```
!pip -q install langchain huggingface_hub tiktoken pypdf
```

```
!pip install google-generativeai chromadb
```

Show hidden output

```
!pip install sentence-transformers
```

Show hidden output

```
!pip install langchain_community
```

Show hidden output

```
import os
from google.colab import userdata


GOOGLE_API_KEY=userdata.get('GOOGLE_API_KEY')
os.environ["GOOGLE_API_KEY"]=GOOGLE_API_KEY
```

```
!pip install langchain_google_genai
```

Show hidden output

```
from langchain_google_genai import ChatGoogleGenerativeAI
llm=ChatGoogleGenerativeAI(model="gemini-1.5-pro")


from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain.vectorstores.chroma import Chroma
import langchain
```

```
from langchain_community.document_loaders import PyPDFLoader


data_path="/content/RAG-FUSION.pdf"


loader=PyPDFLoader(data_path)
docs=loader.load()


docs
```

Show hidden output

```
text_splitter=RecursiveCharacterTextSplitter(chunk_size=500,chunk_overlap=100)


texts = []
for doc in docs:
    texts.extend(text_splitter.split_text(doc.page_content))


texts
```

Show hidden output

```
from langchain.embeddings import HuggingFaceBgeEmbeddings
model_name="BAAI/bge-small-en-v1.5"


encode_kwargs={'normalize_embeddings':True}


embedding_function=HuggingFaceBgeEmbeddings(model_name=model_name,encode_kwargs=encode_kwargs)


db=Chroma.from_texts(texts,embedding_function,persist_directory="./chroma_db")


query="challenges of rag fusion"
```

```
db.similarity_search(query,k=5)
```

```
[Document(metadata={}, page_content='Challenges of RAG-Fusion'),
 Document(metadata={}, page_content='RAG vs RAG-Fusion'),
 Document(metadata={}, page_content='Figure 1: Diagram illustrating the high level process of RAG-Fusion starting with the
original'),
 Document(metadata={}, page_content='that the slowness of RAG-Fusion'),
 Document(metadata={}, page_content='runs shows that RAG-Fusion')]
```

```
retriever=db.as_retriever(k=3)
```

```
retriever.get_relevant_documents(query)
```

```
[Document(metadata={}, page_content='Challenges of RAG-Fusion'),
 Document(metadata={}, page_content='RAG vs RAG-Fusion'),
 Document(metadata={}, page_content='Figure 1: Diagram illustrating the high level process of RAG-Fusion starting with the
original'),
 Document(metadata={}, page_content='that the slowness of RAG-Fusion')]
```

```
from operator import itemgetter
from langchain.prompts import ChatPromptTemplate
from langchain.schema.output_parser import StrOutputParser
from langchain.schema.runnable import RunnablePassthrough,RunnableLambda
```

```
template="""Give detail answer based on the following question
        context:{context}
        question:{question}
        """
```

```
prompt=ChatPromptTemplate.from_template(template)
chain=({"context":retriever,"question":RunnablePassthrough()}
      |prompt
      |RunnableLambda(lambda x: x.messages[0].content)
      |StrOutputParser()
      |llm)
```

```
text_reply=chain.invoke("what is rag vs rag fusion")
print(text_reply)
```

```
or enhanced retrieval and/or processing approach compared to standard RAG.  More information is needed to fully understand the dis
```

```
from langchain.prompts import ChatPromptTemplate, SystemMessagePromptTemplate, HumanMessagePromptTemplate, PromptTemplate

prompt = ChatPromptTemplate(
    input_variables=["original_query"],
    messages=[
        SystemMessagePromptTemplate(
            prompt=PromptTemplate(
                input_variables=[],
                template="You are a helpful assistant that generates multiple search queries based on a single input query"
            )
        ),
        HumanMessagePromptTemplate(
            prompt=PromptTemplate(
                input_variables=['original_query'],
                template="Generate multiple search queries related to: {original_query} \n Output('4 queries'):"
            )
        )
    ]
)


original_query="What is MEMS?"
generate_queries=(prompt |llm |StrOutputParser() |(lambda x:x.split("\n")))


from langchain.load import dumps,loads


def reciprocal_rank_fusion(results: list[list], k=60):
  fused_score = {}
  for docs in results:
    for rank, doc in enumerate(docs):
      doc_str = dumps(doc)
```

```
        if doc_str not in fused_score:
          fused_score[doc_str] = 0
        previous_score = fused_score[doc_str]
        fused_score[doc_str] += 1 / (rank + k)
    reranked_result = [(loads(doc), score)
           for doc, score in sorted(fused_score.items(), key=lambda x: x[1], reverse=True)]
    return reranked_result


ragfusion_chain=generate_queries |retriever.map() |reciprocal_rank_fusion
langchain.debug=True


ragfusion_chain.input_schema.schema()
ragfusion_chain.invoke({"original_query":original_query})
```

```
        reranked_result = [(loads(doc), score)
```

```
Document(metadata={}, page_content='was regarding whether MEMs'),
0.04972677595628415),
Document(metadata={}, page_content='MEMS (Micro-Electro-Mechanical Systems) microphones are small devices that convert sound
ves'),
0.048131080389144903),
Document(metadata={}, page_content='• What are MEMs microphones and how do they work?'),
0.03306010928961749),
Document(metadata={}, page_content='to our silicon MEMs microphones and metal-oxide-semiconductor field-effect transistors
OSFETs).'),
0.03279569892473118),
Document(metadata={}, page_content='Dual Membrane MEMS technology.'),
0.032266458495966696),
Document(metadata={}, page_content='handling noise, as MEMS microphones can be more sensitive to vibrations and movements
mpared'),
0.01639344262295082),
Document(metadata={}, page_content='RAG-Fusion: a New Take on Retrieval-Augmented Generation\ningress IP57 protection at a
crophone level because of Infineon’s latest Sealed Dual Membrane MEMS technology.\nThis missing part may be solved, however,
 increasing the number of queries generated or documents retrieved.\nIn addition to technical product questions, many
gineers also ask for help troubleshooting or product instructions.'),
0.016129032258064516),
Document(metadata={}, page_content='comprehensive answer drawing on its prior trained knowledge and database of MEMs
crophones.'),
0.015873015873015872),
Document(metadata={}, page_content='comprehensive answer drawing on its prior trained knowledge and database of MEMs
crophones.\nMEMS (Micro-Electro-Mechanical Systems) microphones are small devices that convert sound waves\ninto electrical
gnals. They utilize a diaphragm that vibrates in response to sound waves and an\nacoustic sensor that detects these
brations. The sensor then translates the vibrations into electrical\nsignals, which can be amplified and processed for
rious applications.'),
0.015873015873015872)]
```

Start coding or generate with AI.