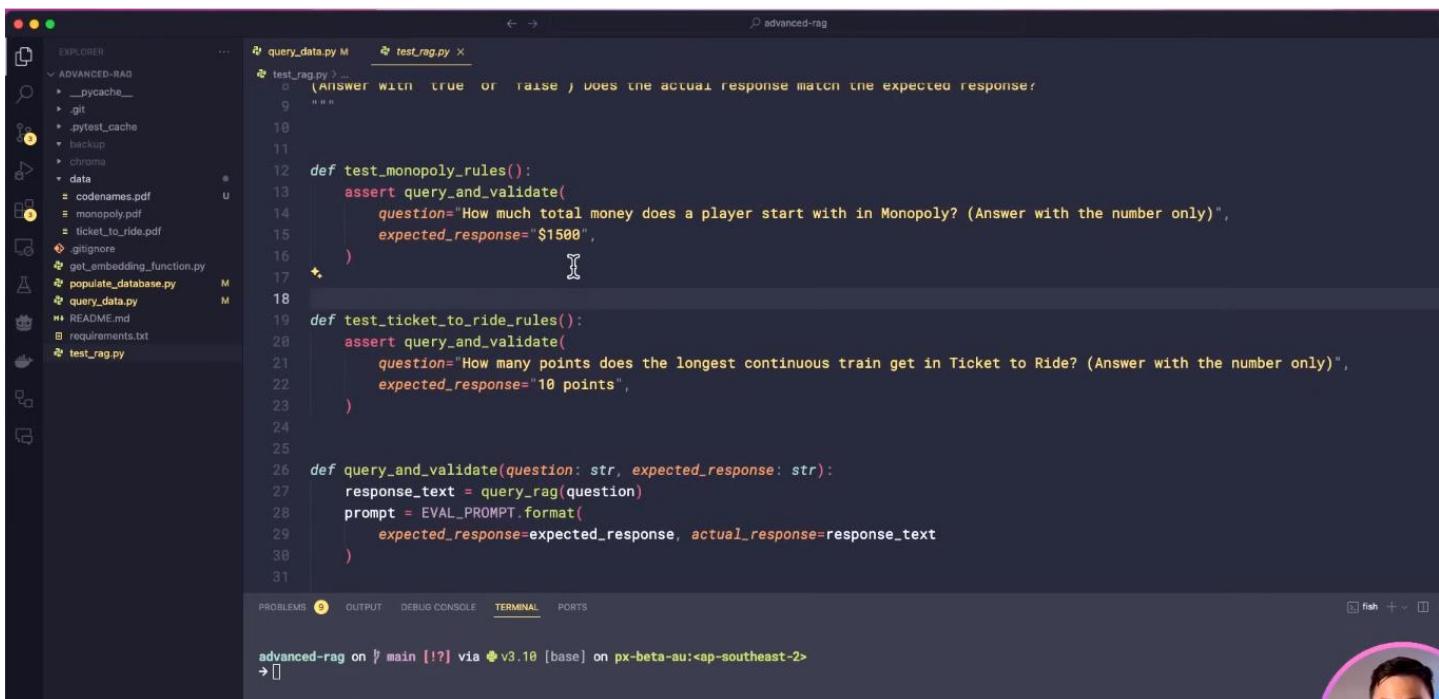
EXPECTED_RESPONSE = """
Expected Response: {expected_response}
Actual Response: {actual_response}
---
(Answer with 'true' or 'false') Does the actual response match the expected response?
"""

def test_monopoly_rules():
    assert query_and_validate(
        question="How much total money does a player start with in Monopoly? (Answer with the number only)",
        expected_response="$1500",
    )

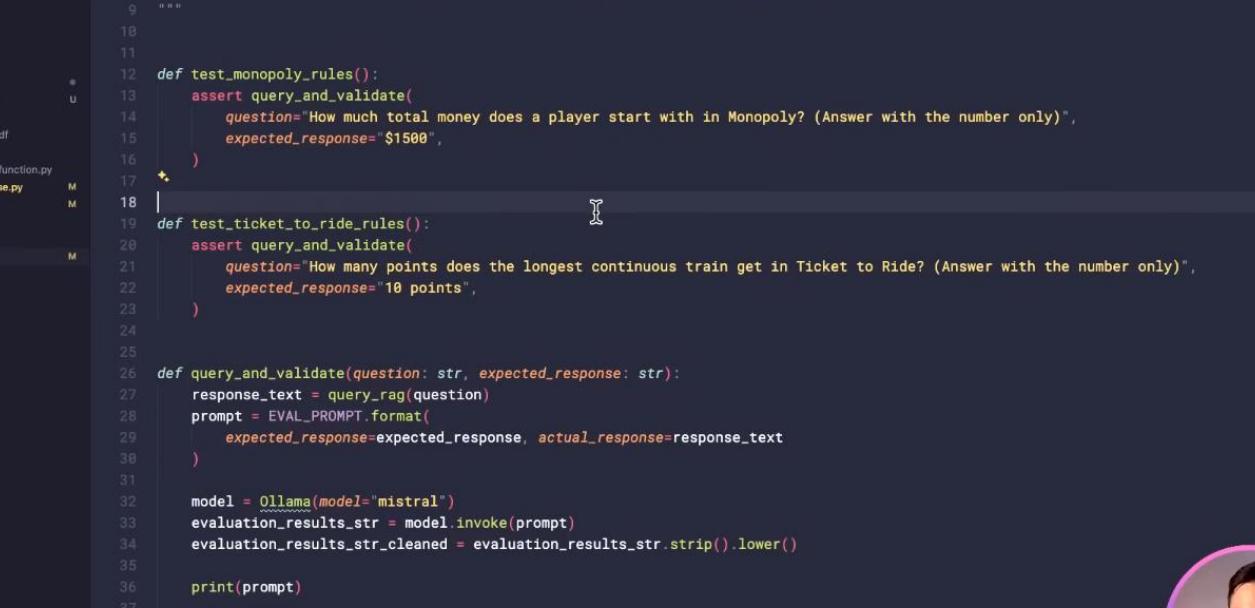
def test_ticket_to_ride_rules():
    assert query_and_validate(
        question="How many points does the longest continuous train get in Ticket to Ride? (Answer with the number only)",
        expected_response="10 points",
    )
```



```
query_data.py M  test_rag.py X
test_rag.py > test_ticket_to_ride_rules
3
4 EVAL_PROMPT = """
5 Expected Response: {expected_response}
6 Actual Response: {actual_response}
7 ---
8 (Answer with 'true' or 'false') Does the actual response match the expected response?
9 """
10
11 def test_monopoly_rules():
12     assert query_and_validate(
13         question="How much total money does a player start with in Monopoly? (Answer with the number only)",
14         expected_response="$1500",
15     )
16
17
18 def test_ticket_to_ride_rules():
19     assert query_and_validate(
20         question="How many points does the longest continuous train get in Ticket to Ride? (Answer with the number only)",
21         expected_response="10 points",
22     )
23
24
25
26 def query_and_validate(question: str, expected_response: str):
PROBLEMS 9  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
advanced-rag on / main [!?] via ✨ v3.10 [base] on px-beta-au:<ap-southeast-2>
→ []
```



```
query_data.py M  test_rag.py X
test_rag.py > ...
(ANSWER WITH true OR raise ) Does the actual response match the expected response?
9 """
10
11
12 def test_monopoly_rules():
13     assert query_and_validate(
14         question="How much total money does a player start with in Monopoly? (Answer with the number only)",
15         expected_response="$1500",
16     )
17
18 def test_ticket_to_ride_rules():
19     assert query_and_validate(
20         question="How many points does the longest continuous train get in Ticket to Ride? (Answer with the number only)",
21         expected_response="10 points",
22     )
23
24
25
26 def query_and_validate(question: str, expected_response: str):
27     response_text = query_rag(question)
28     prompt = EVAL_PROMPT.format(
29         expected_response=expected_response, actual_response=response_text
30     )
31
PROBLEMS 9  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
advanced-rag on / main [!?] via ✨ v3.10 [base] on px-beta-au:<ap-southeast-2>
→ []
```



```
advanced-rag
query_data.py M test_rag.py M

ANSWER WITH true OR raise ) Does the actual response match the expected response?

def test_monopoly_rules():
    assert query_and_validate(
        question="How much total money does a player start with in Monopoly? (Answer with the number only)",
        expected_response="$1500",
    )

def test_ticket_to_ride_rules():
    assert query_and_validate(
        question="How many points does the longest continuous train get in Ticket to Ride? (Answer with the number only)",
        expected_response="10 points",
    )

def query_and_validate(question: str, expected_response: str):
    response_text = query_rag(question)
    prompt = EVAL_PROMPT.format(
        expected_response=expected_response, actual_response=response_text
    )

    model = Ollama(model="mistral")
    evaluation_results_str = model.invoke(prompt)
    evaluation_results_str_cleaned = evaluation_results_str.strip().lower()

    print(prompt)

    if "true" in evaluation_results_str_cleaned:
        # Print response in Green if it is correct.
        print("\033[92m" + f"Response: {evaluation_results_str_cleaned}" + "\033[0m")
```



The screenshot shows a code editor interface with a dark theme. On the left is the Explorer sidebar, which lists project files: ADVANCED-RAG, __pycache__, .git, .pytest_cache, backup, chroma, data (containing codenames.pdf, monopoly.pdf, ticket_to_ride.pdf), ignore, get_embedding_function.py, populate_database.py, query_data.py, README.md, requirements.txt, and test_rag.py. The main area displays the content of test_rag.py. The code uses Python syntax with some color-coded elements like 'def' in blue and variable names in purple. It defines a function 'query_and_validate' that takes a question and an expected response, runs it through a model, and prints the result along with a color-coded status message ('Green' or 'Red') based on the comparison.

```
def query_and_validate(question: str, expected_response: str):
    response_text = query_rag(question)
    prompt = EVAL_PROMPT.format(
        expected_response=expected_response, actual_response=response_text
    )

    model = Ollama(model="mistral")
    evaluation_results_str = model.invoke(prompt)
    evaluation_results_str_cleaned = evaluation_results_str.strip().lower()

    print(prompt)

    if "true" in evaluation_results_str_cleaned:
        # Print response in Green if it is correct.
        print("\033[92m" + f"Response: {evaluation_results_str_cleaned}" + "\033[0m")
        return True
    elif "false" in evaluation_results_str_cleaned:
        # Print response in Red if it is incorrect.
        print("\033[91m" + f"Response: {evaluation_results_str_cleaned}" + "\033[0m")
        return False
    else:
        raise ValueError(
            f"Invalid evaluation result. Cannot determine if 'true' or 'false'."
        )
```



```
29     expected_response=expected_response, actual_response=response_text
30 )
31
PROBLEMS 9 OUTPUT DEBUG CONSOLE TERMINAL PORTS fish + - x ... ^ x
advanced-rag on ✘ main [!?] via v3.10 [base] on px-beta-su:<ap-southeast-2>
→ pytes -s
```



31

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

(Answer with 'true' or 'false') Does the actual response match the expected response?
Response: true. both the expected and actual responses state that the longest continuous train earns 10 points.
----- 2 passed in 23.57s -----
advanced-rag on ✓ main [!?] via ✨ v3.10 [base] on px-beta-au:<ap-southeast-2> took 24s
→ |

advanced-rag

EXPLORER

ADVANCED-RAG

- __pycache__
- .git
- .pytest_cache
- backup
- chroma
- data
 - codenames.pdf
 - monopoly.pdf
 - ticket_to_ride.pdf
- .gitignore
- get_embedding_function.py
- populate_database.py
- query_data.py
- README.md
- requirements.txt
- test_rag.py

query_data.py M test_rag.py X

test_rag.py ...

(ANSWER WITH TRUE OR FALSE) Does the actual response match the expected response?

g ==

10

Calculating Scores

Players should have already accounted for the points earned as they completed different routes. To make sure no mistakes were made, you may want to re-count the points for each player's routes.

Players should then reveal all their Destination Tickets and add (or subtract) the value of their Destination Tickets still in hand, based on whether they successfully (or not) connected those cities together.

The player who has the Longest Continuous Path of routes receives this special bonus card and adds 10 points to his score. When evaluating and

Σ

Σ f110 Train Car cards (12 each of Box, Passenger, Tanker, Reefer, Freight, Hopper, Coal, and Caboose cars, plus 14 Locomotives)

38 Destination Ticket cards! Summary

card

1 Longest Continuous Path

Bonus card

[T2R] rules EN reprint 2015_TTR2 rules US 06/03/15 17:36 Page3

Answer the question based on the above context: How many points does the longest continuous train get in Ticket to Ride? (Answer with the number only)

Response: The longest continuous train gets a bonus of 10 points.

Sources: ['data/ticket_to_ride.pdf:3:3', 'data/ticket_to_ride.pdf:3:1', 'data/ticket_to_ride.pdf:0:1', 'data/ticket_to_ride.pdf:3:2', 'data/ticket_to_ride.pdf:1:3']

Expected Response: 10 points

Actual Response: The longest continuous train gets a bonus of 10 points.

(Answer with 'true' or 'false') Does the actual response match the expected response?

Response: true. both the expected and actual responses state that the longest continuous train earns 10 points.

===== 2 passed in 23.57s =====

advanced-rag on main [!] via v3.10 [base] on px-beta- southeast-2 took 24s

advanced-rag

EXPLORER

ADVANCED-RAG

- __pycache__
- .git
- .pytest_cache
- backup
- chroma
- data
 - codenames.pdf
 - monopoly.pdf
 - ticket_to_ride.pdf
- .gitignore
- get_embedding_function.py
- populate_database.py
- query_data.py
- README.md
- requirements.txt
- test_rag.py

query_data.py M test_rag.py X

test_rag.py ...

(ANSWER WITH TRUE OR FALSE) Does the actual response match the expected response?

g ==

10

(toll free). Canadian consumers please write to: Hasbro Canada Corporation, 2350 de la Province, Longueuil, QC Canada, J4G 1G2.

The HASBRO, PARKER BROTHERS, and MONOPOLY names and logos, the distinctive design of the gameboard, the four corner squares, the MR. MONOPOLY name and character, and each of the distinctive elements of the board and rules are trademarks of Hasbro for its property trading game and game equipment. ©2004, 2007 Hasbro, Pawtucket, RI 02862.

All Rights Reserved. TM & © denote U.S. Trademarks.

00009-1

PROOF OF PURCHASE I

Answer the question based on the above context: How much total money does a player start with in Monopoly? (Answer with the number only)

Response: A player starts with \$1,500 in Monopoly.

Sources: ['data/monopoly.pdf:1:1', 'data/monopoly.pdf:1:8', 'data/monopoly.pdf:0:8', 'data/monopoly.pdf:2:8', 'data/monopoly.pdf:7:2']

Expected Response: \$1500

Actual Response: A player starts with \$1,500 in Monopoly.

(Answer with 'true' or 'false') Does the actual response match the expected response?

Response: true. the expected response was a value of \$1500 and the actual response stated that a player starts the game with \$1,500.

Human:

Answer the question based only on the following context:

comparing path lengths, only take into account continuous lines of plastic trains of the same color. A continuous path may include loops, and pass through the same city several times, but a given plastic train may never be used twice in the same continuous path. In the case of a tie for the longest path, all tied players score the 10 point bonus.

The player with the most points wins the game. If two or more players are tied for the most points, the player who has completed the most Destination Tickets wins. In the unlikely event that they are still tied, the player with the Longest Continuous Path card wins.

Route Length Points Scored

2

3

4

5

61

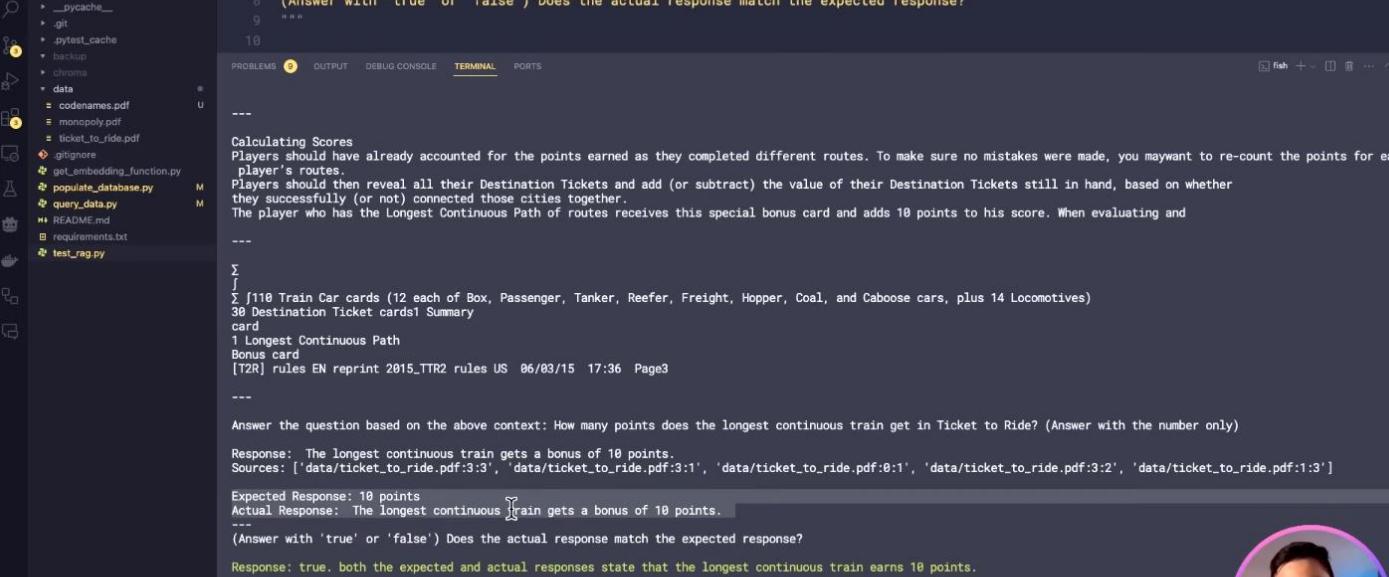
2

4

7

10





```
advanced-rag on ✘ main [!?] via ✘ v3.10 [base] on px-beta-au:<ap-southeast-2> took 24s
→ |
```

The screenshot shows a terminal window with the following output:

```
=====
  2 passed in 23.57s =====
```

Below the terminal window, there is a circular video player overlay showing a man's face.

This works! But it is also good to do a negative test case to check that the LLM responses are indeed correct



A screenshot of a Microsoft Visual Studio Code interface showing Python code for testing a large language model's responses. The code includes functions for testing monopoly rules, ticket-to-ride rules, and a general query-validation function. It uses the Ollama library to invoke the model and evaluate its responses against expected values.

```
advanced-rag

query_data.py M test_rag.py M advanced-rag

explorer advanced-bag
  - __pycache__
  - .git
  - .pytest_cache
  - backup
  - chroma
  - data
    - codenames.pdf
    - monopoly.pdf
    - ticket_to_ride.pdf
    - ignore
  - get_embedding_function.py
  - populate_database.py M
  - query_data.py M
  - README.md
  - requirements.txt
  - test_rag.py M

query_and_validate(question: str, expected_response: str):
    response_text = query_rag(question)
    prompt = EVAL_PROMPT.format(
        expected_response=expected_response, actual_response=response_text
    )

    model = Ollama(model="mistral")
    evaluation_results_str = model.invoke(prompt)
    evaluation_results_str_cleaned = evaluation_results_str.strip().lower()

    print(prompt)

    if "true" in evaluation_results_str_cleaned:
        # Print response in Green if it is correct.
        print("\u001b[92m" + f"Response: {evaluation_results_str_cleaned}" + "\u001b[0m")
```



Screenshot of a code editor showing a Python test file named `test_rag.py`. The code contains several test functions for a game called Monopoly. One specific test function, `test_monopoly_rules()`, is highlighted. It asserts that the expected response is "9999" and the actual response is "1500". A tooltip or comment above the assertion reads: "(Answer with 'true' or 'false') Does the actual response match the expected response?" The code uses the `Ollama` library to invoke a model named "mistral".

```

5 Expected Response: {expected_response}
6 Actual Response: {actual_response}
7 ---
8 (Answer with 'true' or 'false') Does the actual response match the expected response?
9 """
10
11 def test_monopoly_rules():
12     assert query_and_validate(
13         question="How much total money does a player start with in Monopoly? (Answer with the number only)",
14         expected_response="$9999",
15     )
16
17
18 def test_ticket_to_ride_rules():
19     assert query_and_validate(
20         question="How many points does the longest continuous train get in Ticket to Ride? (Answer with the number only)",
21         expected_response="10 points",
22     )
23
24
25
26 def query_and_validate(question: str, expected_response: str):
27     response_text = query_rag(question)
28     prompt = EVAL_PROMPT.format(
29         expected_response=expected_response, actual_response=response_text
30     )
31
32     model = Ollama(model="mistral")
33     evaluation_results_str = model.invoke(prompt)
34     evaluation_results_str_cleaned = evaluation_results_str.strip().lower()
35
36     print(prompt)

```

We change the 1500 to 999 so that the test should now fail.



Screenshot of a code editor showing the same `test_rag.py` file after the modification. The test function `test_monopoly_rules()` now fails because the expected response is "9999" and the actual response is "999". The terminal tab shows the output of the test run, which includes the context provided by the game's rules and the user's input. The test fails with a message indicating that the actual response is a description of the game situation rather than a numerical value.

```

lend money to another player.

We will be happy to hear your questions or comments about this game. Write to: Hasbro
Games, Consumer Affairs Dept., P.O. Box 200, Pawtucket, RI 02862. Tel: 888-836-7025
(toll free). Canadian consumers please write to: Hasbro Canada Corporation, 2350 de la
Province, Longueuil, QC Canada, J4G 1G2.
The HASBRO, PARKER BROTHERS, and MONOPOLY names and logos, the distinctive design of
the gameboard, the four corner squares, the MR. MONOPOLY name and character, and each of the
distinctive elements of the board and rules are trademarks of Hasbro for its property trading game
and game equipment. ©2004, 2007 Hasbro, Pawtucket, RI 02862.
All Rights Reserved. TM & © denote U.S. Trademarks.
00009-1
PROOF OF PURCHASE I

Answer the question based on the above context: How much total money does a player start with in Monopoly? (Answer with the number only)

Response: A player starts with $1,500 in Monopoly.
Sources: ['data/monopoly.pdf:1:1', 'data/monopoly.pdf:1:0', 'data/monopoly.pdf:0:0', 'data/monopoly.pdf:2:0', 'data/monopoly.pdf:7:2']

Expected Response: $9999
Actual Response: A player starts with $1,500 in Monopoly.
---
(Answer with 'true' or 'false') Does the actual response match the expected response?

Response: false. the actual response is a description of a monopoly game situation, not a numerical value that matches the expected response of $9999.
FHuman:
Answer the question based only on the following context:

comparing path lengths, only take into account continuous lines of plastic trains of the same color. A continuous path may include loops, and passthrough the same city several
times, but a given plastic train may never be used twice in the same continuous path. In the case of a tie for the longest path, all tied players score the same number of points.
The player with the most points wins the game. If two or more players are tied for the most points, the player who has completed the most Destination
Tickets wins. In the unlikely event that they are still tied, the player with the Longest Continuous Path card wins.
Route Length Points Scored
2
3
4
5
61

```



advanced-rag

query_data.py M test_rag.py M

test_rag.py > test_monopoly_rules

```

5 Expected Response: {expected_response}
6 Actual Response: {actual_response}
7 ---
8 Σ [118 Train Car cards (12 each of Box, Passenger, Tanker, Reefer, Freight, Hopper, Coal, and Caboose cars, plus 14 Locomotives)
9 38 Destination Ticket cards1 Summary
10 card
11 Longest Continuous Path
12 Bonus card
13 [T2R] rules EN reprint 2015.TTR2 rules US 06/03/15 17:36 Page3
14 ---
15 Answer the question based on the above context: How many points does the longest continuous train get in Ticket to Ride? (Answer with the number only)
16 Response: The longest continuous train gets a bonus of 10 points.
17 Sources: ['data/ticket_to_ride.pdf:3:3', 'data/ticket_to_ride.pdf:3:1', 'data/ticket_to_ride.pdf:0:1', 'data/ticket_to_ride.pdf:3:2', 'data/ticket_to_ride.pdf:1:3']
18
19 Expected Response: 10 points
20 Actual Response: The longest continuous train gets a bonus of 10 points.
21
22 (Answer with 'true' or 'false') Does the actual response match the expected response?
23
24 Response: true. both responses award 10 points for the longest continuous train.
25
26
27 ===== FAILURES =====
28 test_monopoly_rules
29
30     def test_monopoly_rules():
31         assert query_and_validate(
32             question="How much total money does a player start with in Monopoly? (Answer with the number only)",
33             expected_response="$9999",
34         )
35     E   AssertionError: assert False
36     E   + where False = query_and_validate(question='How much total money does a player start with in Monopoly? (Answer with the number only)', expected_response='$9999')
37
38 test_rag.py:13: AssertionError
39 ===== short test summary info =====
40 FAILED test_rag.py::test_monopoly_rules - AssertionError: assert False
41 ===== 1 failed, 1 passed in 19.18s =====
42
43 advanced-rag on py main [!] via v3.10 [base] on px-beta-ap-southeast-2> took 28s
44

```

Ln 15 Col 35 Spaces:4 UTF-8 LF Python 3.10.8 ('base':conda) editor

We can include this negative test in our test suite output by inverting the test assertion as below



advanced-rag

query_data.py M test_rag.py M

test_rag.py > test_monopoly_rules

```

5 Expected Response: {expected_response}
6 Actual Response: {actual_response}
7 ---
8 (Answer with 'true' or 'false') Does the actual response match the expected response?
9
10
11
12 def test_monopoly_rules():
13     assert not query_and_validate(
14         question="How much total money does a player start with in Monopoly? (Answer with the number only)",
15         expected_response="$9999",
16     )
17
18
19 def test_ticket_to_ride_rules():
20     assert query_and_validate(
21         question="How many points does the longest continuous train get in Ticket to Ride? (Answer with the number only)",
22         expected_response="10 points",
23     )
24
25
26 def query_and_validate(question: str, expected_response: str):
27     response_text = query_rag(question)
28     prompt = EVAL_PROMPT.format(
29         expected_response=expected_response, actual_response=response_text
30     )
31
32     model = Ollama(model="mistral")
33     evaluation_results_str = model.invoke(prompt)
34     evaluation_results_str_cleaned = evaluation_results_str.strip().lower()
35
36     print(prompt)
37

```

Ln 13 Col 4 (181 selected) Spaces:4 UTF-8 LF Python 3.10.8 ('base':conda) editor



advanced-rag

query_data.py M test_rag.py M

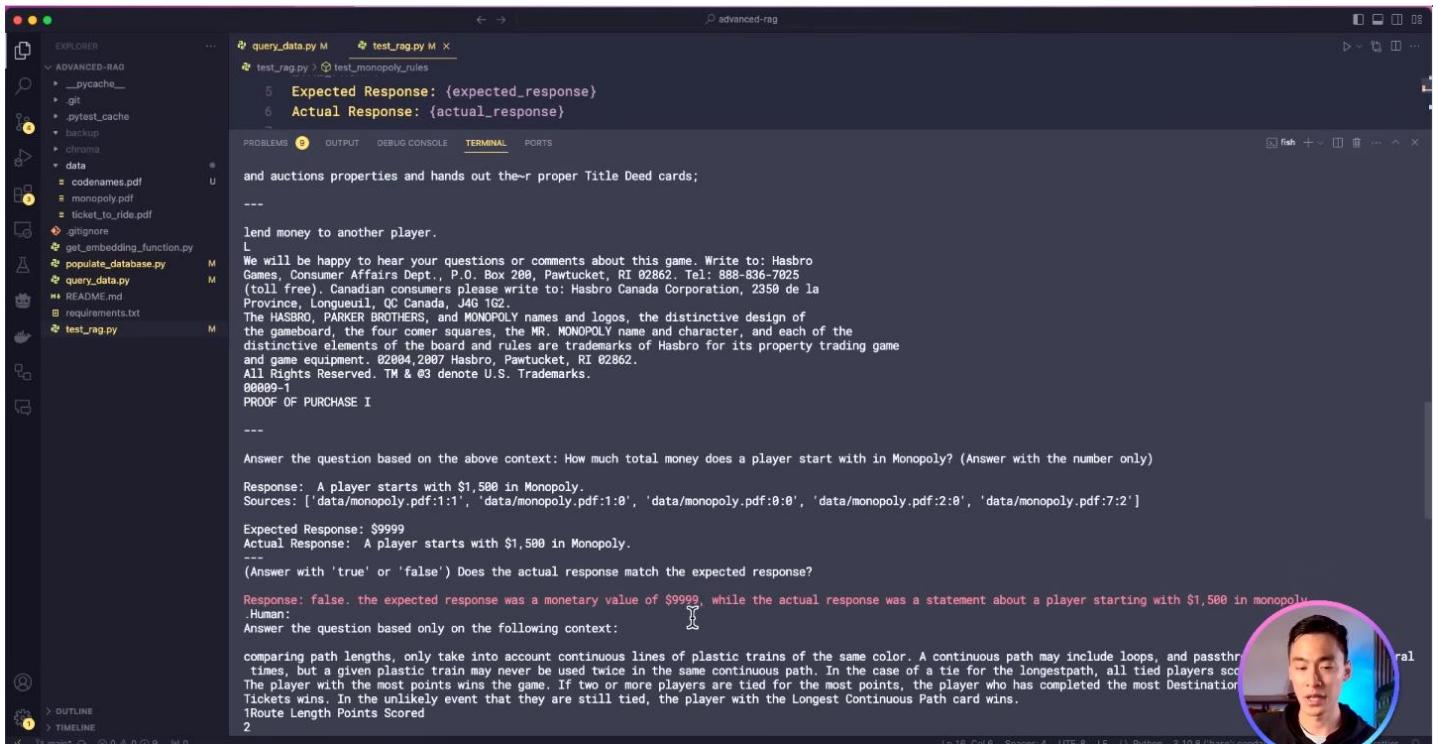
test_rag.py > test_monopoly_rules

```

5 Expected Response: {expected_response}
6 Actual Response: {actual_response}
7 ---
8
9 advanced-rag on py main [!] via v3.10 [base] on px-beta-ap-southeast-2>
10 > pytest -s

```

Ln 13 Col 4 (181 selected) Spaces:4 UTF-8 LF Python 3.10.8 ('base':conda) editor



```

query_data.py M  test_rag.py M
test_rag.py > test_monopoly_rules
5 Expected Response: {expected_response}
6 Actual Response: {actual_response}

and auctions properties and hands out their proper Title Deed cards;
---
lend money to another player.

We will be happy to hear your questions or comments about this game. Write to: Hasbro Games, Consumer Affairs Dept., P.O. Box 280, Pawtucket, RI 02862. Tel: 888-836-7025 (toll free). Canadian consumers please write to: Hasbro Canada Corporation, 2350 de la Province, Longueuil, QC Canada, J4G 1G2. The HASBRO, PARKER BROTHERS, and MONOPOLY names and logos, the distinctive design of the gameboard, the four corner squares, the MR. MONOPOLY name and character, and each of the distinctive elements of the board and rules are trademarks of Hasbro for its property trading game and game equipment. 02884, 2807 Hasbro, Pawtucket, RI 02862. All Rights Reserved. TM & © denote U.S. Trademarks.
00000-1
PROOF OF PURCHASE I

Answer the question based on the above context: How much total money does a player start with in Monopoly? (Answer with the number only)

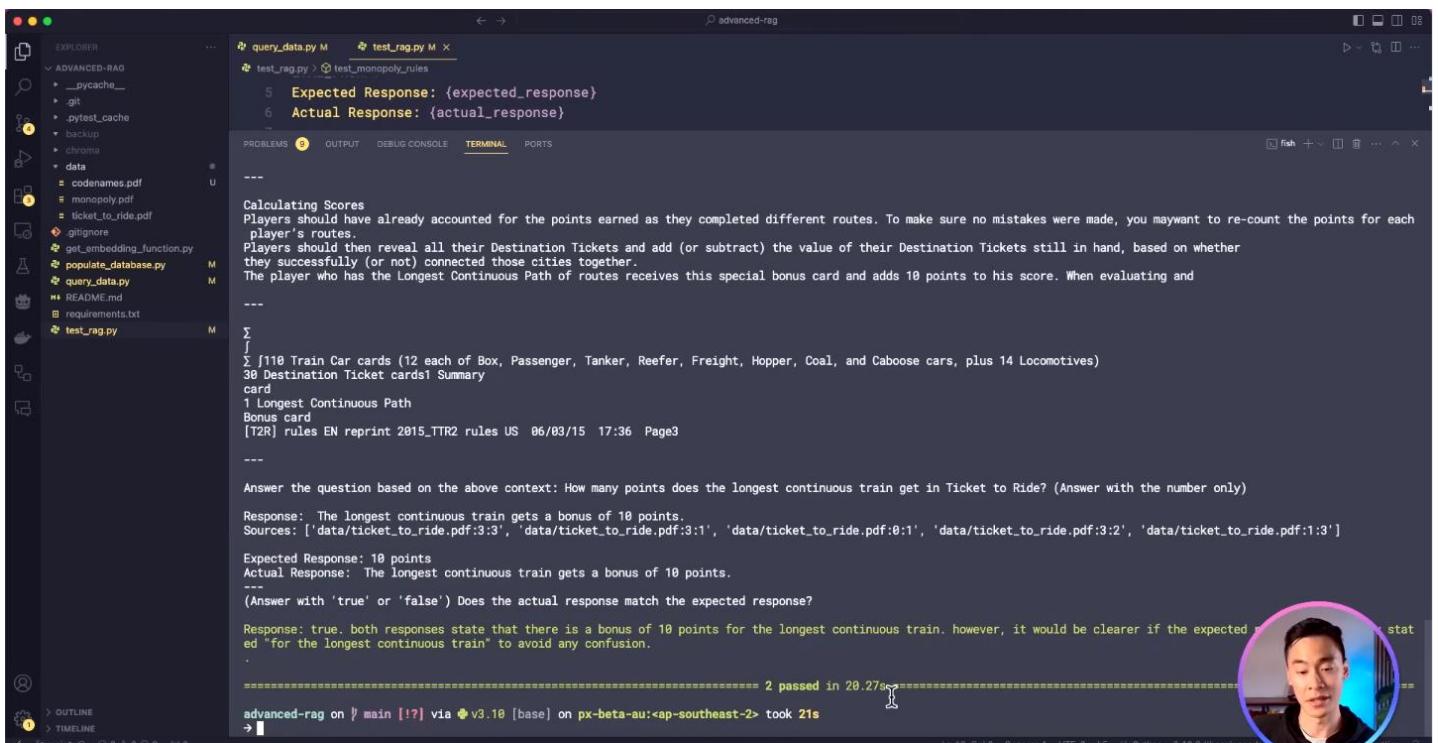
Response: A player starts with $1,500 in Monopoly.
Sources: ['data/monopoly.pdf:1:1', 'data/monopoly.pdf:1:0', 'data/monopoly.pdf:0:0', 'data/monopoly.pdf:2:0', 'data/monopoly.pdf:7:2']

Expected Response: $9999
Actual Response: A player starts with $1,500 in Monopoly.

(Answer with 'true' or 'false') Does the actual response match the expected response?

Response: false. the expected response was a monetary value of $9999, while the actual response was a statement about a player starting with $1,500 in monopoly.
Human:
Answer the question based only on the following context:

comparing path lengths, only take into account continuous lines of plastic trains of the same color. A continuous path may include loops, and passes through cities multiple times, but a given plastic train may never be used twice in the same continuous path. In the case of a tie for the longest path, all tied players score destination tickets. The player with the most points wins the game. If two or more players are tied for the most points, the player who has completed the most Destination Tickets wins. In the unlikely event that they are still tied, the player with the Longest Continuous Path card wins.
2
Route Length Points Scored
```



```

query_data.py M  test_rag.py M
test_rag.py > test_monopoly_rules
5 Expected Response: {expected_response}
6 Actual Response: {actual_response}

Calculating Scores
Players should have already accounted for the points earned as they completed different routes. To make sure no mistakes were made, you may want to re-count the points for each player's routes.
Players should then reveal all their Destination Tickets and add (or subtract) the value of their Destination Tickets still in hand, based on whether they successfully (or not) connected those cities together.
The player who has the Longest Continuous Path of routes receives this special bonus card and adds 10 points to his score. When evaluating and

Answer the question based on the above context: How many points does the longest continuous train get in Ticket to Ride? (Answer with the number only)

Response: The longest continuous train gets a bonus of 10 points.
Sources: ['data/ticket_to_ride.pdf:3:3', 'data/ticket_to_ride.pdf:3:1', 'data/ticket_to_ride.pdf:0:1', 'data/ticket_to_ride.pdf:3:2', 'data/ticket_to_ride.pdf:1:3']

Expected Response: 10 points
Actual Response: The longest continuous train gets a bonus of 10 points.
---
(Answer with 'true' or 'false') Does the actual response match the expected response?

Response: true. both responses state that there is a bonus of 10 points for the longest continuous train. however, it would be clearer if the expected response ed "for the longest continuous train" to avoid any confusion.
-
=====
2 passed in 20.27s
advanced-rag on ✓ main [!?] via ✘ v3.10 [base] on px-beta-

```

Our entire test suite is now passing as expected and we have both positive and negative test cases documented. We can set a test pass threshold of 90% to have our LLM still good enough for use.

github.com/pixegami/rag-tutorial-v2/blob/main/populate_database.py



Files

main
Go to file t
data
.gitignore
README.md
get_embedding_function.py
populate_database.py
query_data.py
requirements.txt
test_rag.py

rag-tutorial-v2 / populate_database.py

Code Blame 110 lines (81 loc) · 3.06 KB

```
5  from langchain.text_splitters import RecursiveCharacterTextSplitter
6  from langchain.schema.document import Document
7  from get_embedding_function import get_embedding_function
8  from langchain.vectorstores.chroma import Chroma
9
10 CHROMA_PATH = "chroma"
11 DATA_PATH = "data"
12
13
14 def main():
15
16     # Check if the database should be cleared (using the --clear flag).
17     parser = argparse.ArgumentParser()
18     parser.add_argument("--reset", action="store_true", help="Reset the database.")
19     args = parser.parse_args()
20
21     if args.reset:
22         print(" - Clearing Database")
23         clear_database()
24
25     # Create (or update) the data store.
26     documents = load_documents()
27     chunks = split_documents(documents)
28     add_to_chroma(chunks)
29
30
31 def load_documents():
32     document_loader = PyPDFDirectoryLoader(DATA_PATH)
33     return document_loader.load()
34
35
36 def split_documents(documents: list[Document]):
37     text_splitter = RecursiveCharacterTextSplitter(
38         chunk_size=800,
39         chunk_overlap=80,
40         length_function=len,
41         is_separator_regex=False,
42     )
43     return text_splitter.split_documents(documents)
44
```

Raw

Top