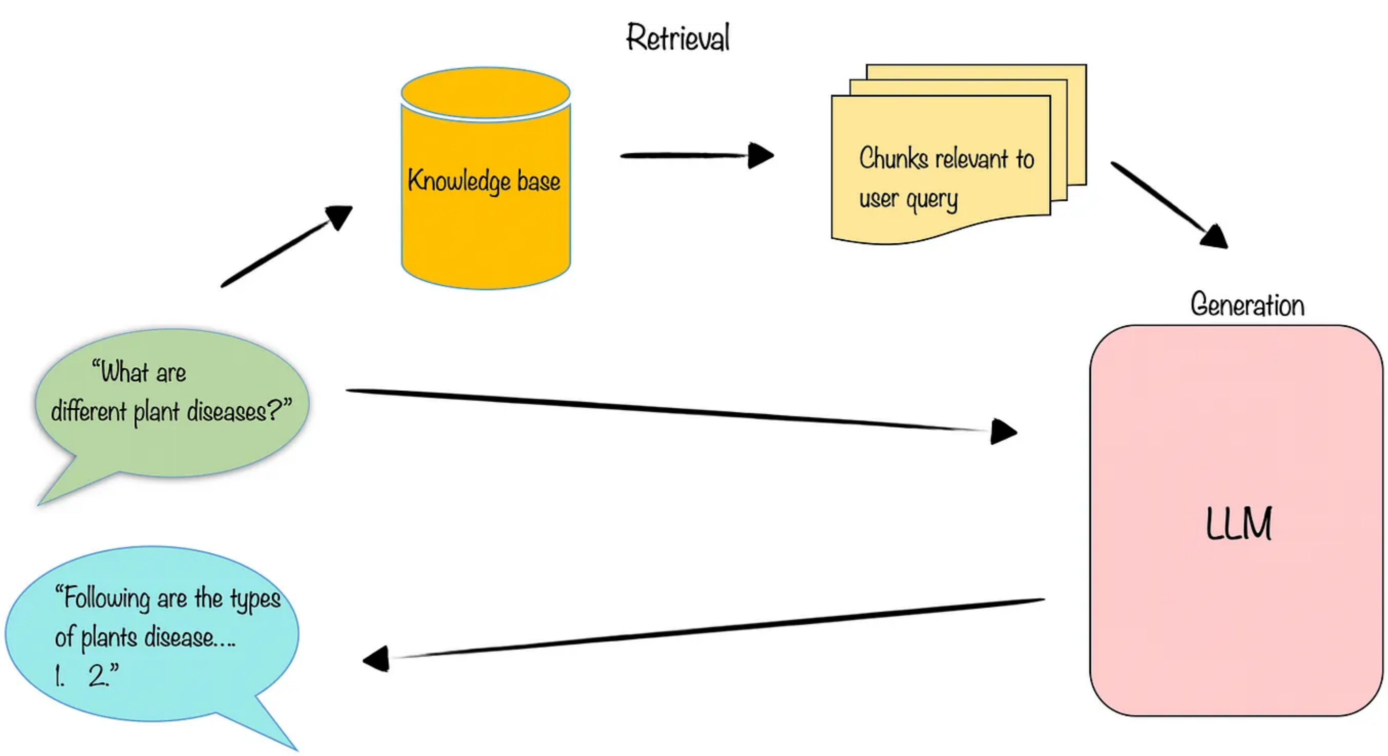
Implementing RAG using Langchain and Ollama

Abhishek Selokar · Follow

14 min read · Apr 10, 2024

-- 1

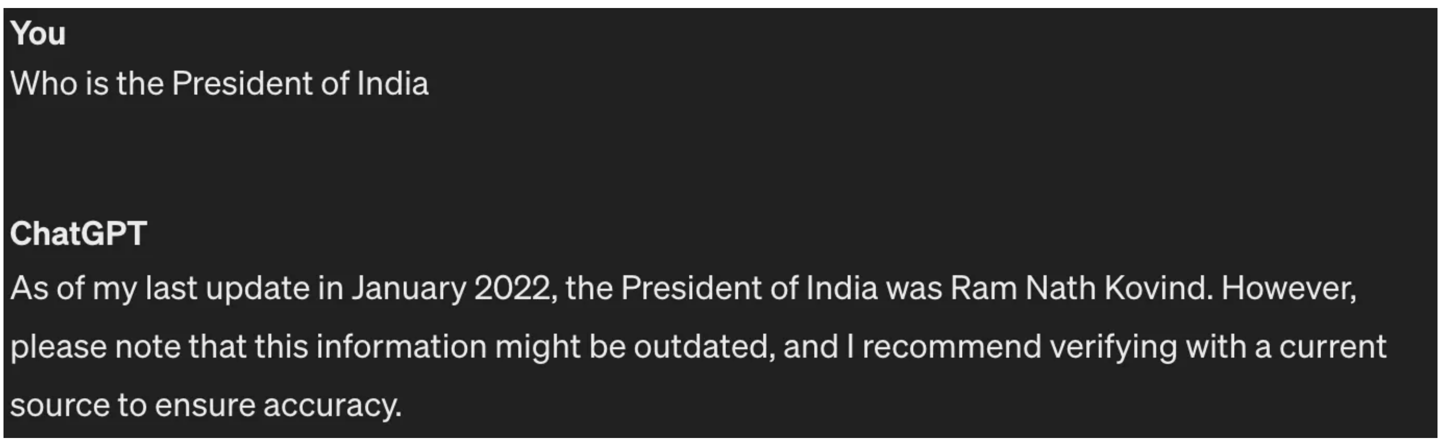
Let's use one of the most famous techniques to ground the LLM and guide the LLM to respond with more accurate information.



LLMs are great at understanding language and carving out the context from the piece of the text. Despite being so powerful, it too faces some problems that may lead to unreliability for some use cases where information from the model needs to be precise and up to date.

Problems

1. **Hallucination:** They tend to hallucinate very confidently, which may lead to misinformation
2. **Limited by Training Data:** They know nothing outside of their training data.


You
Who is the President of India

ChatGPT
As of my last update in January 2022, the President of India was Ram Nath Kovind. However, please note that this information might be outdated, and I recommend verifying with a current source to ensure accuracy.

3. **Black Box Outputs:** One cannot confidently find out what has led to the generation of particular content.

RAG at your service, sir !!!!

It is an AI framework that helps ground LLM with external sources. It's useful to answer questions or generate content by leveraging external knowledge.

There are two main steps in RAG:

1. Retrieval: Retrieving the most relevant information from a knowledge base with text embeddings stored in a vector store with respect to the user query
2. Generation: Using the retrieved information as input to the LLM to generate content based on the given query and provided context.

Throughout the blog, I will be using ***Langchain,*** which is a framework designed to simplify the creation of applications using large language models, and ***Ollama,*** which provides a simple API for creating, running, and managing models.

Make sure to install the following dependencies

```
pip install langchain==0.1.14
pip install langchain-experimental==0.0.56
pip install langchain-community==0.0.31
pip install faiss-cpu==1.8.0
pip install pdfplumber==0.11.0
pip install gradio==4.25.0
```

1. Load Data

For the demo, I am using a PDF consisting of information regarding plant diseases. You can use any other relevant pdf.

```
from langchain_community.document_loaders import PDFPlumberLoader
loader = PDFPlumberLoader("11pests1disease.pdf")
docs = loader.load()

# Check the number of pages
print("Number of pages in the PDF:",len(docs))

# Load the random page content
docs[2].page_content

# Output
"""
Leaf Diseases Caused By Fungi and Bacteria
Leaf Spots
Bacteria or fungi can cause leaf spots that vary in size, shape, and color. Usu
halo. A fungal leaf spot nearly always has a growth of some type in the spot, p
moldy growth of spores. Often the structures are visible through a hand lens. N
photo: Paul Bachi, University of Kentucky Research and Education Center, Bugwoo
Septoria brown spot is a common fungal disease of soybeans. It causes small ang
trifoliate leaves 2 to 3 weeks after planting. Numerous spots will cause leaves
on trifoliate leaves that later turn dark brown. Individual spots will frequent
Defoliation from the bottom to the top of severely diseased trifoliate leaves i
annually in almost every field in Kentucky. Late-season brown spot is much more
The fungus survives from season to season in crop debris and seed. Warm, moist
spread by wind and rain. Hot, dry weather can stop disease development.
Leaf Blights
Leaf blights generally affect larger leaf areas and are more irregular than lea
photo: Margaret McGrath, Cornell University, Bugwood.org
Northern corn leaf blight (NCLB), caused by a fungus, first develops on the low
telltale sign of northern corn leaf blight is the 1-to-6 inch long cigar-shaped
develops, the lesions spread to all leafy structures.
Wet weather and moderate temperatures favor NCLB. Symptoms can be confused with
"""
```

2. Split the document into chunks

As the context window of the LLM is limited, it's not possible to feed the whole content into the LLM at once. Even models with large window sizes can struggle to find information in very long inputs and can perform very badly. So we chunk it into pieces, create embeddings of each chunk, and store it. It helps to retrieve only the relevant information from the corpus and use that as a context for LLM to generate a response.

Here I'm using *SemanticChunker* to split the text based on semantic similarity. There are other functions too. One such is *RecursiveCharacterTextSplitter* which will recursively split the document using common separators `["\n\n", "\n", " ", ""]` until each chunk is the appropriate size. This helps to keep all the paragraphs, then sentences, and then words together, because it makes sense that each of those will be semantically related if kept together.

```
from langchain_experimental.text_splitter import SemanticChunker
from langchain.embeddings import HuggingFaceEmbeddings

text_splitter = SemanticChunker(HuggingFaceEmbeddings())
documents = text_splitter.split_documents(docs)
```

```
# Check number of chunks created
print("Number of chunks created: ", len(documents))
# Output
"""
Number of chunks created:  23
"""

# Printing first few chunks
for i in range(len(documents)):
    print()
    print(f"CHUNK : {i+1}")
    print(documents[i].page_content)
# Output
"""

CHUNK : 1
Kentucky Pesticide Education Program
copyright © 2016 University of Kentucky Department of Entomology
Agricultural Plant Diseases
```

Plant Diseases
A plant disease is any harmful condition that affects a plant’s appearance or f
bacteria, and viruses. Some nematodes are plant disease agents. Temperature ext
non-infectious factors. The disease triangle is a fundamental concept in plant
host, apathogen (the agent that causes disease), and an environment favorable f
the disease triangle: the plant, the pathogen, and/or the environment. Infectio
environmental conditions are favorable, the pathogen begins to develop. The pla
1. overdevelopment of tissue - galls, swellings, or leaf curls;
2.

CHUNK : 2
underdevelopment of tissue - stunting, lack of chlorophyll, or incomplete devel
3. tissue death - blight, leaf spot, wilting, and cankers. Plant disease pathog
  by wind;
  rain;
  animals;
  soil;
  nursery grafts;
  vegetative propagation;
  contaminated equipment and tools;
  infected seed stock;
  pollen;
  dust storms;
  irrigation water; and
  people.

CHUNK : 3
Infectious Organisms that Cause Diseases
Fungi are the most common infectious organisms causing plant disease. They do n

CHUNK : 4
They must
get it by living on another organism. Most fungi are beneficial. They contribut
parasites on living plants. They may attack plants and plant products both abov
species; others have to only one host species. Most fungi reproduce by spores,
tremendous numbers. Often spores can survive for weeks, months, or even years w
always needed for spore germination and active fungal growth. Spores can spread
equipment. Fungal infections frequently are identified by the vegetative body o
they are large enough to see. Symptoms of fungal infections include
  soft rot of fruits,
  plant stunting,
  smuts,
  rusts,
  leaf spots,
  wilting, or
  thickening and curling of leaves. Powdery and downy mildew, smut, root and st

CHUNK : 5

```
They can build up
quickly under warm, humid weather conditions. Leaf, growing shoots, and fruit d
seasons in crop residue, in seeds or cuttings, or in weeds.

"""
```

3. Create embeddings for each text chunk

For each text chunk, we create text embeddings, which means we find the numerical representations of those text chunks. I'm using the open-source embedding model *HuggingFaceEmbeddings* to create embeddings and store those in a vector database called *FAISS*, which allows for efficient similarity search. You can use any database of your choice.

```
from langchain_community.embeddings import HuggingFaceEmbeddings
from langchain_community.vectorstores import FAISS

# Instantiate the embedding model
embedder = HuggingFaceEmbeddings()

# Create the vector store
vector = FAISS.from_documents(documents, embedder)
```

4. Retrieval from the vector database

This is a part where we can find the most semantically similar text chunks related to the user query from the vector store. Any `VectorStore` can easily be turned into a `Retriever` with `VectorStore.as_retriever()`.

Let’s find the top 3 most similar paragraphs based on the provided query.

```
# Input
retriever = vector.as_retriever(search_type="similarity", search_kwargs={"k": 3
retrieved_docs = retriever.invoke("How does plant respond to disease?")

# Output
"""
Doc 1 content:
 Kentucky Pesticide Education Program
copyright © 2016 University of Kentucky Department of Entomology
Agricultural Plant Diseases
Plant Diseases
A plant disease is any harmful condition that affects a plant’s appearance or f
bacteria, and viruses. Some nematodes are plant disease agents. Temperature ext
non-infectious factors. The disease triangle is a fundamental concept in plant 
host, apathogen (the agent that causes disease), and an environment favorable f
the disease triangle: the plant, the pathogen, and/or the environment. Infectio
environmental conditions are favorable, the pathogen begins to develop. The pla
1. overdevelopment of tissue - galls, swellings, or leaf curls;
2.

Doc 2 content:
 underdevelopment of tissue - stunting, lack of chlorophyll, or incomplete deve
3. tissue death - blight, leaf spot, wilting, and cankers. Plant disease pathog
  by wind;
  rain;
  animals;
  soil;
  nursery grafts;
  vegetative propagation;
  contaminated equipment and tools;
  infected seed stock;
  pollen;
  dust storms;
  irrigation water; and
  people.

Doc 3 content:
   Reproduction occurs on resistant soybeans.   Moves every way that soil moves
A correct diagnosis is the first step in disease management. To recognize a dis
you are trying to identify the cause of a plant disease, you need to look for s
presence of the disease agent. Many different plant diseases cause similar symp
or stunted growth. For example, similar symptoms may be a result of mechanical 
the only way to pinpoint the cause is to find the observable signs that the par
bacterial ooze.
"""
```

5. Generation

As soon as we get the most semantically similar text chunk related to the user query from the vector store, it's time to feed both of them (*retrieved text chunks and user query*) to the LLM as input to provide more context for an accurate response.

I have used Ollama to use the LLM model locally. To set it up in your system, check out this link. I'm using "mistral" for this demo. You can experiment with a model of your choice.

```
from langchain_community.llms import Ollama

# Define llm
llm = Ollama(model="mistral")
```

We first load the LLM model and then set up a custom prompt. Prompt templates are predefined recipes for generating prompts, which may include instructions on how LLM should respond, few-shot examples, specific context, and questions for language models.

An LLMChain is a simple chain that adds some functionality to language models.

StuffDocumentChain takes a list of documents, inserts them all into a prompt, and passes that prompt to an LLM.

```
from langchain.chains import RetrievalQA
from langchain.chains.llm import LLMChain
from langchain.chains.combine_documents.stuff import StuffDocumentsChain
from langchain.prompts import PromptTemplate

prompt = """
1. Use the following pieces of context to answer the question at the end.
2. If you don't know the answer, just say that "I don't know" but don't make up
3. Keep the answer crisp and limited to 3,4 sentences.

Context: {context}

Question: {question}

Helpful Answer:"""


QA_CHAIN_PROMPT = PromptTemplate.from_template(prompt)

llm_chain = LLMChain(
                  llm=llm,
                  prompt=QA_CHAIN_PROMPT,
                  callbacks=None,
                  verbose=True)

document_prompt = PromptTemplate(
    input_variables=["page_content", "source"],
    template="Context:\ncontent:{page_content}\nsource:{source}",
)

combine_documents_chain = StuffDocumentsChain(
                  llm_chain=llm_chain,
                  document_variable_name="context",
                  document_prompt=document_prompt,
                  callbacks=None,
              )

qa = RetrievalQA(
                  combine_documents_chain=combine_documents_chain,
                  verbose=True,
                  retriever=retriever,
                  return_source_documents=True,
              )
```

```
# Input
print(qa("How does plant respond to disease?")["result"])


# Output
"""
> Entering new LLMChain chain...
Prompt after formatting:
Use the following pieces of context delimmited by <> to answer the question at
If you don't know the answer, just say that you don't know, don't make up answe
Keep the answer crisp and not greater than 3 sentences.

Context: <Context:
content:Kentucky Pesticide Education Program
copyright © 2016 University of Kentucky Department of Entomology
Agricultural Plant Diseases
Plant Diseases
A plant disease is any harmful condition that affects a plant’s appearance or f
bacteria, and viruses. Some nematodes are plant disease agents. Temperature ext
non-infectious factors. The disease triangle is a fundamental concept in plant
host, apathogen (the agent that causes disease), and an environment favorable f
the disease triangle: the plant, the pathogen, and/or the environment. Infectio
environmental conditions are favorable, the pathogen begins to develop. The pla
1. overdevelopment of tissue - galls, swellings, or leaf curls;
2.
source:/Users/abhi/11pests1disease.pdf

Context:
content:underdevelopment of tissue - stunting, lack of chlorophyll, or incomple
3. tissue death - blight, leaf spot, wilting, and cankers. Plant disease pathog
  by wind;
  rain;
  animals;
  soil;
  nursery grafts;
  vegetative propagation;
  contaminated equipment and tools;
  infected seed stock;
  pollen;
  dust storms;
  irrigation water; and
  people.
source:/Users/abhi/11pests1disease.pdf

Context:
```

```
content:  Reproduction occurs on resistant soybeans.   Moves every way that soi
A correct diagnosis is the first step in disease management. To recognize a dis
you are trying to identify the cause of a plant disease, you need to look for s
presence of the disease agent. Many different plant diseases cause similar symp
or stunted growth. For example, similar symptoms may be a result of mechanical
the only way to pinpoint the cause is to find the observable signs that the par
bacterial ooze.
source:/Users/abhi/11pests1disease.pdf

Question: How does plant respond to disease?
Helpful Answer:

> Finished chain.

> Finished chain.

Plants respond to diseases in three main ways:
(1) overdevelopment of tissue, such as galls, swellings, or leaf curls;
(2) underdevelopment of tissue, including stunting, lack of chlorophyll, or inc
(3) tissue death, which can manifest as blight, leaf spot, wilting, and cankers
"""
```

As you can see above, LLM used the retrieved information for the vector store and then used that as a. context to provide an accurate answer.

Now let’s see how RAG has overcome all the problems which we discussed earlier

Problem 1: Hallucinations

Solution:

As we are providing some context to the LLM related to the user query, it is prone to generate answers based on that rather than guessing and generating some absurd answers

Problem 2: Knowledge cut off

Solution:

PDF or the external knowledge base can be updated at any time based on the requirement. Information can be added, deleted, and modified. This will help ground the LLM with up-to-date knowledge.

Problem 3: No Interpretability

Solution:

Based on the user query, most similar text chunks are retrieved from the database and are used as context. So now, as we know the source {retrieved chunks} based on which LLM has produced the output, we can easily trace back to that particular chunk to know why it said, what it said.

Isn't RAG great?? Definitely it is.

RadioBot: Your Interactive Chat Companion

Let's wrap up this by putting all these things together and making a small chatbot-based interface using ***Gradio***.

```
from langchain_community.document_loaders import PDFPlumberLoader
from langchain_experimental.text_splitter import SemanticChunker
from langchain_community.embeddings import HuggingFaceEmbeddings
from langchain_community.vectorstores import FAISS
from langchain_community.llms import Ollama
from langchain.prompts import PromptTemplate
from langchain.chains.llm import LLMChain
from langchain.chains.combine_documents.stuff import StuffDocumentsChain
from langchain.chains import RetrievalQA
import gradio as gr
```

```
# Load the PDF
loader = PDFPlumberLoader("11pests1disease.pdf")
docs = loader.load()

# Split into chunks
text_splitter = SemanticChunker(HuggingFaceEmbeddings())
documents = text_splitter.split_documents(docs)


# Instantiate the embedding model
embedder = HuggingFaceEmbeddings()

# Create the vector store and fill it with embeddings
vector = FAISS.from_documents(documents, embedder)
retriever = vector.as_retriever(search_type="similarity", search_kwargs={"k": 3

# Define llm
llm = Ollama(model="mistral")

# Define the prompt
prompt = """
1. Use the following pieces of context to answer the question at the end.
2. If you don't know the answer, just say that "I don't know" but don't make up
3. Keep the answer crisp and limited to 3,4 sentences.

Context: {context}

Question: {question}

Helpful Answer:"""

QA_CHAIN_PROMPT = PromptTemplate.from_template(prompt)

llm_chain = LLMChain(
                  llm=llm,
                  prompt=QA_CHAIN_PROMPT,
                  callbacks=None,
                  verbose=True)

document_prompt = PromptTemplate(
    input_variables=["page_content", "source"],
    template="Context:\ncontent:{page_content}\nsource:{source}",
)

combine_documents_chain = StuffDocumentsChain(
                  llm_chain=llm_chain,
                  document_variable_name="context",
                  document_prompt=document_prompt,
```

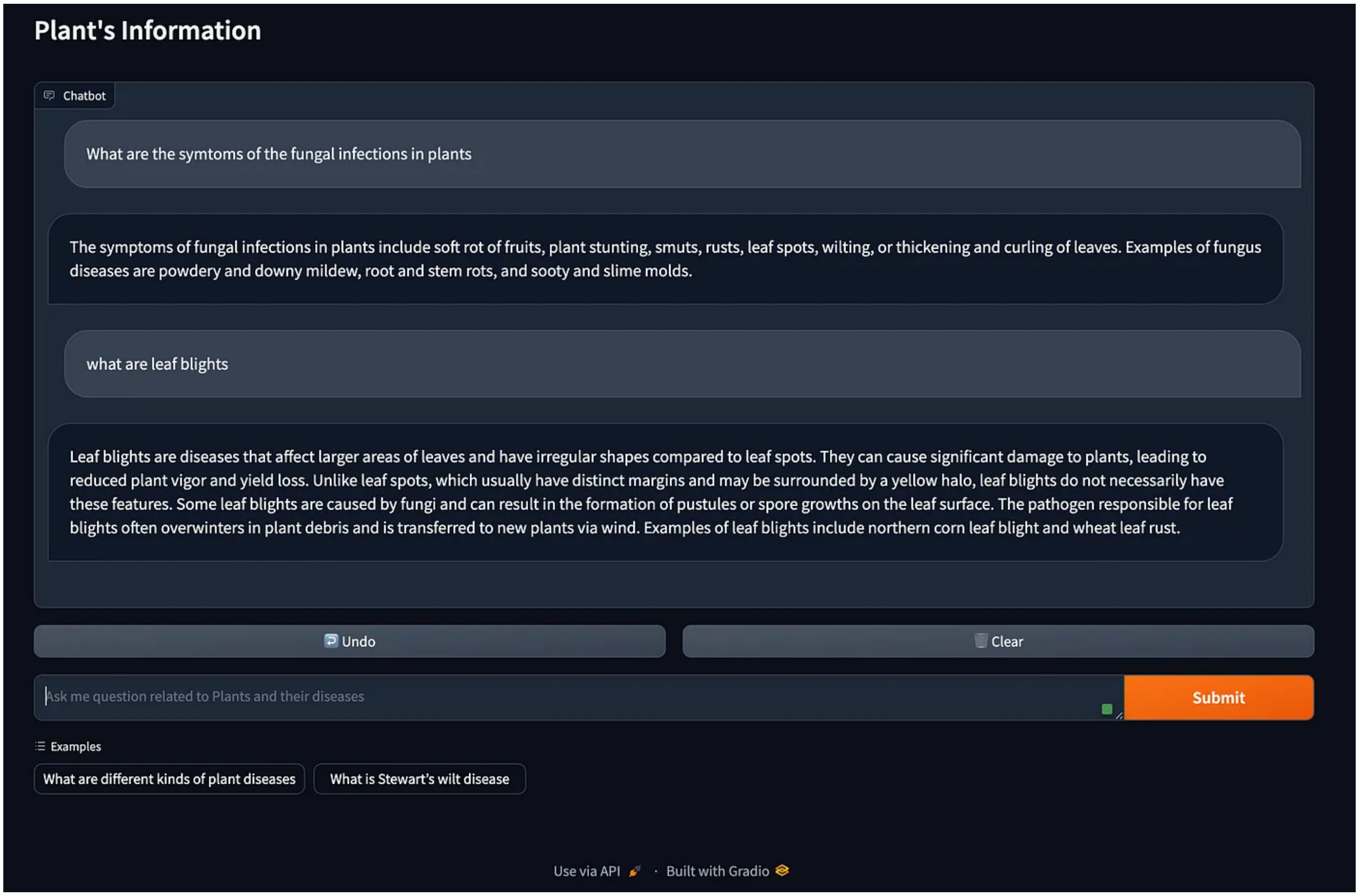
```
                    callbacks=None)

qa = RetrievalQA(
                    combine_documents_chain=combine_documents_chain,
                    verbose=True,
                    retriever=retriever,
                    return_source_documents=True)

def respond(question,history):
    return qa(question)["result"]


gr.ChatInterface(
    respond,
    chatbot=gr.Chatbot(height=500),
    textbox=gr.Textbox(placeholder="Ask me question related to Plants and their
    title="Plant's Chatbot",
    examples=["What are different kinds of plant diseases", "What is Stewart’s
    cache_examples=True,
    retry_btn=None,

).launch(share = True)
```



This is a very basic example of RAG, moving forward we will explore more functionalities of Langchain, and Llamaindex and gradually move to advanced concepts.

Enjoyyyy…!!! It's time to rock, sorry RAG

Find more such blogs below

Medium Search Write

Basic to Advanced RAG using LlamaIndex ~1

Welcome to "Basic to Advanced RAG using LlamaIndex ~1" the first installment in a comprehensive blog series dedicated...

medium.com

Basic to Advanced RAG using LlamaIndex (Estimating optimal chunk size)~ 2

Welcome back to the series of Basic to Advanced RAG. We are moving forward with the second installment of the series...

medium.com

Basic to Advanced RAG using LlamaIndex (Mastering Embedding Model Selection for Peak Performance)~...

At the heart of RAG's success lies a critical component: choosing the right embedding models. In this blog, we'll...

medium.com

Basic to Advanced RAG using LlamaIndex (Optimizing Performance with Rerankers)~4

Re-rankers are key to making RAG systems more accurate. They help by sorting and selecting the most relevant...

medium.com

Playing with Google Gemini Pro

Imagine a tool that can craft captivating stories, translate languages, summarize images, write poems for you, and…

medium.com

References:

https://python.langchain.com/docs/get_started/introduction

https://medium.com/@akriti.upadhyay/implementing-rag-with-langchain-and-hugging-face-28e3ea66c5f7

Enhancing LangChain's RetrievalQA for Real Source Links

Introduction

nakamasato.medium.com

Retrieval Augmented Gen | Ollama | Langchain | Mistral | Llm

Written by Abhishek Selokar

Follow

143 Followers · 112 Following

Masters Student @ Indian Institute Of Technology, Kharagpur || Thirsty to learn more about AI

Responses (1)

See all responses

Help Status About Careers Press Blog Privacy Terms Text to speech Teams