```
%pip install -qU langchain_community langchain_pinecone
langchain_google_genai langchain_text_splitters pypdf
```

```
Note: you may need to restart the kernel to use updated packages.
```

```
import os
os.environ["GOOGLE_API_KEY"] = "AIzaSyCxKCDxVCPWGYpPn74G-
bJ1M0ZQ5nghZb4"
os.environ["PINECONE_API_KEY"] =
"pcsk_74Qr8K_9eEJkqzjZw954mBFdSnYgiNWVBdQiw5eVrd81g1aP73BX5Lc33csBE1zE
czirbD"
```

```
from langchain_pinecone import PineconeVectorStore
from langchain_google_genai import ChatGoogleGenerativeAI
from langchain_community.document_loaders import PyPDFLoader
from langchain_text_splitters import RecursiveCharacterTextSplitter
from langchain.chains import RetrievalQA
```

```
#load the pdf
#specify the pdf path
pdf_path = "C:/Users/jithe/Downloads/Distributed representation.pdf"
```

```
#create a PyPDFLoader instance
loader = PyPDFLoader(pdf_path)
```

```
#load the document
data = loader.load()
```

```
len(data)
```

```
9
```

```
#Split pdf to chunks
text_splitter = RecursiveCharacterTextSplitter(chunk_size = 1000,
chunk_overlap = 200)
split_docs = text_splitter.split_documents(data)
```

```
split_docs
```

```
len (split_docs)
```

```
41
```

```
#Initiating embedding model
#Push data to Pinecone
embeddings = GoogleGenerativeAIEmbeddings(model="models/embedding-
001")
index_name = "jijaypdf"
```

```
from langchain_google_genai import GoogleGenerativeAIEmbeddings
```

```
#Initiating embedding model
embeddings = GoogleGenerativeAIEmbeddings(model="models/embedding-
001")
index_name = "jijaypdf"

#Create vector store to upload data to Pinecode
vectorstore = PineconeVectorStore.from_documents(split_docs,
embeddings, index_name=index_name)
vectorstore
```

```
<langchain_pinecone.vectorstores.PineconeVectorStore at 0x288bd8401a0>
```

```
#fetching similar results from vector store
query = " what is learning good vector representations?"
similar_docs = vectorstore.similarity_search(query)
```

```
similar_docs
```

```
len(similar_docs)
```

```
4
```

```
#Setup retrieval QA chain
llm = ChatGoogleGenerativeAI(model="gemini-1.5-pro", temprature = 0)
qa = RetrievalQA.from_chain_type(
    llm=llm,
    chain_type = "stuff",
    retriever = vectorstore.as_retriever()
)
```

```
response = qa.invoke('What is the main content of the document?')
response
```

```
#qauery the pdf
response = qa.invoke('What is the main content of the document?')
response['result']
```

```
'This document introduces the Skip-gram model for learning distributed
representations of words and phrases.  Key contributions include
demonstrating the linear structure of these representations, enabling
analogical reasoning; training on much larger datasets than previous
work, leading to higher quality representations, especially for rare
words; introducing the Negative Sampling training algorithm; and
exploring methods for combining word vectors, such as simple vector
addition, to represent phrases.  The authors also compare their model
to previously published work, showing significant improvements in
representation quality.'
```