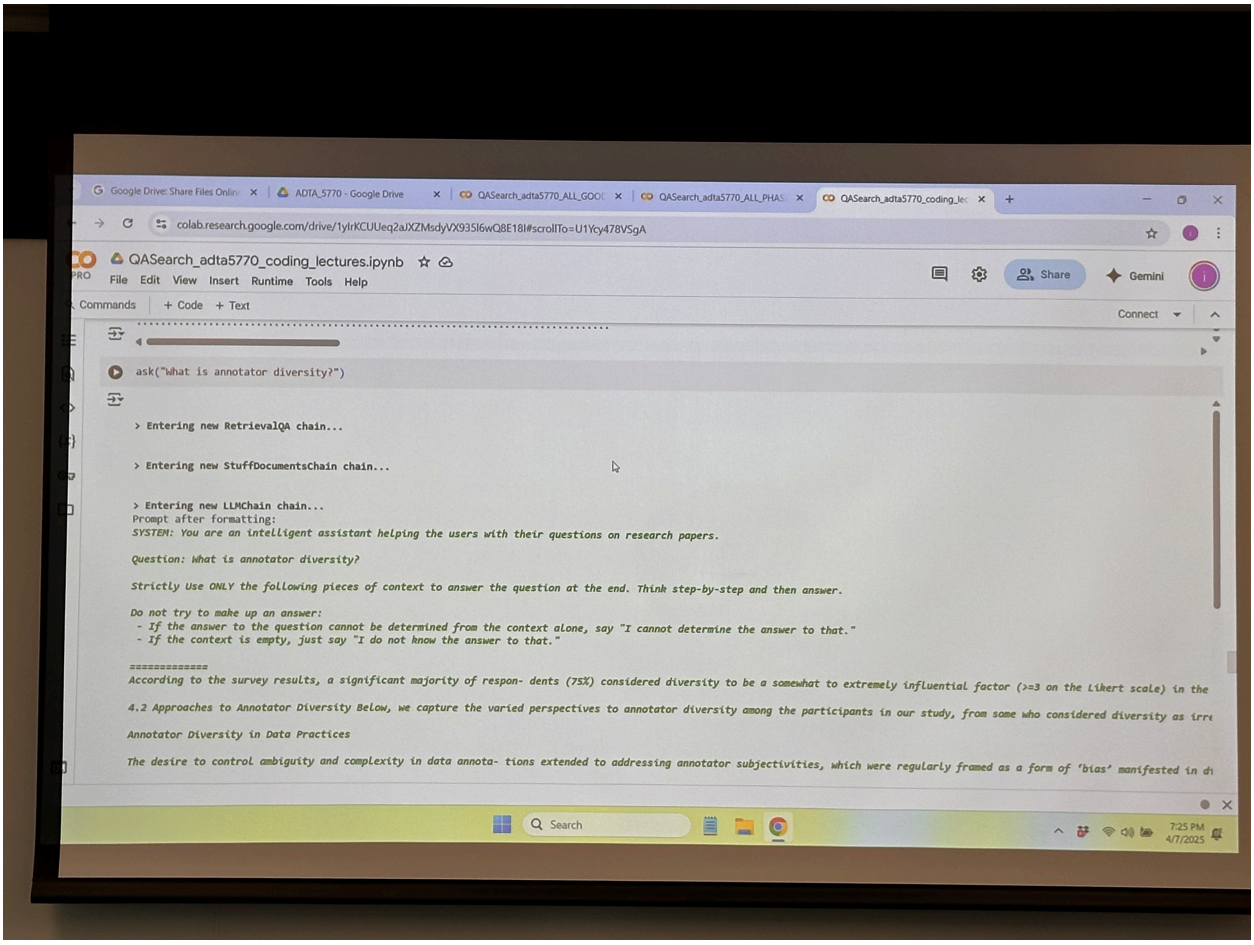
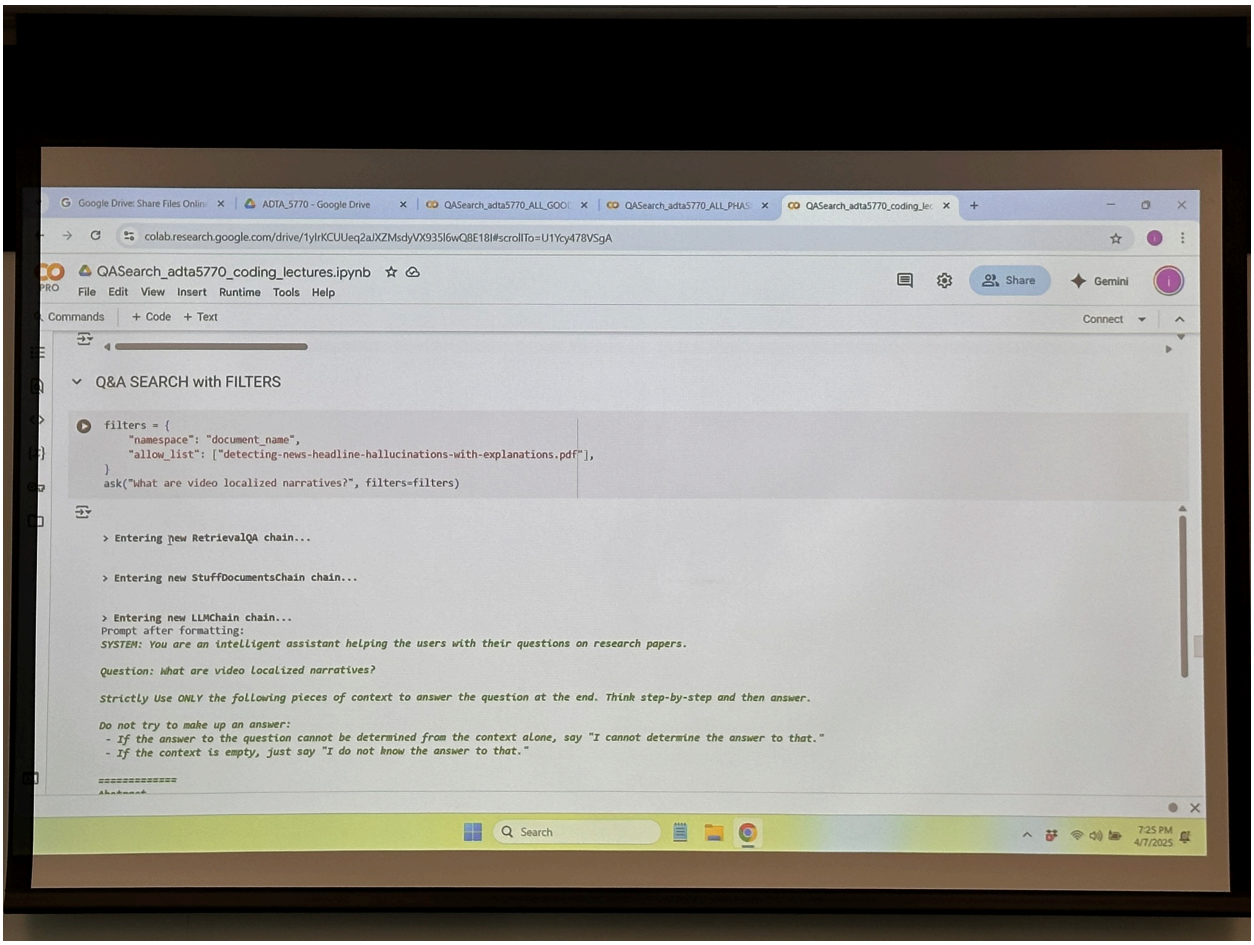
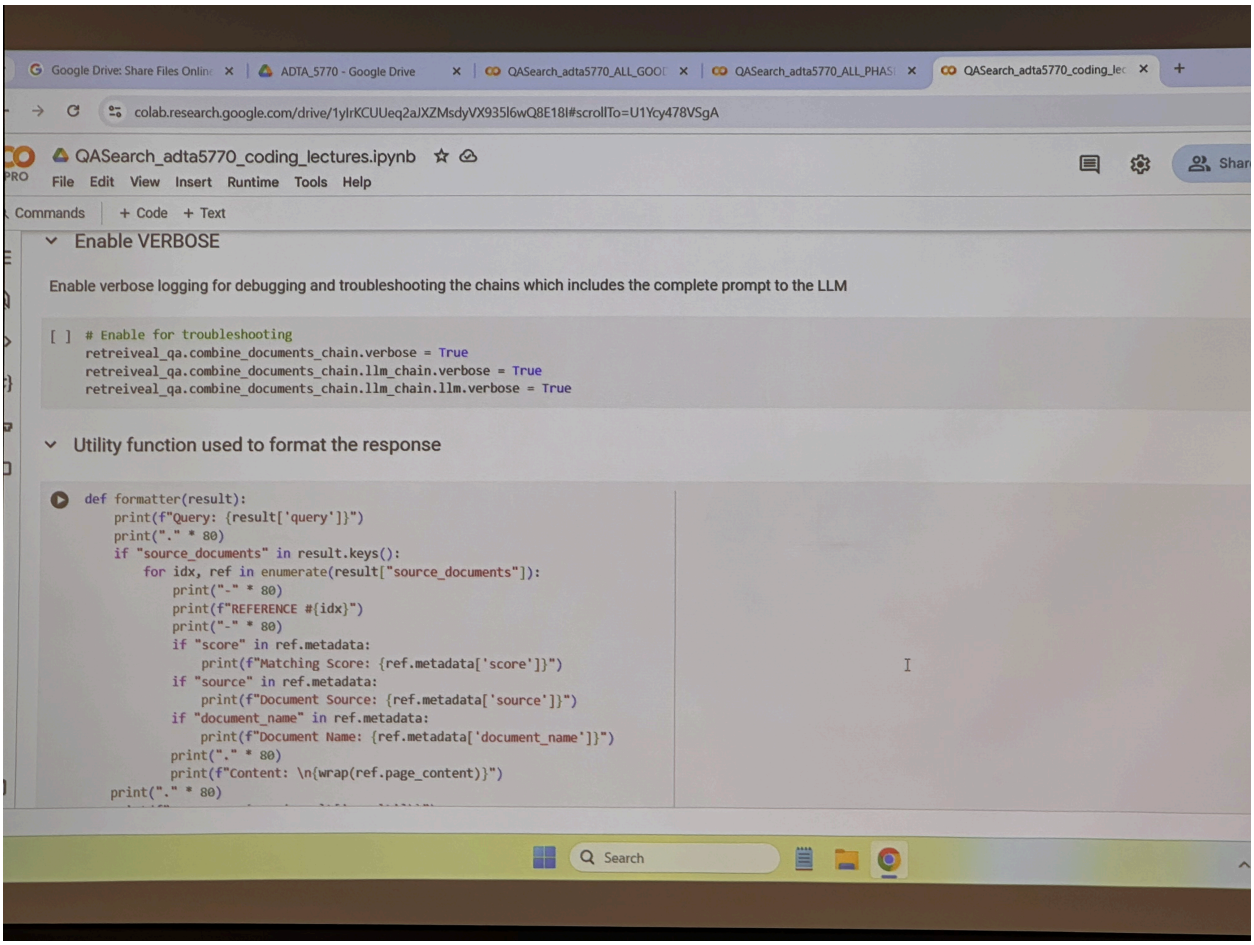
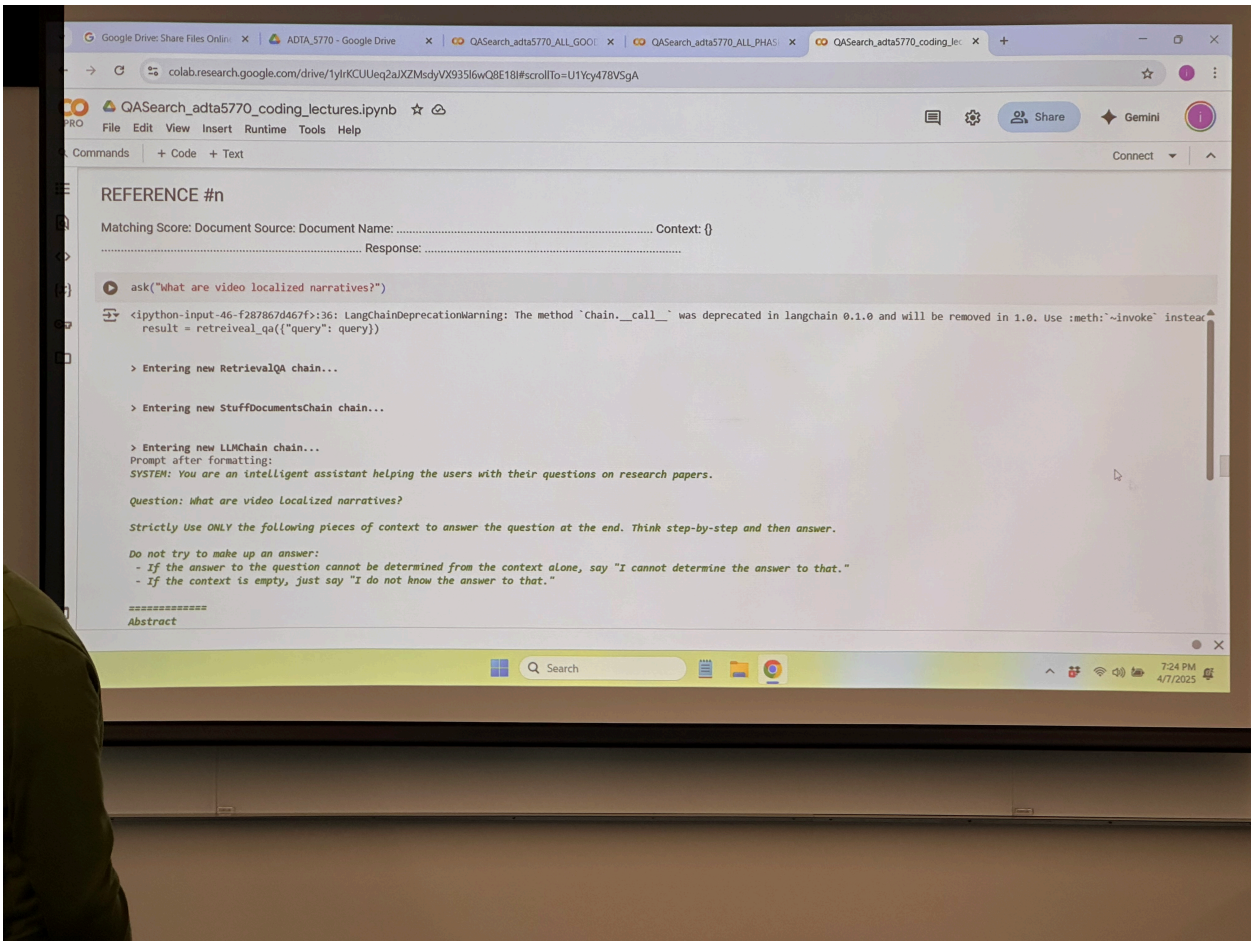
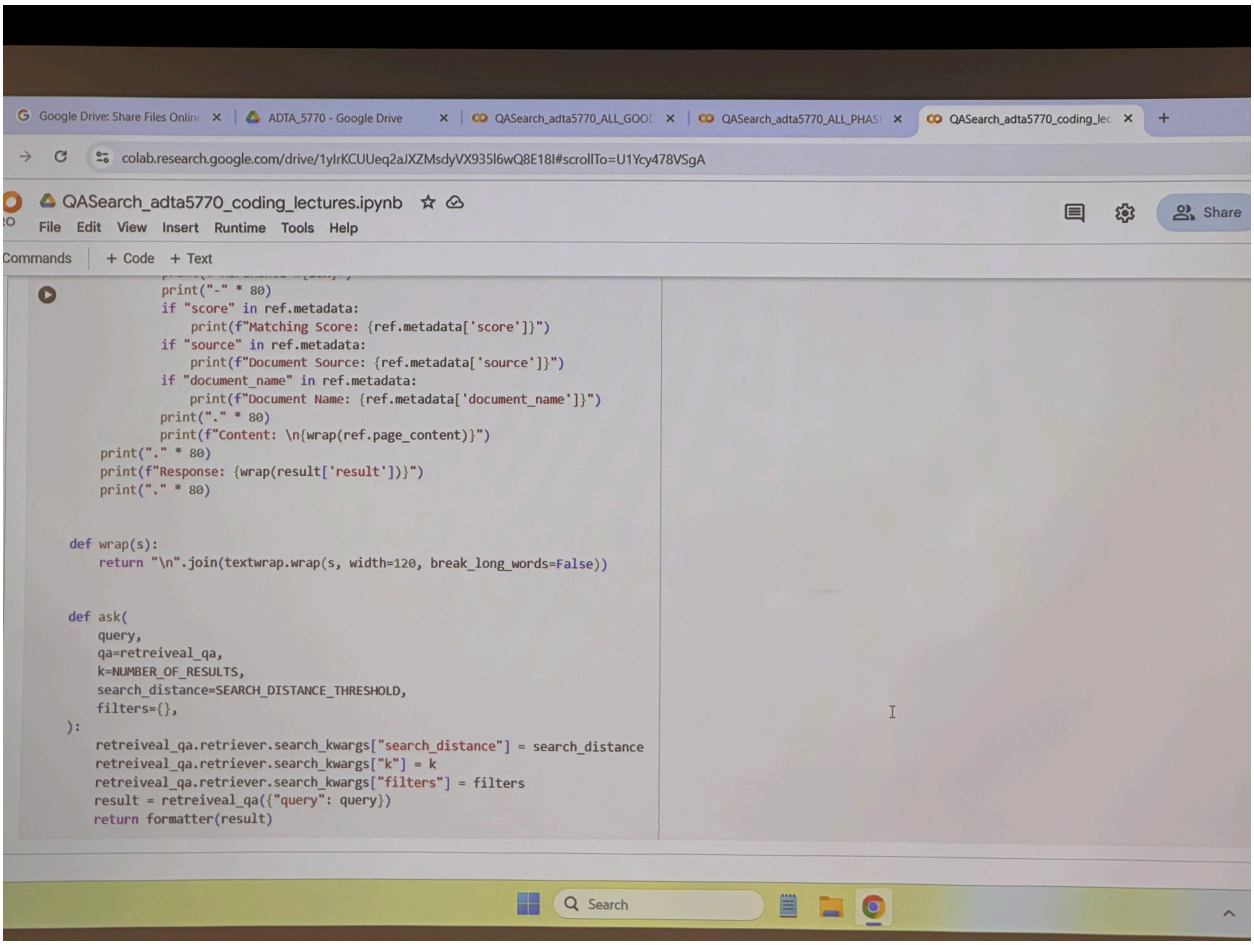
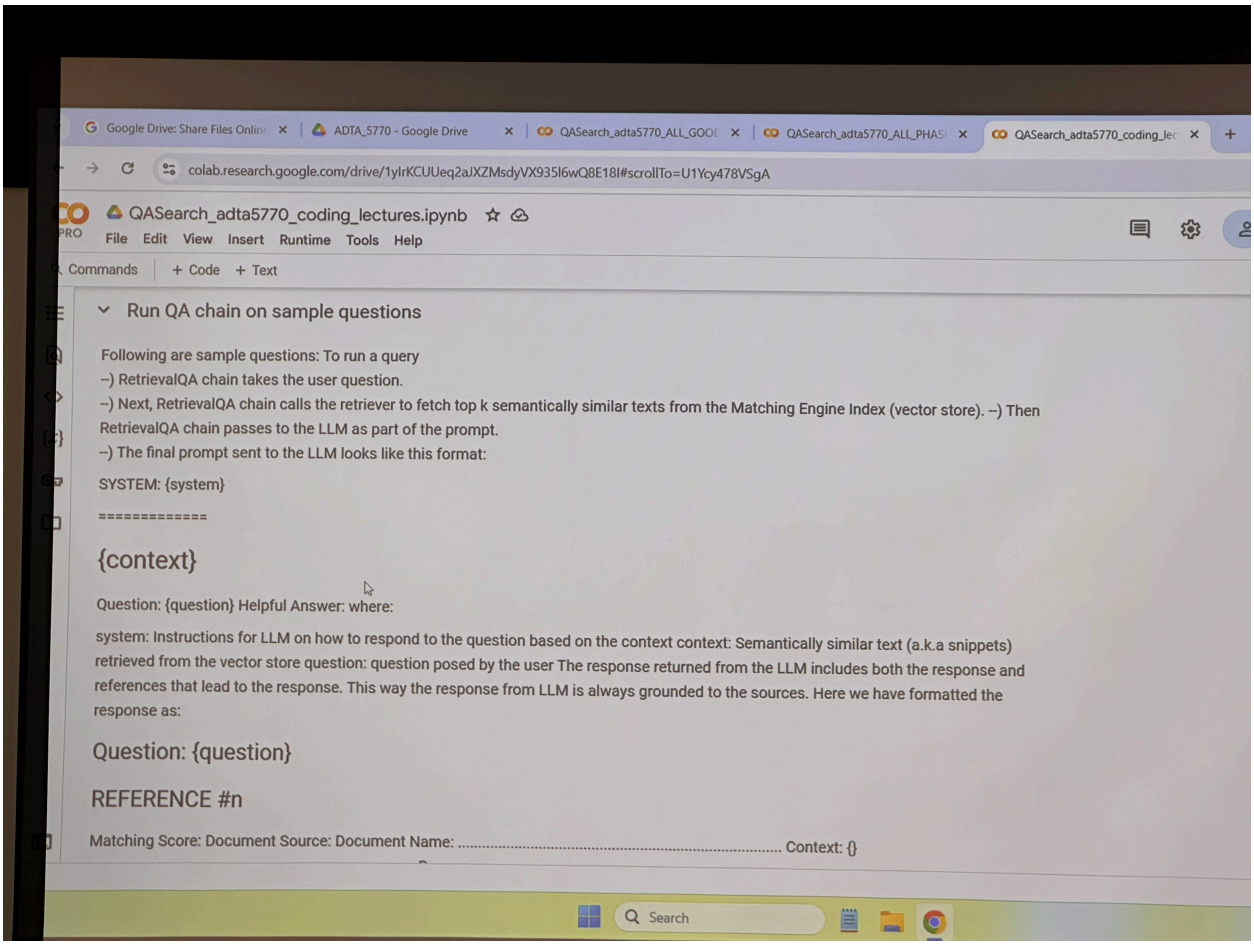

QASearch_adta5770_coding_lectures.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Connect
ask("what is annotator diversity?")
> Entering new RetrievalQA chain...
> Entering new StuffDocumentsChain chain...
> Entering new LLMChain chain...
Prompt after formatting:
SYSTEM: You are an intelligent assistant helping the users with their questions on research papers.
Question: What is annotator diversity?
Strictly Use ONLY the following pieces of context to answer the question at the end. Think step-by-step and then answer.
Do not try to make up an answer:
- If the answer to the question cannot be determined from the context alone, say "I cannot determine the answer to that."
- If the context is empty, just say "I do not know the answer to that."
==========
According to the survey results, a significant majority of respon- dents (75%) considered diversity to be a somewhat to extremely influential factor (>=3 on the Likert scale) in the
4.2 Approaches to Annotator Diversity Below, we capture the varied perspectives to annotator diversity among the participants in our study, from some who considered diversity as irr
Annotator Diversity in Data Practices
The desire to control ambiguity and complexity in data annota- tions extended to addressing annotator subjectivities, which were regularly framed as a form of 'bias' manifested in d
Search
7:25 PM
4/7/2025

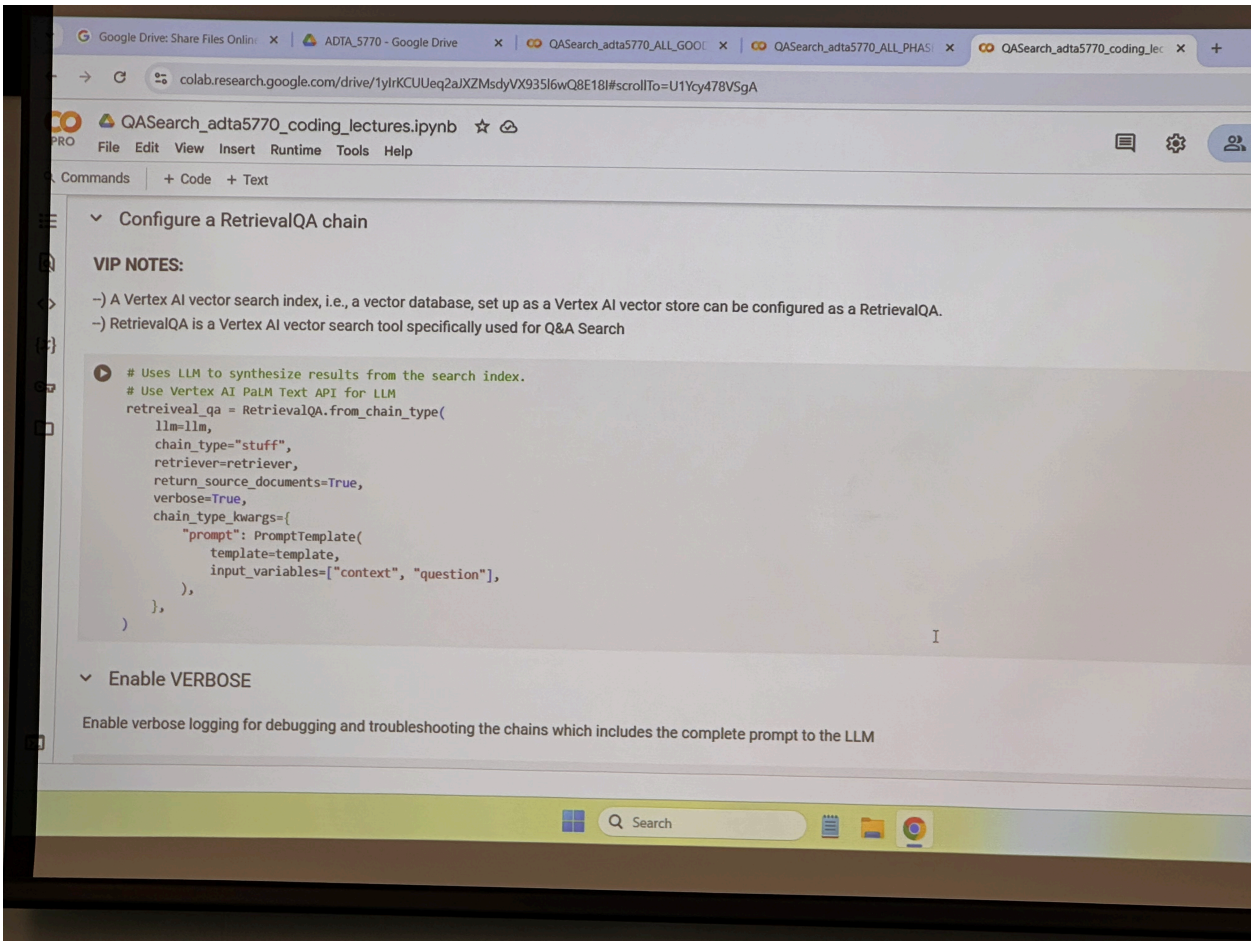

QASearch_adta5770_coding_lectures.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Connect
+ Code + Text
Q&A SEARCH with FILTERS
filters = {
"namespace": "document_name",
"allow_list": ["detecting-news-headline-hallucinations-with-explanations.pdf"],
}
ask("What are video localized narratives?", filters=filters)
> Entering new RetrievalQA chain...
> Entering new StuffDocumentsChain chain...
> Entering new LLMChain chain...
Prompt after formatting:
SYSTEM: You are an intelligent assistant helping the users with their questions on research papers.
Question: What are video localized narratives?
Strictly Use ONLY the following pieces of context to answer the question at the end. Think step-by-step and then answer.
Do not try to make up an answer:
- If the answer to the question cannot be determined from the context alone, say "I cannot determine the answer to that."
- If the context is empty, just say "I do not know the answer to that."
==========
Search
7:25 PM
4/7/2025

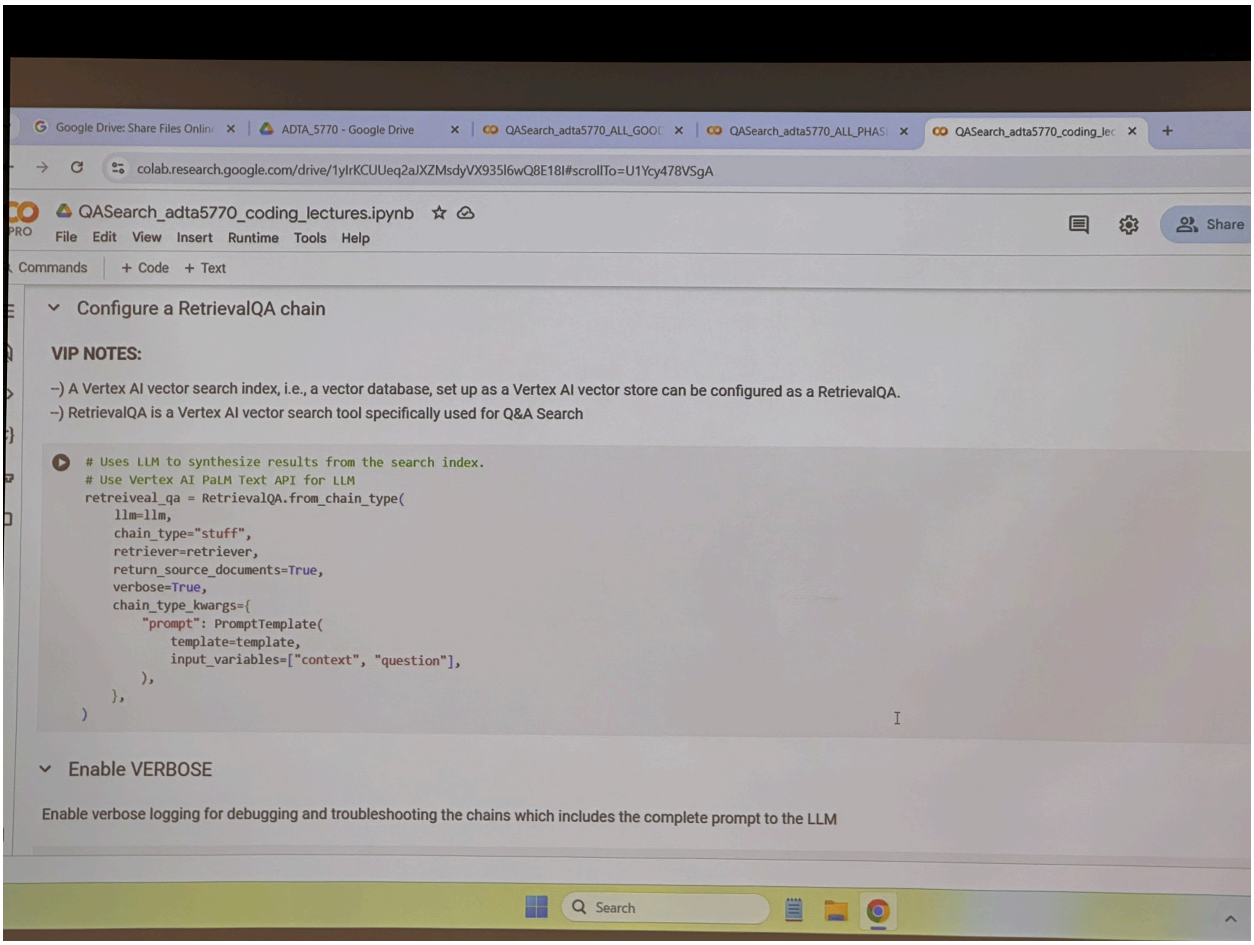

Google Drive: Share Files Online
ADTA_5770 - Google Drive
QASearch_adta5770_ALL_GOOD
QASearch_adta5770_ALL_PHASE
QASearch_adta5770_coding_lec
colab.research.google.com/drive/1ylrKCUUeq2aJXZMsdyVX935l6wQ8E18I#scrollTo=U1Ycy478VSgA
QASearch_adta5770_coding_lectures.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text
Enable VERBOSE
Enable verbose logging for debugging and troubleshooting the chains which includes the complete prompt to the LLM
[ ] # Enable for troubleshooting
retreiveal_qa.combine_documents_chain.verbose = True
retreiveal_qa.combine_documents_chain.llm_chain.verbose = True
retreiveal_qa.combine_documents_chain.llm_chain.llm.verbose = True
Utility function used to format the response
def formatter(result):
print(f"Query: {result['query']}")
print("." * 80)
if "source_documents" in result.keys():
for idx, ref in enumerate(result["source_documents"]):
print("-" * 80)
print(f"REFERENCE #{idx}")
print("-" * 80)
if "score" in ref.metadata:
print(f"Matching Score: {ref.metadata['score']}")
if "source" in ref.metadata:
print(f"Document Source: {ref.metadata['source']}")
if "document_name" in ref.metadata:
print(f"Document Name: {ref.metadata['document_name']}")
print("." * 80)
print(f"Content: \n{wrap(ref.page_content)}")
print("." * 80)
Search

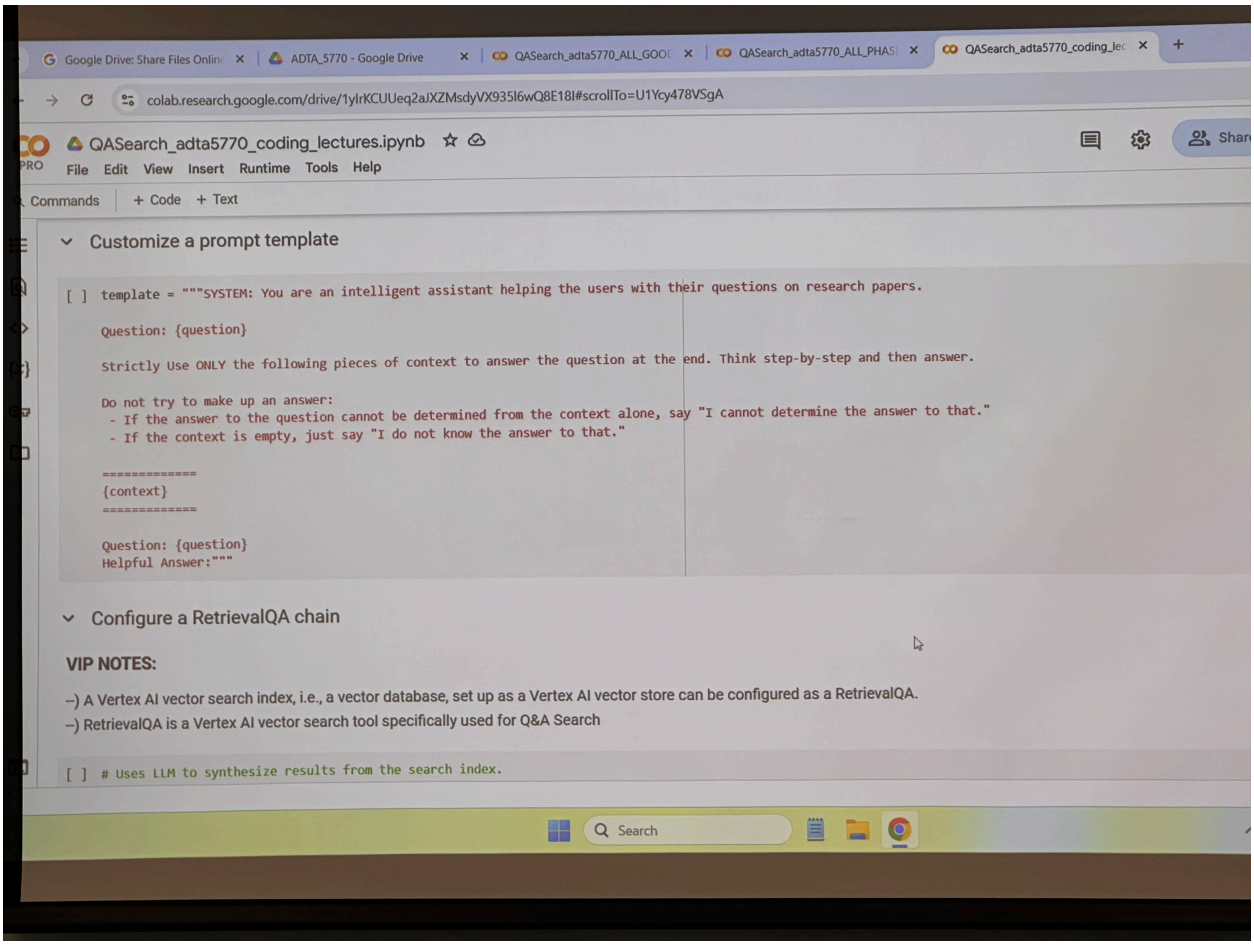

Google Drive: Share Files Onlin
ADTA_5770 - Google Drive
QASearch_adta5770_ALL_GOO
QASearch_adta5770_ALL_PHAS
QASearch_adta5770_coding_lec
colab.research.google.com/drive/1yIrKCUUeq2aJXZMsdyVX935l6wQ8E18l#scrollTo=U1Ycy478VSgA
QASearch_adta5770_coding_lectures.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code + Text
Connect
REFERENCE #n
Matching Score: Document Source: Document Name: ..... Context: {}
..... Response: .....
ask("What are video localized narratives?")
<ipython-input-46-f287867d467f>:36: LangChainDeprecationWarning: The method `Chain.__call__` was deprecated in langchain 0.1.0 and will be removed in 1.0. Use :meth:`~invoke` instead
result = retreiveal_qa({"query": query})
> Entering new RetrievalQA chain...
> Entering new StuffDocumentsChain chain...
> Entering new LLMChain chain...
Prompt after formatting:
SYSTEM: You are an intelligent assistant helping the users with their questions on research papers.
Question: What are video localized narratives?
Strictly Use ONLY the following pieces of context to answer the question at the end. Think step-by-step and then answer.
Do not try to make up an answer:
- If the answer to the question cannot be determined from the context alone, say "I cannot determine the answer to that."
- If the context is empty, just say "I do not know the answer to that."
==========
Abstract
Search
7:24 PM
4/7/2025

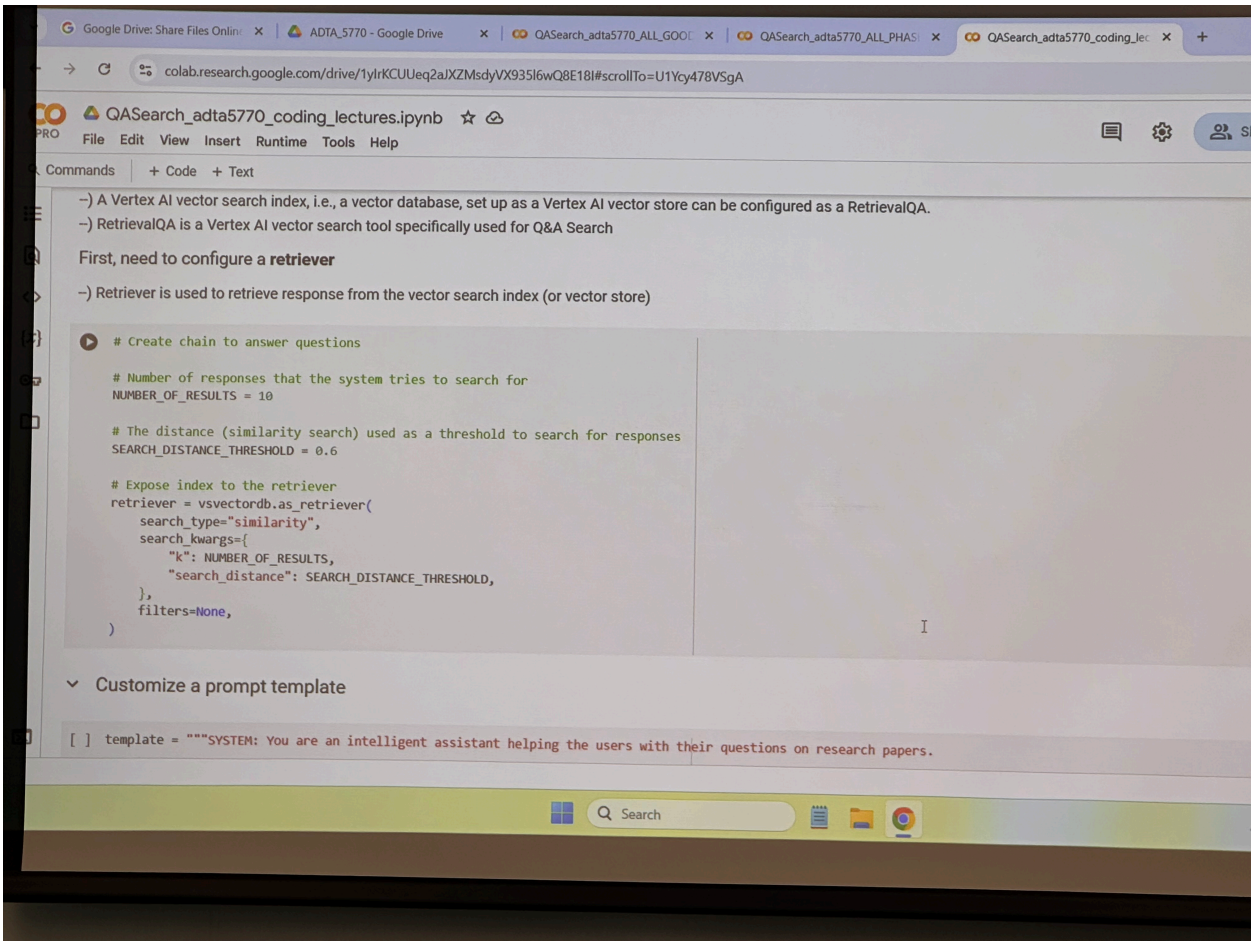

Google Drive: Share Files Onlin
ADTA_5770 - Google Drive
QASearch_adta5770_ALL_GOOI
QASearch_adta5770_ALL_PHA5
QASearch_adta5770_coding_lec
colab.research.google.com/drive/1yIrKCUUeq2aJXZMsdyVX935l6wQ8E18I#scrollTo=U1Ycy478VSgA
QASearch_adta5770_coding_lectures.ipynb
File Edit View Insert Runtime Tools Help
Share
Commands
+ Code
+ Text
print("-" * 80)
if "score" in ref.metadata:
print(f"Matching Score: {ref.metadata['score']}")
if "source" in ref.metadata:
print(f"Document Source: {ref.metadata['source']}")
if "document_name" in ref.metadata:
print(f"Document Name: {ref.metadata['document_name']}")
print("." * 80)
print(f"Content: \n{wrap(ref.page_content)}")
print("." * 80)
print(f"Response: {wrap(result['result'])}")
print("." * 80)
def wrap(s):
return "\n".join(textwrap.wrap(s, width=120, break_long_words=False))
def ask(
query,
qa=retreiveal_qa,
k=NUMBER_OF_RESULTS,
search_distance=SEARCH_DISTANCE_THRESHOLD,
filters={},
):
retreiveal_qa.retriever.search_kwargs["search_distance"] = search_distance
retreiveal_qa.retriever.search_kwargs["k"] = k
retreiveal_qa.retriever.search_kwargs["filters"] = filters
result = retreiveal_qa({"query": query})
return formatter(result)
Search

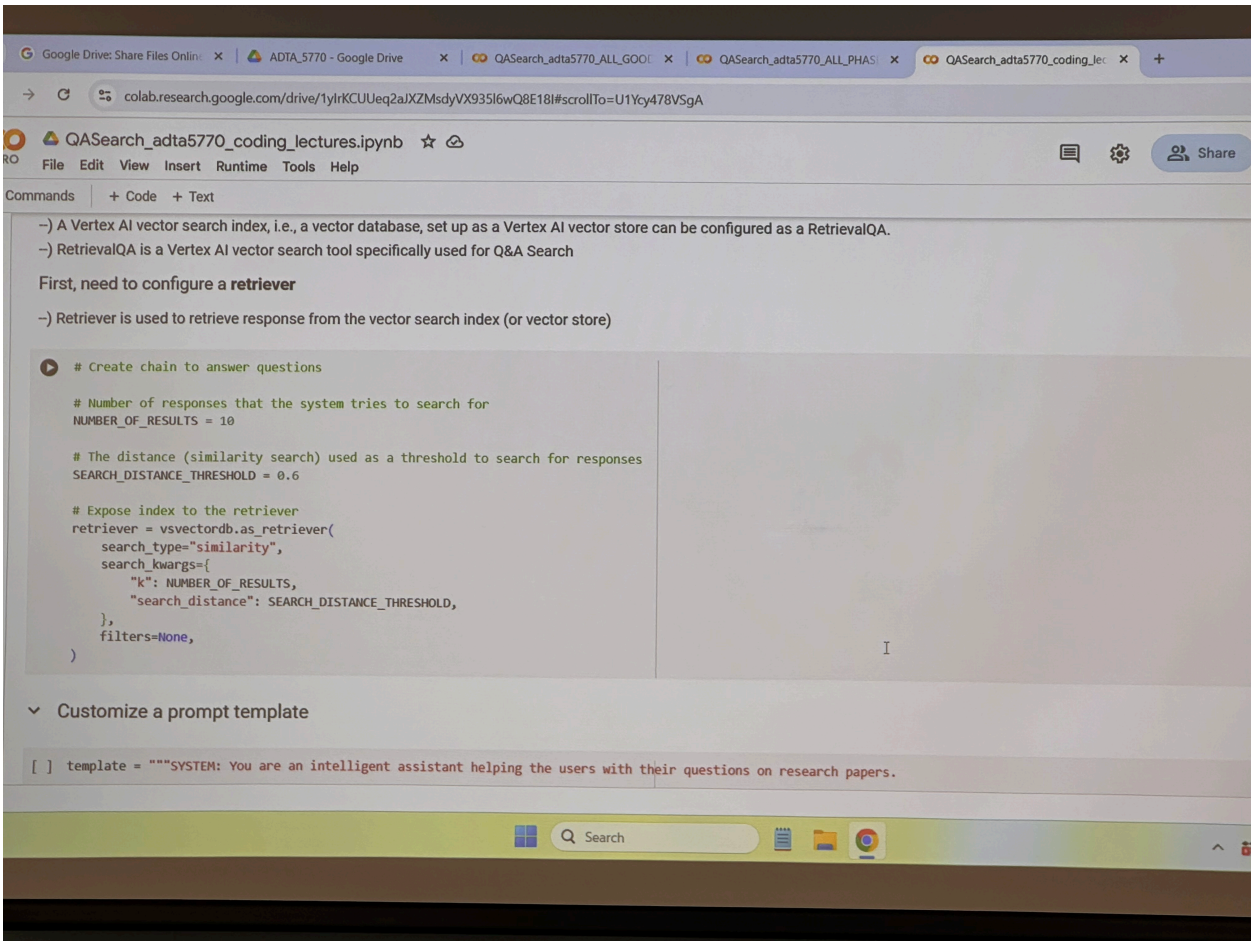

Google Drive: Share Files Onlin
ADTA_5770 - Google Drive
QASearch_adta5770_ALL_GOO
QASearch_adta5770_ALL_PHAS
QASearch_adta5770_coding_lec
colab.research.google.com/drive/1ylrKCUUeq2aJXZMsdyVX935l6wQ8E18I#scrollTo=U1Ycy478VSgA
QASearch_adta5770_coding_lectures.ipynb
File Edit View Insert Runtime Tools Help
Commands
+ Code
+ Text
Run QA chain on sample questions
Following are sample questions: To run a query
–) RetrievalQA chain takes the user question.
–) Next, RetrievalQA chain calls the retriever to fetch top k semantically similar texts from the Matching Engine Index (vector store). –) Then RetrievalQA chain passes to the LLM as part of the prompt.
–) The final prompt sent to the LLM looks like this format:
SYSTEM: {system}
==========
{context}
Question: {question} Helpful Answer: where:
system: Instructions for LLM on how to respond to the question based on the context context: Semantically similar text (a.k.a snippets) retrieved from the vector store question: question posed by the user The response returned from the LLM includes both the response and references that lead to the response. This way the response from LLM is always grounded to the sources. Here we have formatted the response as:
Question: {question}
REFERENCE #n
Matching Score: Document Source: Document Name: ..... Context: {}
Search

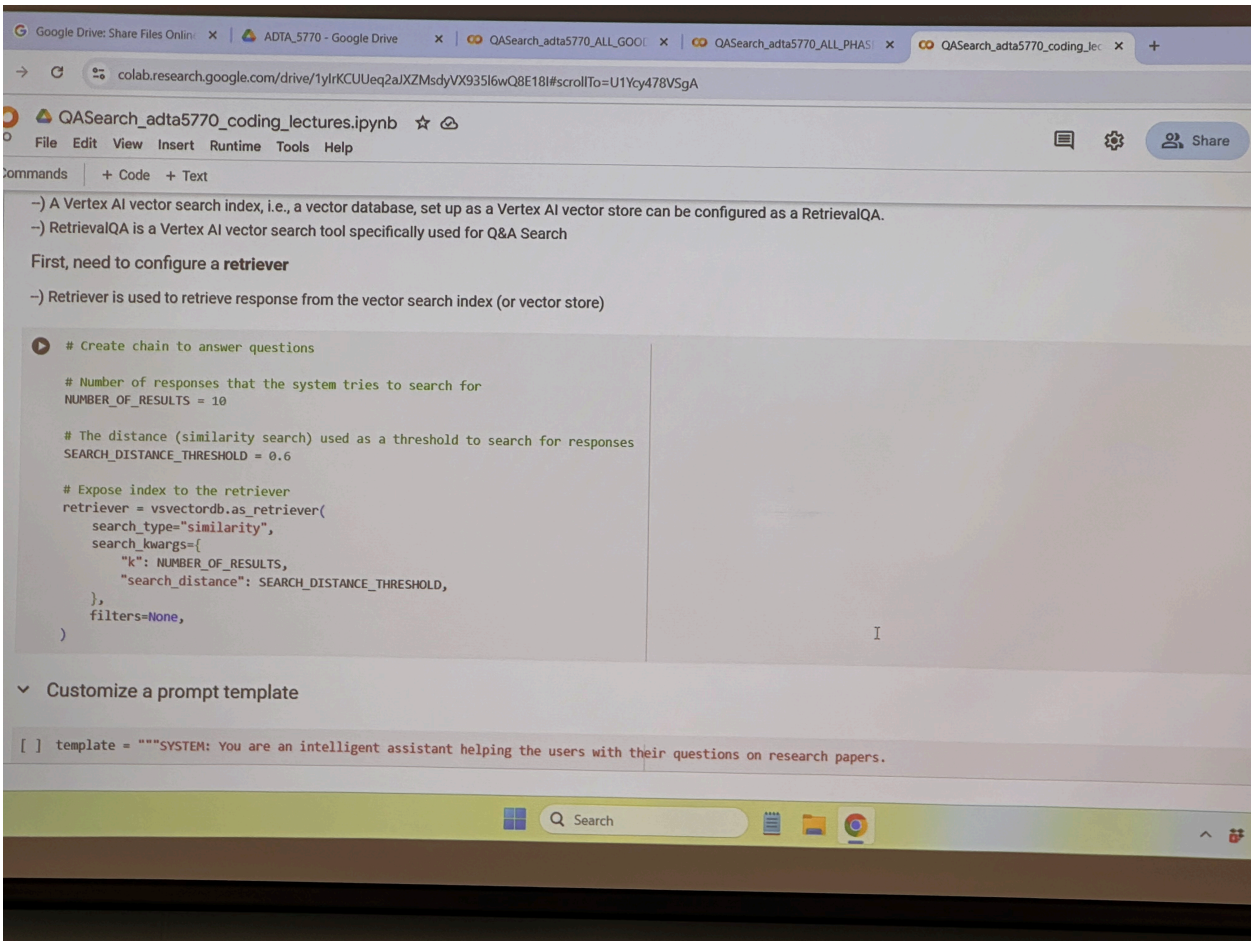

Google Drive: Share Files Online
ADTA_5770 - Google Drive
QASearch_adta5770_ALL_GOOD
QASearch_adta5770_ALL_PHASE
QASearch_adta5770_coding_lec
colab.research.google.com/drive/1yIrKCUUeq2aJXZMsdyVX935l6wQ8E18I#scrollTo=U1Ycy478VSgA
QASearch_adta5770_coding_lectures.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text
Configure a RetrievalQA chain
VIP NOTES:
--) A Vertex AI vector search index, i.e., a vector database, set up as a Vertex AI vector store can be configured as a RetrievalQA.
--) RetrievalQA is a Vertex AI vector search tool specifically used for Q&A Search
# Uses LLM to synthesize results from the search index.
# Use Vertex AI PaLM Text API for LLM
retreiveal_qa = RetrievalQA.from_chain_type(
llm=llm,
chain_type="stuff",
retriever=retriever,
return_source_documents=True,
verbose=True,
chain_type_kwargs={
"prompt": PromptTemplate(
template=template,
input_variables=["context", "question"],
),
},
)
Enable VERBOSE
Enable verbose logging for debugging and troubleshooting the chains which includes the complete prompt to the LLM
Search

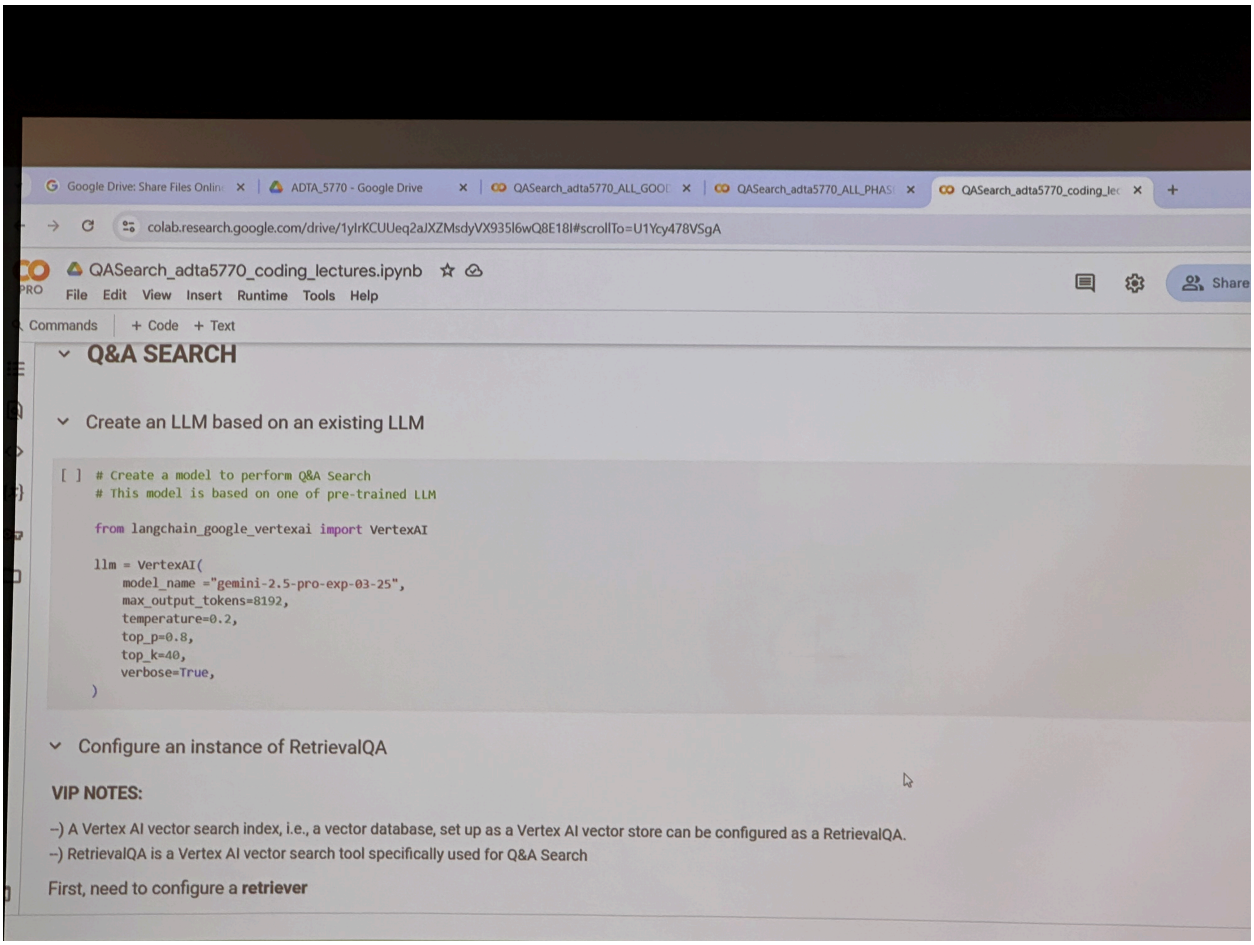

Google Drive: Share Files Onlin
ADTA_5770 - Google Drive
QASearch_adta5770_ALL_GOO
QASearch_adta5770_ALL_PHAS
QASearch_adta5770_coding_lec
colab.research.google.com/drive/1yIrKCUUeq2aJXZMsdyVX935l6wQ8E18I#scrollTo=U1Ycy478VSgA
QASearch_adta5770_coding_lectures.ipynb
File Edit View Insert Runtime Tools Help
Share
Commands
+ Code
+ Text
Configure a RetrievalQA chain
VIP NOTES:
–) A Vertex AI vector search index, i.e., a vector database, set up as a Vertex AI vector store can be configured as a RetrievalQA.
–) RetrievalQA is a Vertex AI vector search tool specifically used for Q&A Search
# Uses LLM to synthesize results from the search index.
# Use Vertex AI PaLM Text API for LLM
retreiveal_qa = RetrievalQA.from_chain_type(
llm=llm,
chain_type="stuff",
retriever=retriever,
return_source_documents=True,
verbose=True,
chain_type_kwargs={
"prompt": PromptTemplate(
template=template,
input_variables=["context", "question"],
),
},
)
Enable VERBOSE
Enable verbose logging for debugging and troubleshooting the chains which includes the complete prompt to the LLM
Search

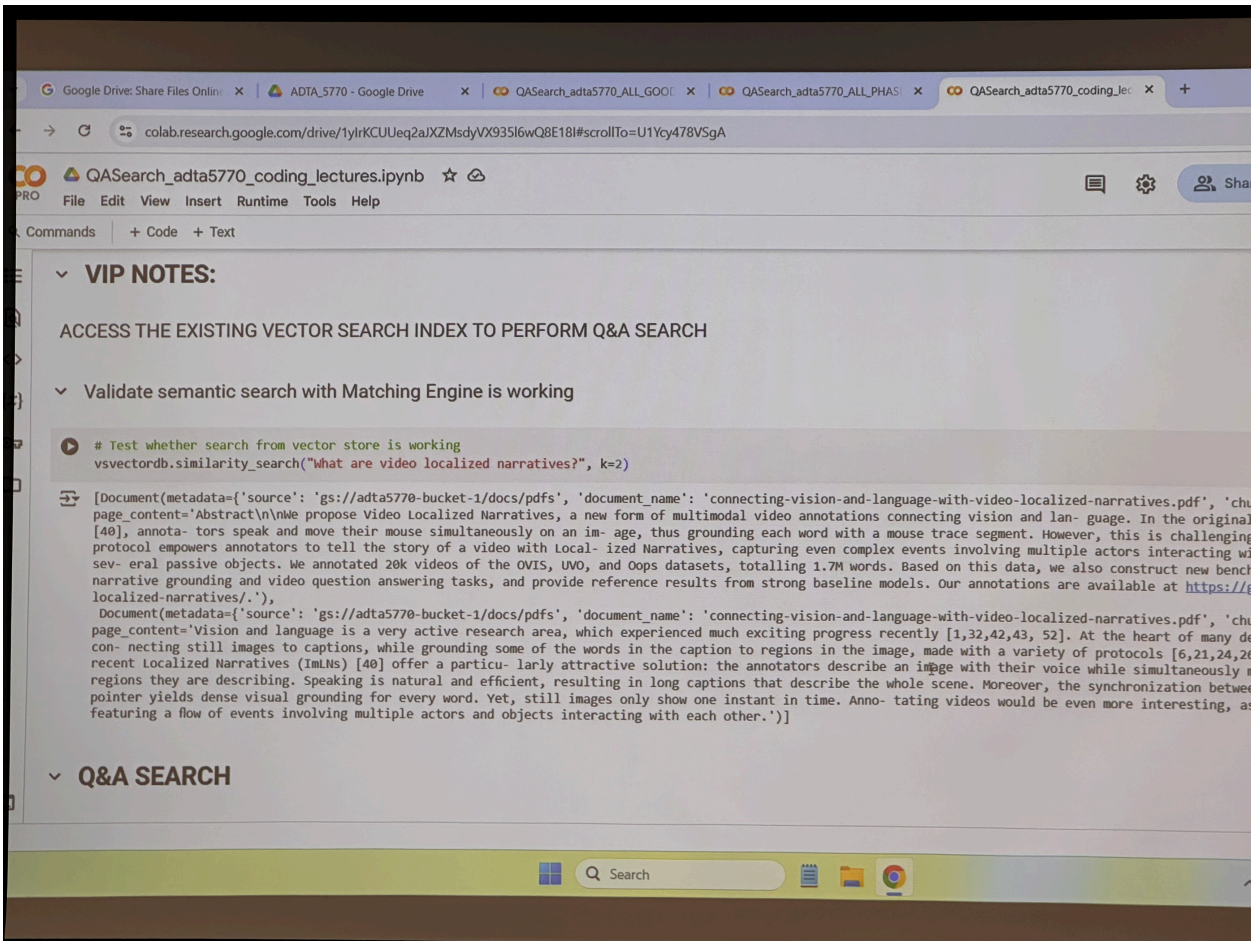

Google Drive: Share Files Onlin
ADTA_5770 - Google Drive
QASearch_adta5770_ALL_GOOD
QASearch_adta5770_ALL_PHAS
QASearch_adta5770_coding_lec
colab.research.google.com/drive/1yIrKCUUeq2aJXZMsdyVX935l6wQ8E18l#scrollTo=U1Ycy478VSgA
QASearch_adta5770_coding_lectures.ipynb
File Edit View Insert Runtime Tools Help
Commands
+ Code
+ Text
Customize a prompt template
[ ] template = """SYSTEM: You are an intelligent assistant helping the users with their questions on research papers.
Question: {question}
Strictly Use ONLY the following pieces of context to answer the question at the end. Think step-by-step and then answer.
Do not try to make up an answer:
- If the answer to the question cannot be determined from the context alone, say "I cannot determine the answer to that."
- If the context is empty, just say "I do not know the answer to that."
==========
{context}
==========
Question: {question}
Helpful Answer:"""
Configure a RetrievalQA chain
VIP NOTES:
–) A Vertex AI vector search index, i.e., a vector database, set up as a Vertex AI vector store can be configured as a RetrievalQA.
–) RetrievalQA is a Vertex AI vector search tool specifically used for Q&A Search
[ ] # Uses LLM to synthesize results from the search index.
Search

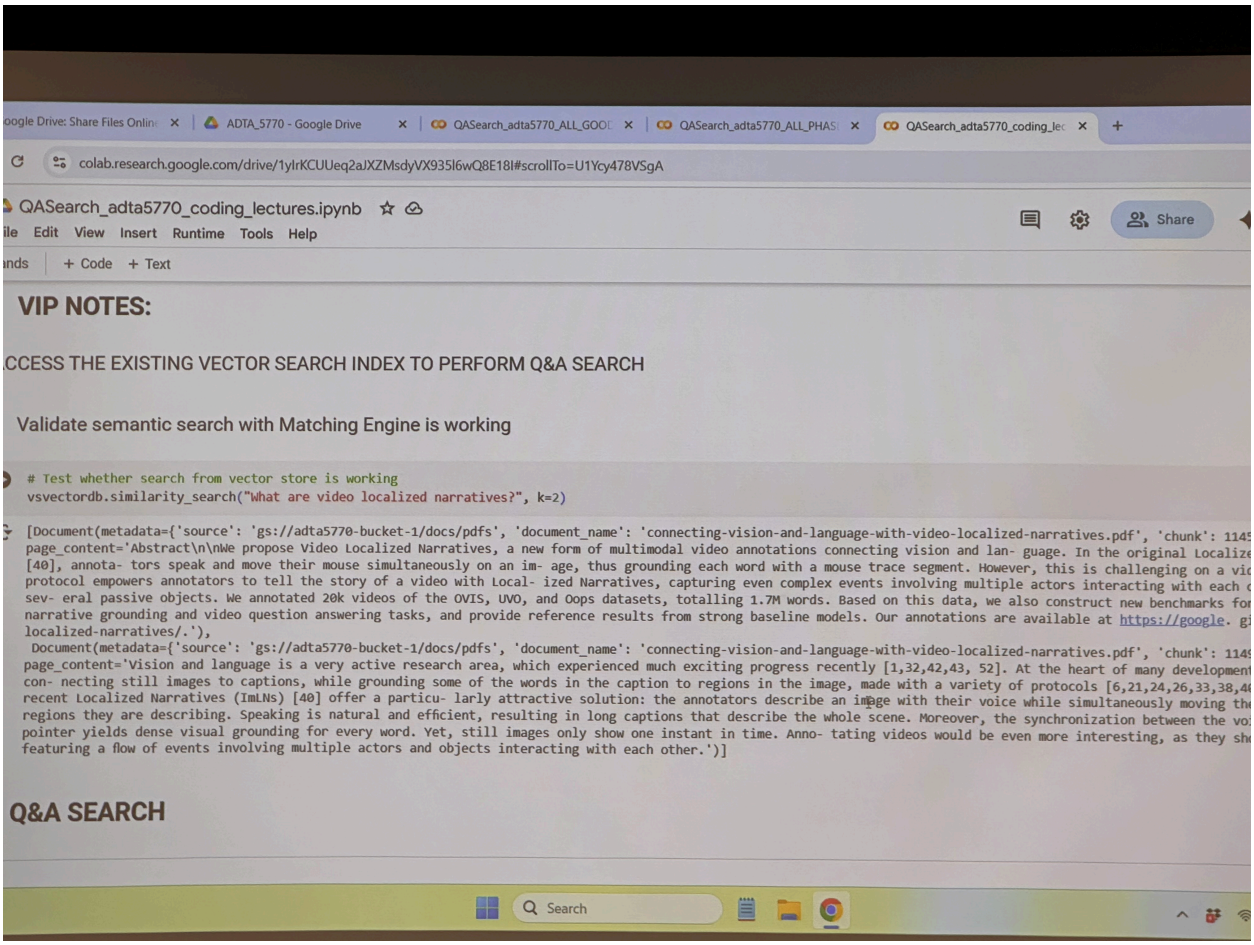

Google Drive: Share Files Onlin
ADTA_5770 - Google Drive
QASearch_adta5770_ALL_GOOE
QASearch_adta5770_ALL_PHAS
QASearch_adta5770_coding_lec
colab.research.google.com/drive/1yIrKCUUeq2aJXZMsdyVX935l6wQ8E18I#scrollTo=U1Ycy478VSgA
QASearch_adta5770_coding_lectures.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text
–) A Vertex AI vector search index, i.e., a vector database, set up as a Vertex AI vector store can be configured as a RetrievalQA.
–) RetrievalQA is a Vertex AI vector search tool specifically used for Q&A Search
First, need to configure a retriever
–) Retriever is used to retrieve response from the vector search index (or vector store)
# Create chain to answer questions
# Number of responses that the system tries to search for
NUMBER_OF_RESULTS = 10
# The distance (similarity search) used as a threshold to search for responses
SEARCH_DISTANCE_THRESHOLD = 0.6
# Expose index to the retriever
retriever = vsvectordb.as_retriever(
search_type="similarity",
search_kwargs={
"k": NUMBER_OF_RESULTS,
"search_distance": SEARCH_DISTANCE_THRESHOLD,
},
filters=None,
)
Customize a prompt template
[ ] template = """SYSTEM: You are an intelligent assistant helping the users with their questions on research papers.
Search

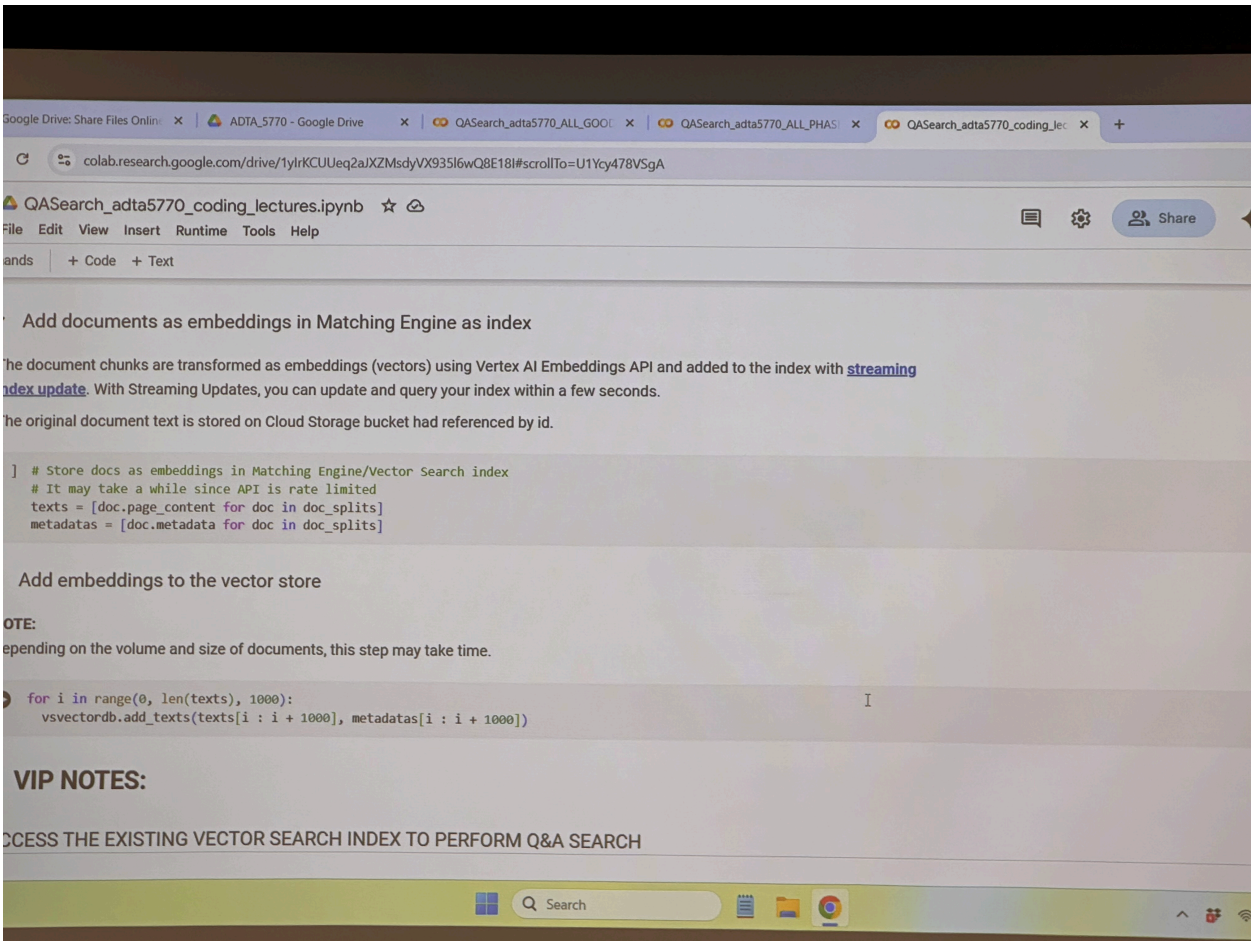

Google Drive: Share Files Online
ADTA_5770 - Google Drive
QASearch_adta5770_ALL_GOOD
QASearch_adta5770_ALL_PHAS
QASearch_adta5770_coding_lec
colab.research.google.com/drive/1ylrKCUUeq2aJXZMsdyVX935l6wQ8E18I#scrollTo=U1Ycy478VSgA
QASearch_adta5770_coding_lectures.ipynb
File Edit View Insert Runtime Tools Help
Share
Commands + Code + Text
–) A Vertex AI vector search index, i.e., a vector database, set up as a Vertex AI vector store can be configured as a RetrievalQA.
–) RetrievalQA is a Vertex AI vector search tool specifically used for Q&A Search
First, need to configure a retriever
–) Retriever is used to retrieve response from the vector search index (or vector store)
# Create chain to answer questions
# Number of responses that the system tries to search for
NUMBER_OF_RESULTS = 10
# The distance (similarity search) used as a threshold to search for responses
SEARCH_DISTANCE_THRESHOLD = 0.6
# Expose index to the retriever
retriever = vsvectordb.as_retriever(
search_type="similarity",
search_kwargs={
"k": NUMBER_OF_RESULTS,
"search_distance": SEARCH_DISTANCE_THRESHOLD,
},
filters=None,
)
Customize a prompt template
[ ] template = """SYSTEM: You are an intelligent assistant helping the users with their questions on research papers.
Search

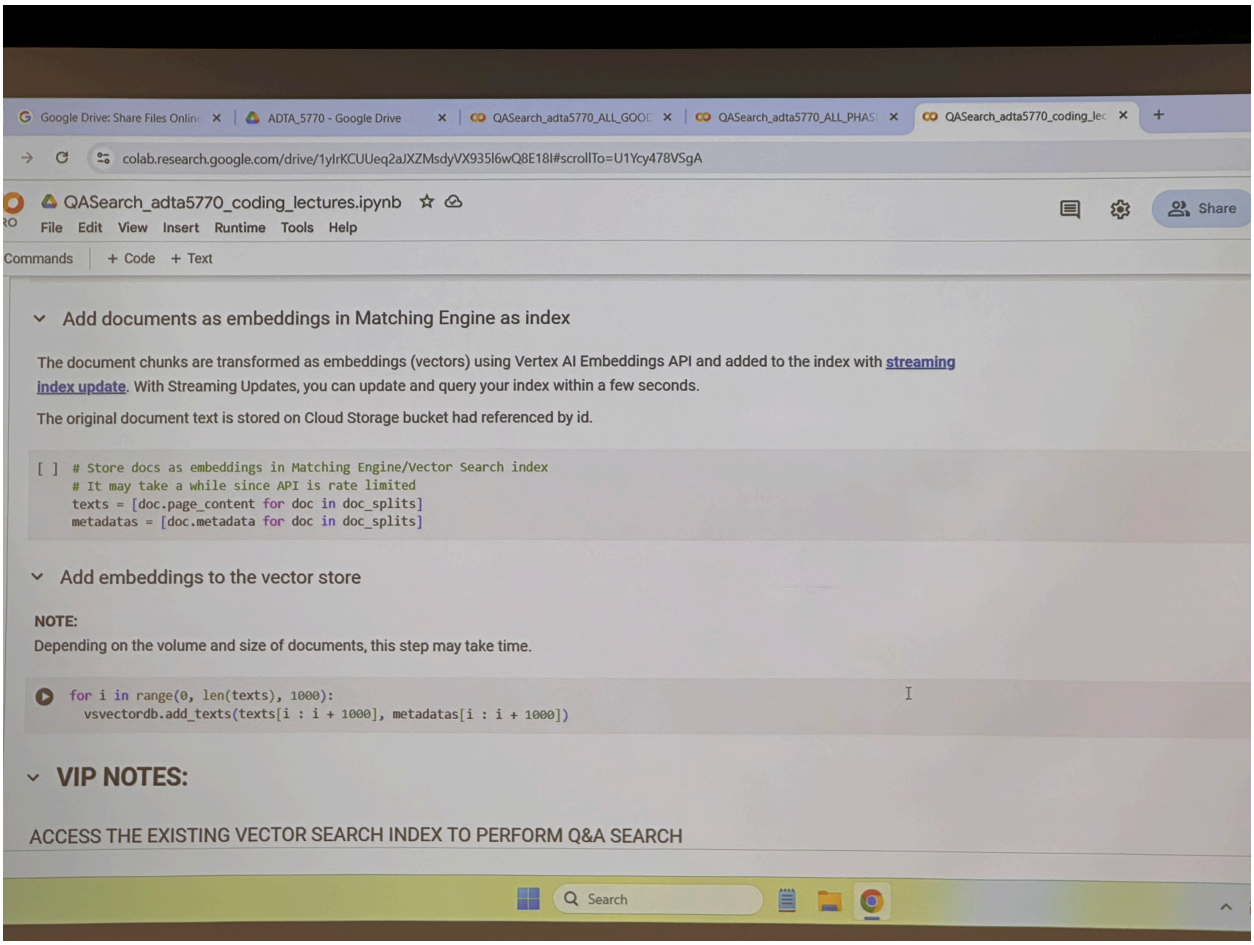

Google Drive: Share Files Onlin
ADTA_5770 - Google Drive
QASearch_adta5770_ALL_GOOI
QASearch_adta5770_ALL_PHAS
QASearch_adta5770_coding_lec
colab.research.google.com/drive/1yIrKCUUeq2aJXZMsdyVX935l6wQ8E18I#scrollTo=U1Ycy478VSgA
QASearch_adta5770_coding_lectures.ipynb
File Edit View Insert Runtime Tools Help
Share
Commands
+ Code + Text
--) A Vertex AI vector search index, i.e., a vector database, set up as a Vertex AI vector store can be configured as a RetrievalQA.
--) RetrievalQA is a Vertex AI vector search tool specifically used for Q&A Search
First, need to configure a retriever
--) Retriever is used to retrieve response from the vector search index (or vector store)
# Create chain to answer questions
# Number of responses that the system tries to search for
NUMBER_OF_RESULTS = 10
# The distance (similarity search) used as a threshold to search for responses
SEARCH_DISTANCE_THRESHOLD = 0.6
# Expose index to the retriever
retriever = vsvectordb.as_retriever(
search_type="similarity",
search_kwargs={
"k": NUMBER_OF_RESULTS,
"search_distance": SEARCH_DISTANCE_THRESHOLD,
},
filters=None,
)
Customize a prompt template
[ ] template = """SYSTEM: You are an intelligent assistant helping the users with their questions on research papers.
Search

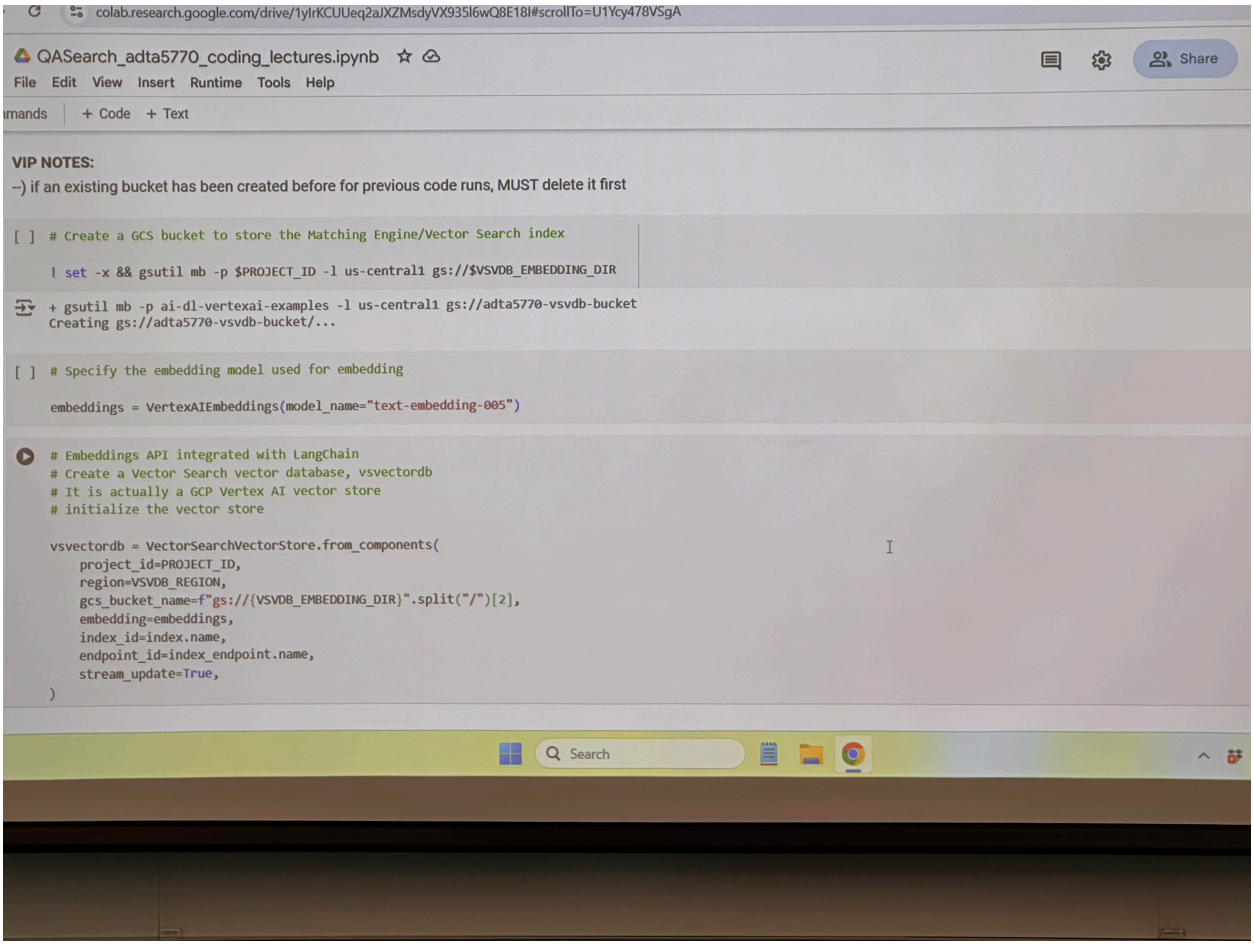

Google Drive: Share Files Onlin
ADTA_5770 - Google Drive
QASearch_adta5770_ALL_GOOD
QASearch_adta5770_ALL_PHAS
QASearch_adta5770_coding_lec
colab.research.google.com/drive/1yIrKCUUeq2aJXZMsdyVX935l6wQ8E18I#scrollTo=U1Ycy478VSgA
QASearch_adta5770_coding_lectures.ipynb
File Edit View Insert Runtime Tools Help
Share
Commands
+ Code
+ Text
Q&A SEARCH
Create an LLM based on an existing LLM
[ ] # Create a model to perform Q&A Search
# This model is based on one of pre-trained LLM
from langchain_google_vertexai import VertexAI
llm = VertexAI(
model_name ="gemini-2.5-pro-exp-03-25",
max_output_tokens=8192,
temperature=0.2,
top_p=0.8,
top_k=40,
verbose=True,
)
Configure an instance of RetrievalQA
VIP NOTES:
--) A Vertex AI vector search index, i.e., a vector database, set up as a Vertex AI vector store can be configured as a RetrievalQA.
--) RetrievalQA is a Vertex AI vector search tool specifically used for Q&A Search
First, need to configure a retriever

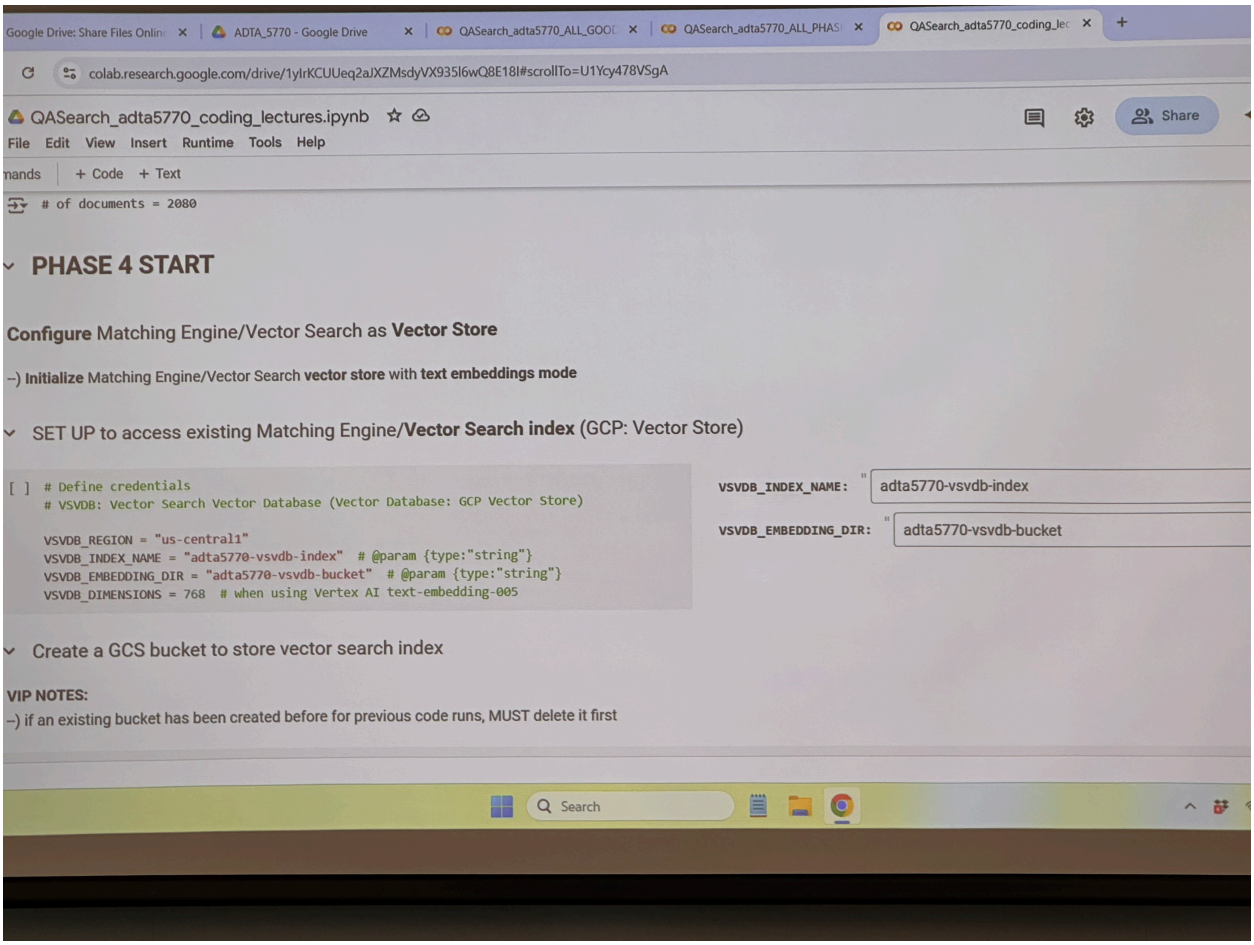

Google Drive: Share Files Onlin
ADTA_5770 - Google Drive
QASearch_adta5770_ALL_GOOI
QASearch_adta5770_ALL_PHAS
QASearch_adta5770_coding_lec
colab.research.google.com/drive/1yIrKCUUeq2aJXZMsdyVX935l6wQ8E18I#scrollTo=U1Ycy478VSgA
QASearch_adta5770_coding_lectures.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text
VIP NOTES:
ACCESS THE EXISTING VECTOR SEARCH INDEX TO PERFORM Q&A SEARCH
Validate semantic search with Matching Engine is working
# Test whether search from vector store is working
vsvectordb.similarity_search("What are video localized narratives?", k=2)
[Document(metadata={'source': 'gs://adta5770-bucket-1/docs/pdfs', 'document_name': 'connecting-vision-and-language-with-video-localized-narratives.pdf', 'chu
page_content='Abstract\n\nWe propose Video Localized Narratives, a new form of multimodal video annotations connecting vision and lan- guage. In the original
[40], annota- tors speak and move their mouse simultaneously on an im- age, thus grounding each word with a mouse trace segment. However, this is challenging
protocol empowers annotators to tell the story of a video with Local- ized Narratives, capturing even complex events involving multiple actors interacting wi
sev- eral passive objects. We annotated 20k videos of the OVIS, UVO, and Oops datasets, totalling 1.7M words. Based on this data, we also construct new bench
narrative grounding and video question answering tasks, and provide reference results from strong baseline models. Our annotations are available at https://g
localized-narratives/.'),
Document(metadata={'source': 'gs://adta5770-bucket-1/docs/pdfs', 'document_name': 'connecting-vision-and-language-with-video-localized-narratives.pdf', 'chu
page_content='Vision and language is a very active research area, which experienced much exciting progress recently [1,32,42,43, 52]. At the heart of many de
con- necting still images to captions, while grounding some of the words in the caption to regions in the image, made with a variety of protocols [6,21,24,26
recent Localized Narratives (ImLNs) [40] offer a particu- larly attractive solution: the annotators describe an image with their voice while simultaneously m
regions they are describing. Speaking is natural and efficient, resulting in long captions that describe the whole scene. Moreover, the synchronization betwee
pointer yields dense visual grounding for every word. Yet, still images only show one instant in time. Anno- tating videos would be even more interesting, as
featuring a flow of events involving multiple actors and objects interacting with each other.')]
Q&A SEARCH
Search

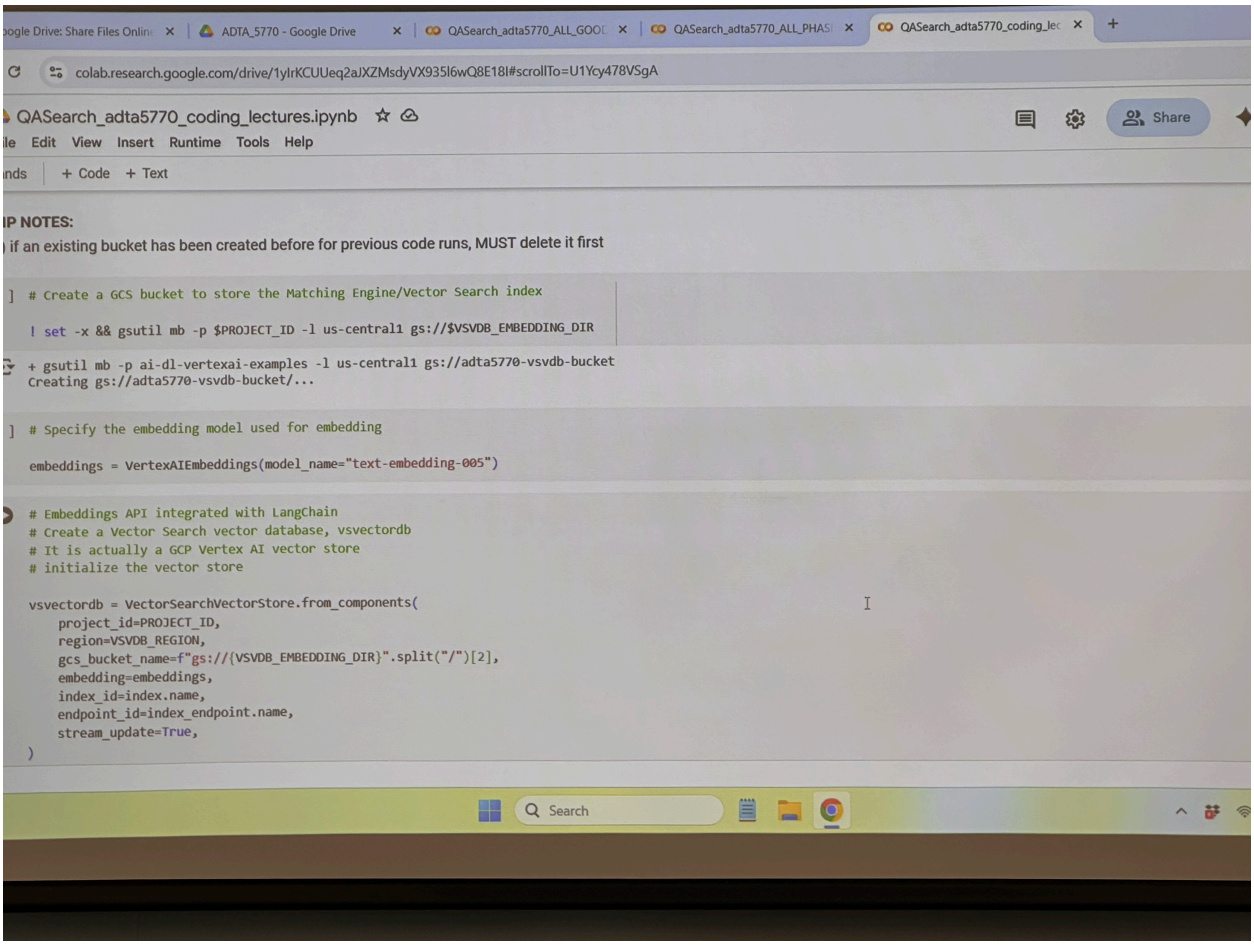

ADTA_5770 - Google Drive
QASearch_adta5770_coding_lec
colab.research.google.com/drive/1ylrKCUUeq2aJXZMsdyVX935l6wQ8E18I#scrollTo=U1Ycy478VSgA
QASearch_adta5770_coding_lectures.ipynb
File Edit View Insert Runtime Tools Help
Share
+ Code + Text
VIP NOTES:
CCESS THE EXISTING VECTOR SEARCH INDEX TO PERFORM Q&A SEARCH
Validate semantic search with Matching Engine is working
# Test whether search from vector store is working
vsvectordb.similarity_search("What are video localized narratives?", k=2)
[Document(metadata={'source': 'gs://adta5770-bucket-1/docs/pdfs', 'document_name': 'connecting-vision-and-language-with-video-localized-narratives.pdf', 'chunk': 114
page_content='Abstract\n\nWe propose Video Localized Narratives, a new form of multimodal video annotations connecting vision and lan- guage. In the original Localize
[40], annota- tors speak and move their mouse simultaneously on an im- age, thus grounding each word with a mouse trace segment. However, this is challenging on a vi
protocol empowers annotators to tell the story of a video with Local- ized Narratives, capturing even complex events involving multiple actors interacting with each
sev- eral passive objects. We annotated 20k videos of the OVIS, UVO, and Oops datasets, totalling 1.7M words. Based on this data, we also construct new benchmarks fo
narrative grounding and video question answering tasks, and provide reference results from strong baseline models. Our annotations are available at https://google. g
localized-narratives/.'),
Document(metadata={'source': 'gs://adta5770-bucket-1/docs/pdfs', 'document_name': 'connecting-vision-and-language-with-video-localized-narratives.pdf', 'chunk': 114
page_content='Vision and language is a very active research area, which experienced much exciting progress recently [1,32,42,43, 52]. At the heart of many developmen
con- necting still images to captions, while grounding some of the words in the caption to regions in the image, made with a variety of protocols [6,21,24,26,33,38,4
recent Localized Narratives (ImLNs) [40] offer a particu- larly attractive solution: the annotators describe an image with their voice while simultaneously moving th
regions they are describing. Speaking is natural and efficient, resulting in long captions that describe the whole scene. Moreover, the synchronization between the vo
pointer yields dense visual grounding for every word. Yet, still images only show one instant in time. Anno- tating videos would be even more interesting, as they sh
featuring a flow of events involving multiple actors and objects interacting with each other.')]
Q&A SEARCH
Search

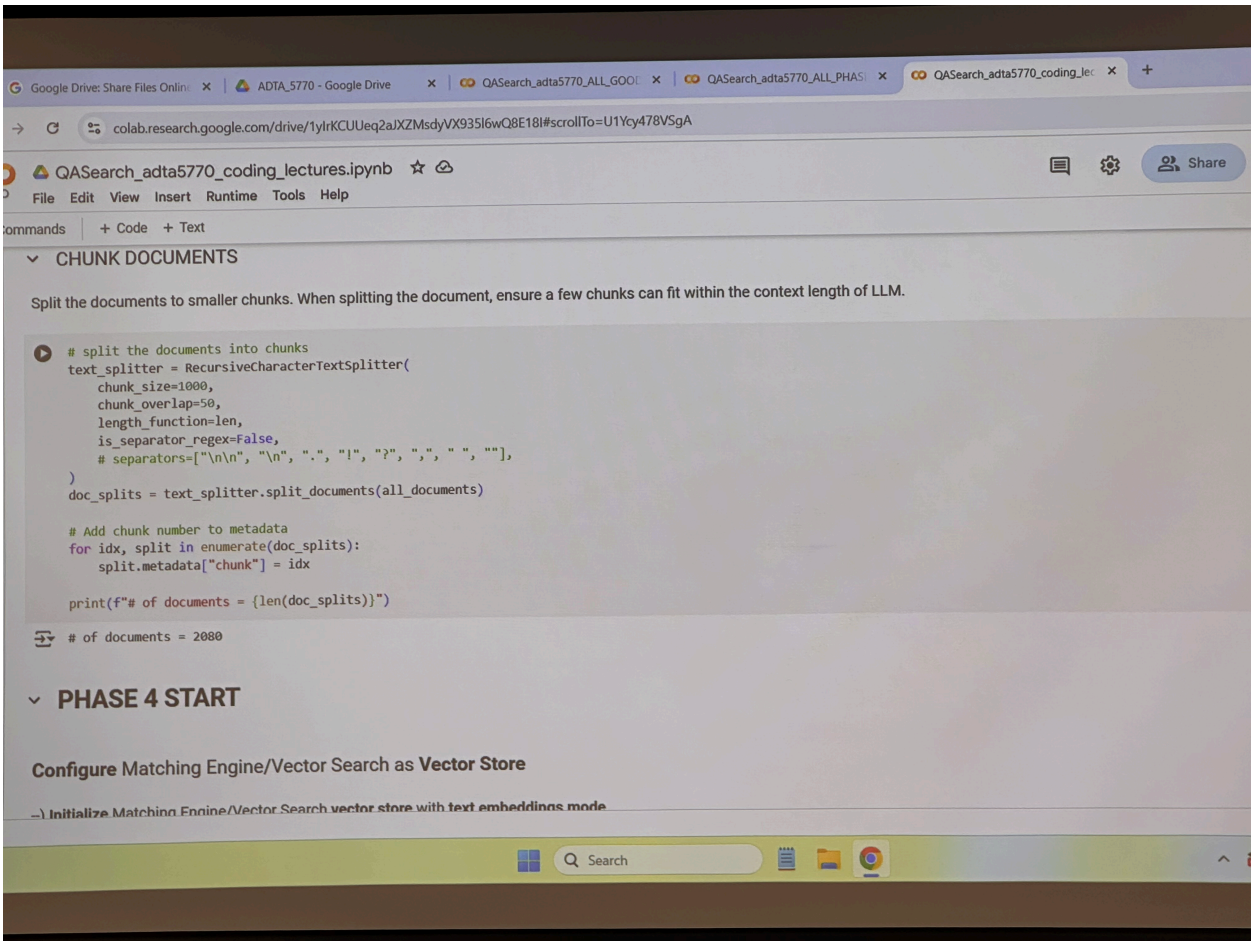

ADTA_5770 - Google Drive
colab.research.google.com/drive/1yIrKCUUeq2aJXZMsdyVX935l6wQ8E18I#scrollTo=U1Ycy478VSgA
QASearch_adta5770_coding_lectures.ipynb
File Edit View Insert Runtime Tools Help
Share
+ Code + Text
Add documents as embeddings in Matching Engine as index
The document chunks are transformed as embeddings (vectors) using Vertex AI Embeddings API and added to the index with streaming
ndex update. With Streaming Updates, you can update and query your index within a few seconds.
The original document text is stored on Cloud Storage bucket had referenced by id.
# Store docs as embeddings in Matching Engine/Vector Search index
# It may take a while since API is rate limited
texts = [doc.page_content for doc in doc_splits]
metadatas = [doc.metadata for doc in doc_splits]
Add embeddings to the vector store
OTE:
epending on the volume and size of documents, this step may take time.
for i in range(0, len(texts), 1000):
vsvectordb.add_texts(texts[i : i + 1000], metadatas[i : i + 1000])
VIP NOTES:
CCESS THE EXISTING VECTOR SEARCH INDEX TO PERFORM Q&A SEARCH
Search

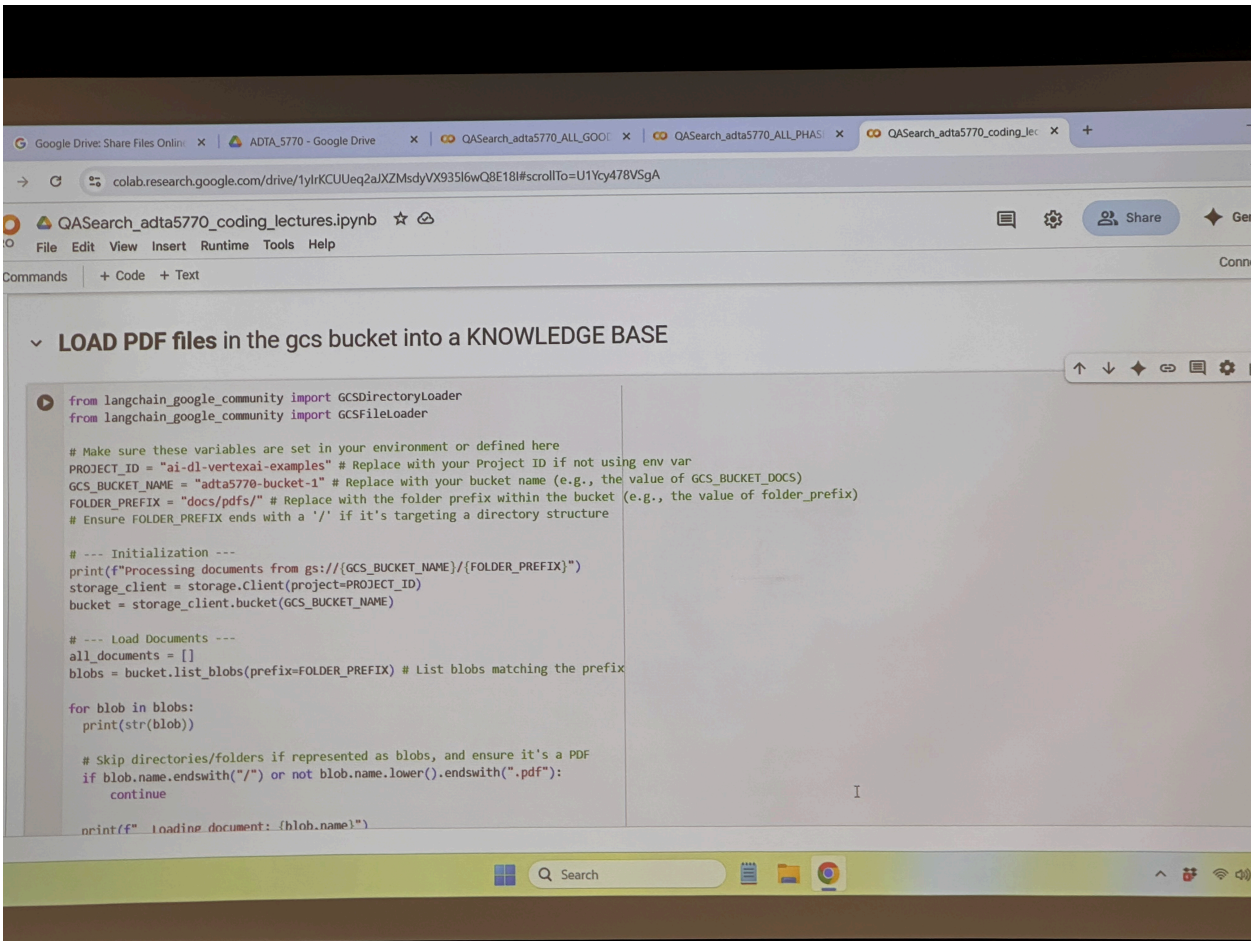

Google Drive: Share Files Onlin
ADTA_5770 - Google Drive
QASearch_adta5770_ALL_GOOD
QASearch_adta5770_ALL_PHAS
QASearch_adta5770_coding_lec
colab.research.google.com/drive/1yIrKCUUeq2aJXZMsdyVX935l6wQ8E18I#scrollTo=U1Ycy478VSgA
QASearch_adta5770_coding_lectures.ipynb
File Edit View Insert Runtime Tools Help
Share
Commands + Code + Text
Add documents as embeddings in Matching Engine as index
The document chunks are transformed as embeddings (vectors) using Vertex AI Embeddings API and added to the index with streaming index update. With Streaming Updates, you can update and query your index within a few seconds.
The original document text is stored on Cloud Storage bucket had referenced by id.
[ ] # Store docs as embeddings in Matching Engine/Vector Search index
# It may take a while since API is rate limited
texts = [doc.page_content for doc in doc_splits]
metadatas = [doc.metadata for doc in doc_splits]
Add embeddings to the vector store
NOTE:
Depending on the volume and size of documents, this step may take time.
for i in range(0, len(texts), 1000):
vsvectordb.add_texts(texts[i : i + 1000], metadatas[i : i + 1000])
VIP NOTES:
ACCESS THE EXISTING VECTOR SEARCH INDEX TO PERFORM Q&A SEARCH
Search

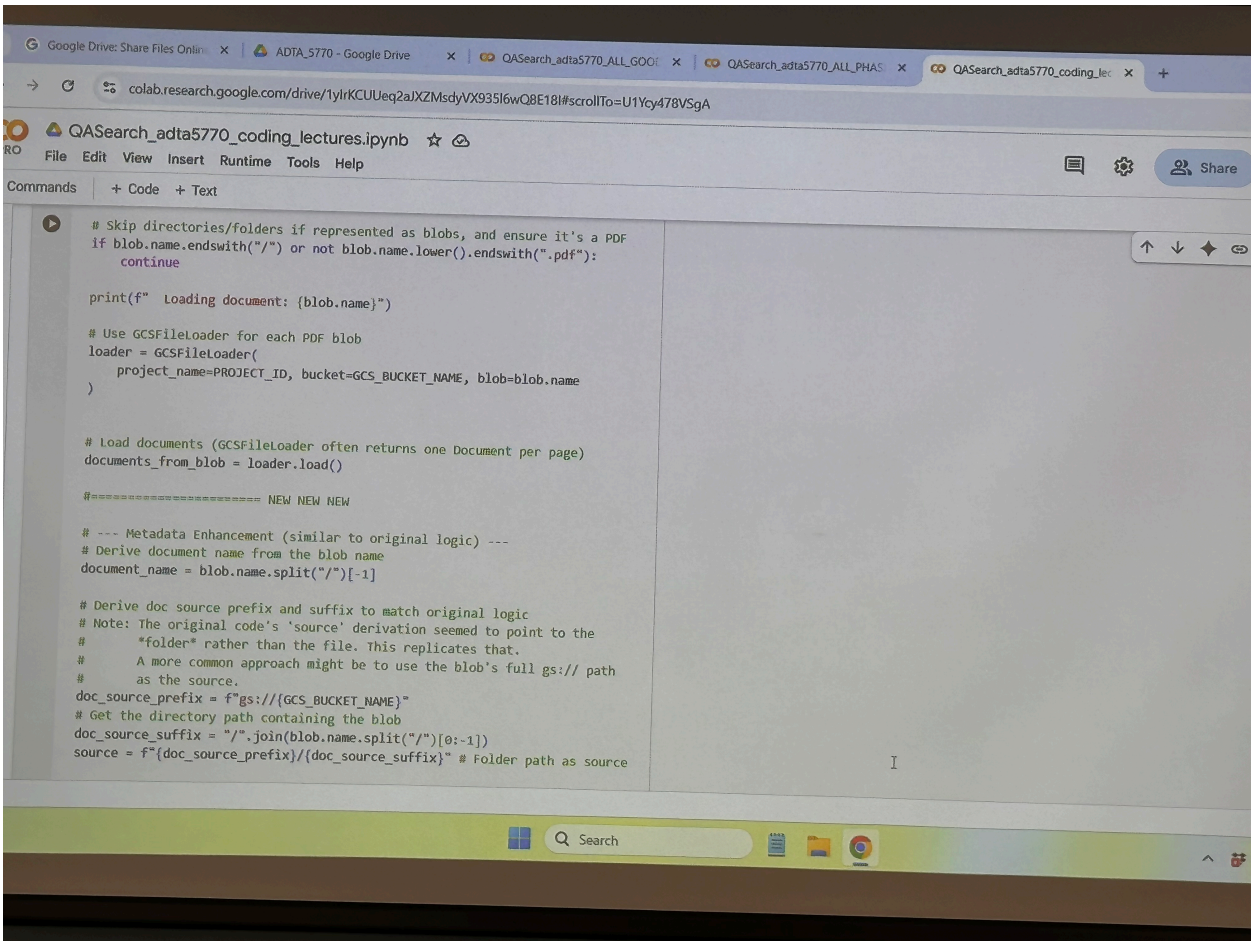

colab.research.google.com/drive/1yIrKCUUeq2aJXZMsdyVX935l6wQ8E18I#scrollTo=U1Ycy478VSgA
QASearch_adta5770_coding_lectures.ipynb
File Edit View Insert Runtime Tools Help
Share
+ Code + Text
VIP NOTES:
--) if an existing bucket has been created before for previous code runs, MUST delete it first
[ ] # Create a GCS bucket to store the Matching Engine/Vector Search index
! set -x && gsutil mb -p $PROJECT_ID -l us-central1 gs://$VSVDB_EMBEDDING_DIR
+ gsutil mb -p ai-dl-vertexai-examples -l us-central1 gs://adta5770-vsvdb-bucket
Creating gs://adta5770-vsvdb-bucket/...
[ ] # Specify the embedding model used for embedding
embeddings = VertexAIEmbeddings(model_name="text-embedding-005")
# Embeddings API integrated with LangChain
# Create a Vector Search vector database, vsvectordb
# It is actually a GCP Vertex AI vector store
# initialize the vector store
vsvectordb = VectorSearchVectorStore.from_components(
project_id=PROJECT_ID,
region=VSVDB_REGION,
gcs_bucket_name=f"gs://{VSVDB_EMBEDDING_DIR}".split("/")[2],
embedding=embeddings,
index_id=index.name,
endpoint_id=index_endpoint.name,
stream_update=True,
)
Search

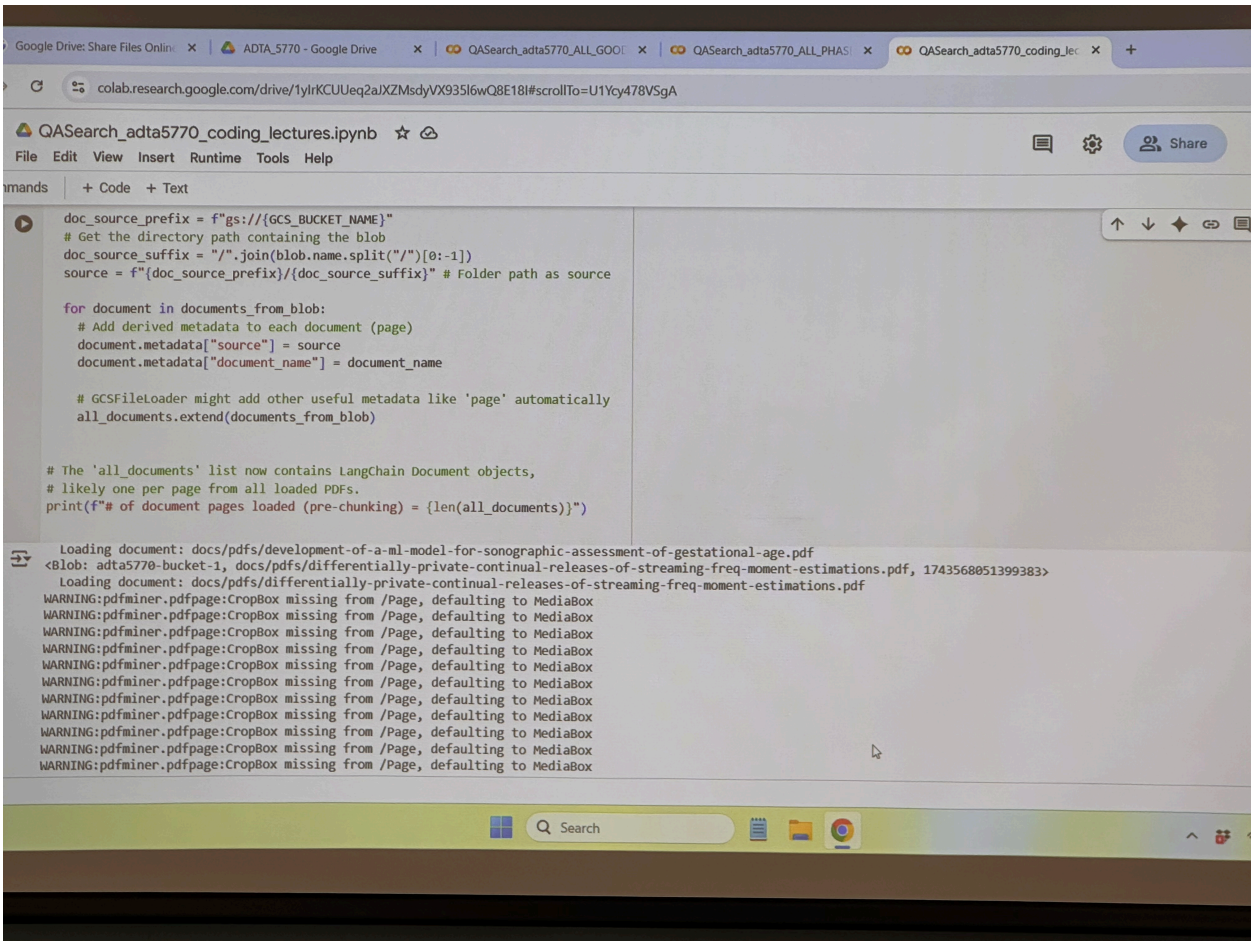

Google Drive: Share Files Onlin
ADTA_5770 - Google Drive
QASearch_adta5770_ALL_GOOD
QASearch_adta5770_ALL_PHAS
QASearch_adta5770_coding_lec
colab.research.google.com/drive/1yIrKCUUeq2aJXZMsdyVX935l6wQ8E18I#scrollTo=U1Ycy478VSgA
QASearch_adta5770_coding_lectures.ipynb
File Edit View Insert Runtime Tools Help
Share
nands
+ Code + Text
# of documents = 2080
PHASE 4 START
Configure Matching Engine/Vector Search as Vector Store
–) Initialize Matching Engine/Vector Search vector store with text embeddings mode
SET UP to access existing Matching Engine/Vector Search index (GCP: Vector Store)
[ ] # Define credentials
# VSVDB: Vector Search Vector Database (Vector Database: GCP Vector Store)
VSVDB_REGION = "us-central1"
VSVDB_INDEX_NAME = "adta5770-vsvdb-index" # @param {type:"string"}
VSVDB_EMBEDDING_DIR = "adta5770-vsvdb-bucket" # @param {type:"string"}
VSVDB_DIMENSIONS = 768 # when using Vertex AI text-embedding-005
VSVDB_INDEX_NAME:
adta5770-vsvdb-index
VSVDB_EMBEDDING_DIR:
adta5770-vsvdb-bucket
Create a GCS bucket to store vector search index
VIP NOTES:
–) if an existing bucket has been created before for previous code runs, MUST delete it first
Search

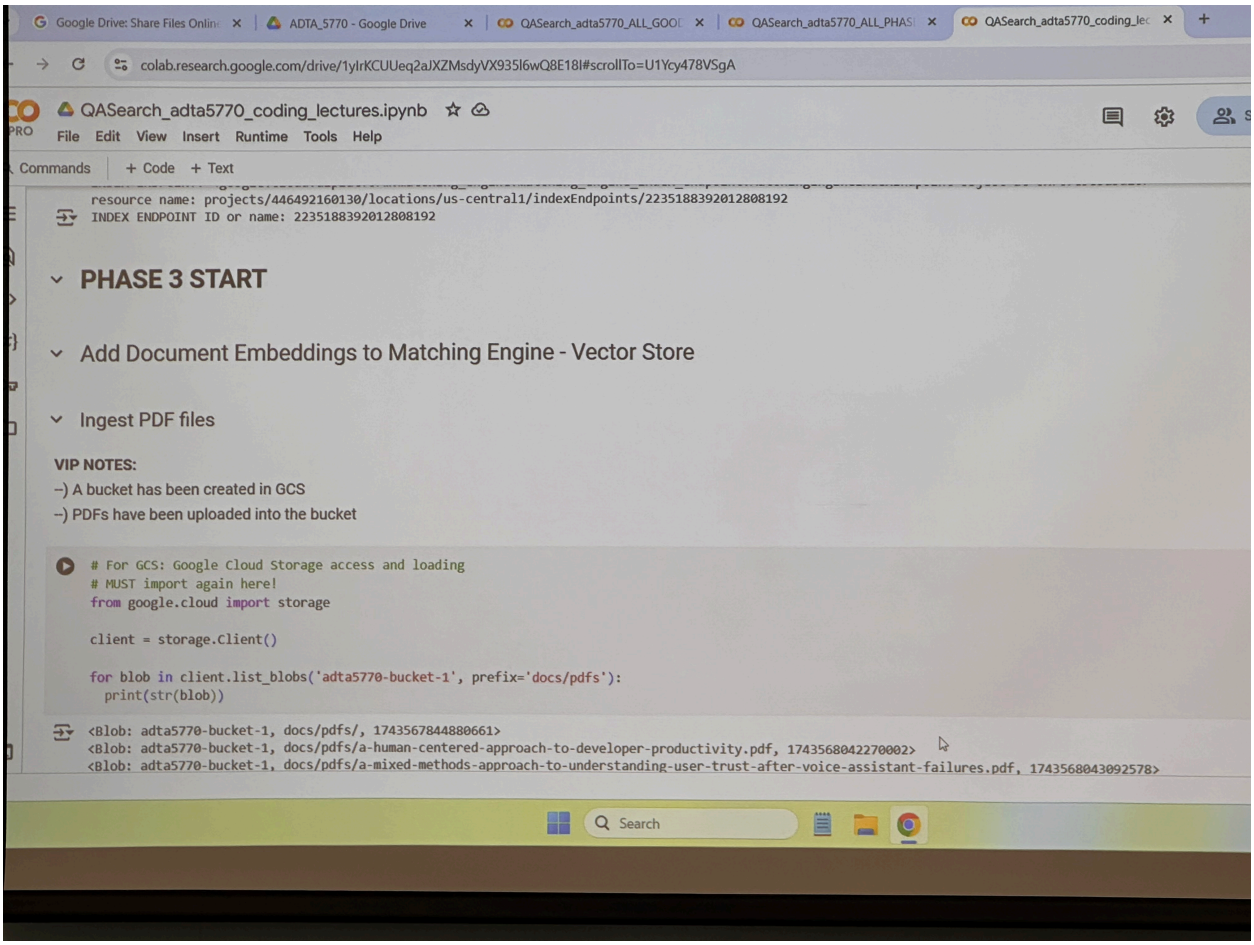

ADTA_5770 - Google Drive
QASearch_adta5770_coding_lec
colab.research.google.com/drive/1ylrKCUUeq2aJXZMsdyVX935l6wQ8E18I#scrollTo=U1Ycy478VSgA
QASearch_adta5770_coding_lectures.ipynb
File Edit View Insert Runtime Tools Help
+ Code + Text
IP NOTES:
if an existing bucket has been created before for previous code runs, MUST delete it first
Create a GCS bucket to store the Matching Engine/Vector Search index
! set -x && gsutil mb -p $PROJECT_ID -l us-central1 gs://$VSVDB_EMBEDDING_DIR
+ gsutil mb -p ai-dl-vertexai-examples -l us-central1 gs://adta5770-vsvdb-bucket
Creating gs://adta5770-vsvdb-bucket/...
# Specify the embedding model used for embedding
embeddings = VertexAIEmbeddings(model_name="text-embedding-005")
# Embeddings API integrated with LangChain
# Create a Vector Search vector database, vsvectordb
# It is actually a GCP Vertex AI vector store
# initialize the vector store
vsvectordb = VectorSearchVectorStore.from_components(
project_id=PROJECT_ID,
region=VSVDB_REGION,
gcs_bucket_name=f"gs://{VSVDB_EMBEDDING_DIR}".split("/")[2],
embedding=embeddings,
index_id=index.name,
endpoint_id=index_endpoint.name,
stream_update=True,
)
Search

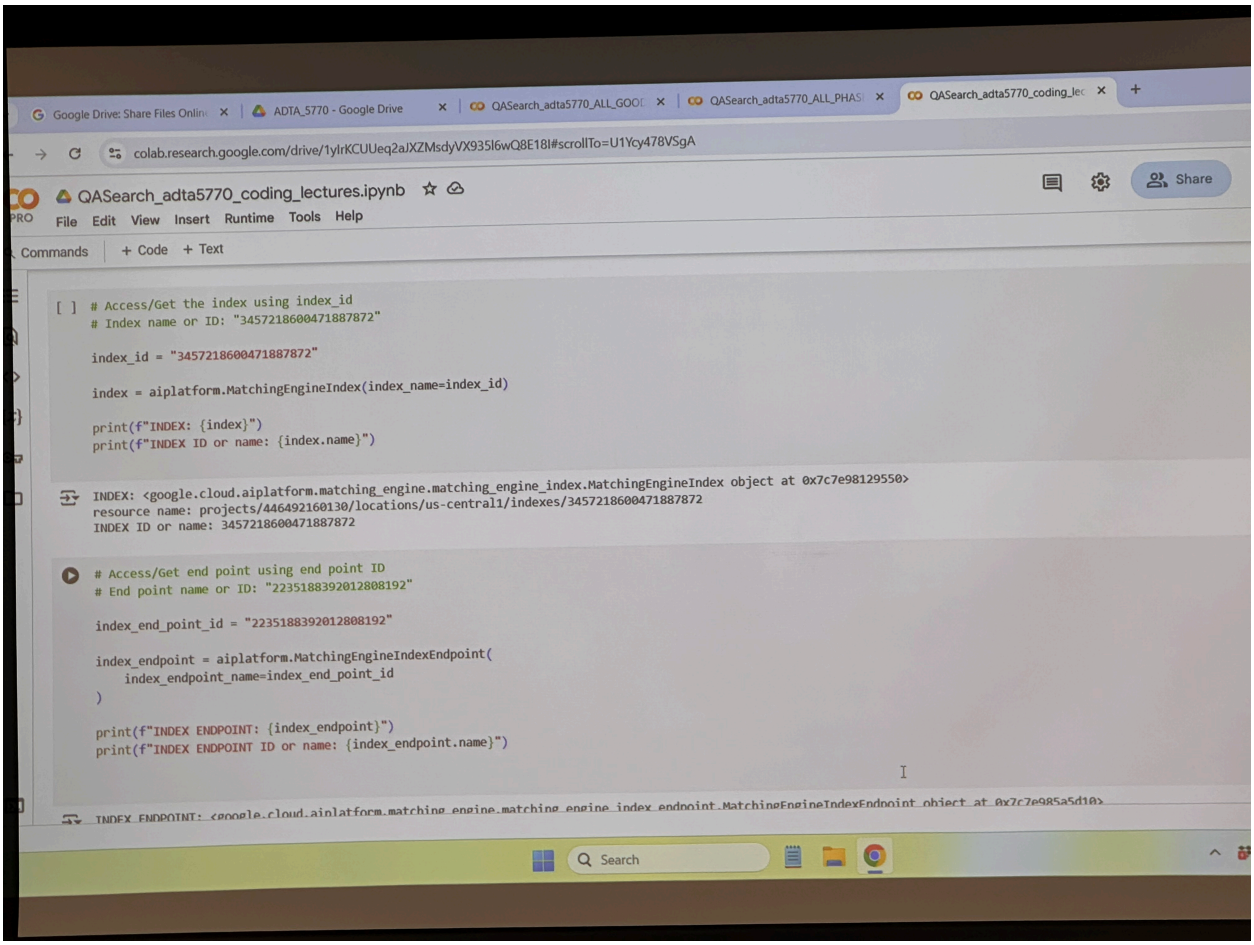

Google Drive: Share Files Onlin
ADTA_5770 - Google Drive
QASearch_adta5770_ALL_GOOD
QASearch_adta5770_ALL_PHAS
QASearch_adta5770_coding_lec
colab.research.google.com/drive/1yIrKCUUeq2aJXZMsdyVX935l6wQ8E18I#scrollTo=U1Ycy478VSgA
QASearch_adta5770_coding_lectures.ipynb
File Edit View Insert Runtime Tools Help
Share
ommands
+ Code + Text
CHUNK DOCUMENTS
Split the documents to smaller chunks. When splitting the document, ensure a few chunks can fit within the context length of LLM.
# split the documents into chunks
text_splitter = RecursiveCharacterTextSplitter(
chunk_size=1000,
chunk_overlap=50,
length_function=len,
is_separator_regex=False,
# separators=["\n\n", "\n", ".", "!", "?", ",", " ", ""],
)
doc_splits = text_splitter.split_documents(all_documents)
# Add chunk number to metadata
for idx, split in enumerate(doc_splits):
split.metadata["chunk"] = idx
print(f"# of documents = {len(doc_splits)}")
# of documents = 2080
PHASE 4 START
Configure Matching Engine/Vector Search as Vector Store
Search

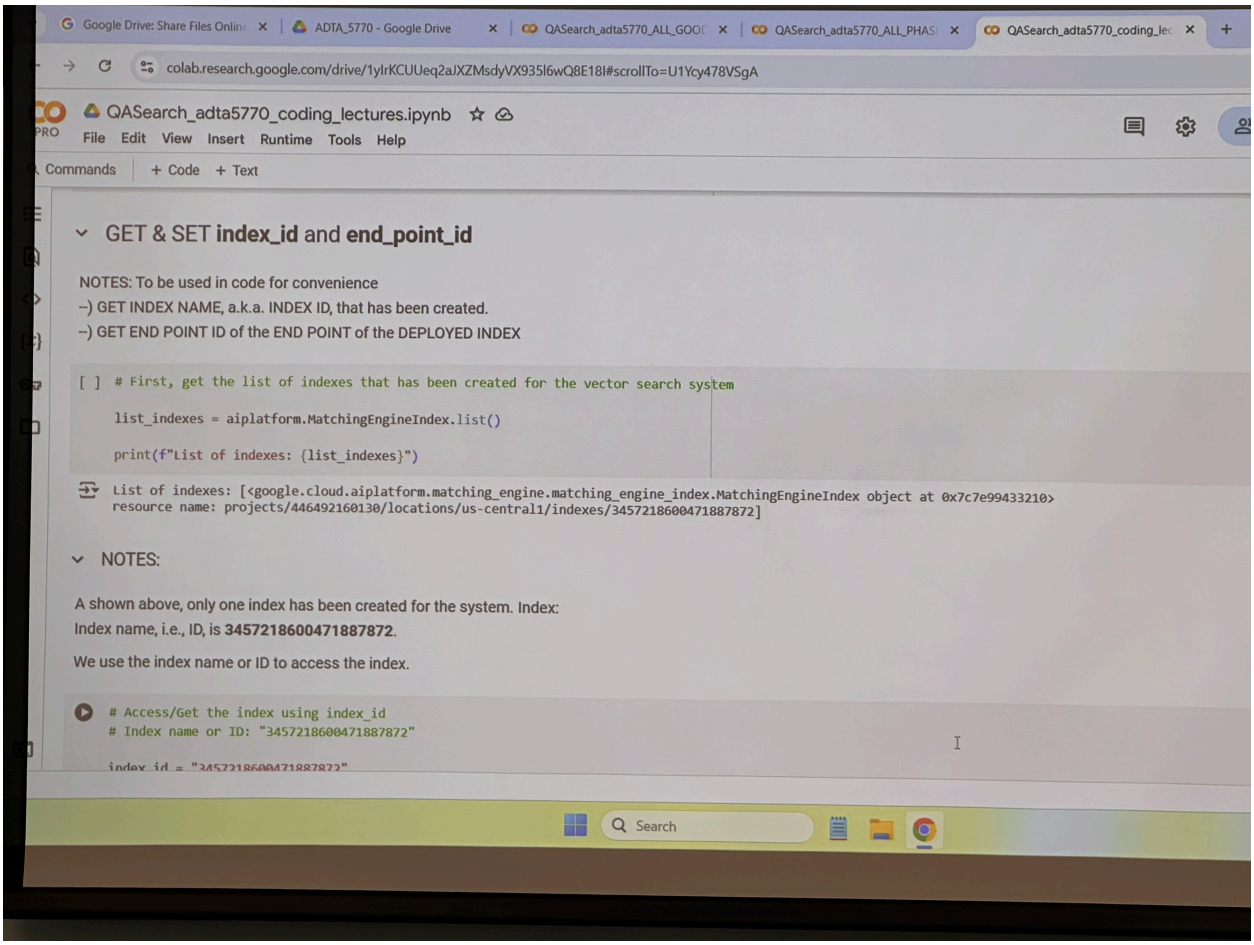

Google Drive: Share Files Onlin
ADTA_5770 - Google Drive
QASearch_adta5770_ALL_GOOI
QASearch_adta5770_ALL_PHAS
QASearch_adta5770_coding_lec
colab.research.google.com/drive/1yIrKCUUeq2aJXZMsdyVX935l6wQ8E18l#scrollTo=U1Ycy478VSgA
QASearch_adta5770_coding_lectures.ipynb
File Edit View Insert Runtime Tools Help
Share
Commands
+ Code + Text
LOAD PDF files in the gcs bucket into a KNOWLEDGE BASE
from langchain_google_community import GCSDirectoryLoader
from langchain_google_community import GCSFileLoader
# Make sure these variables are set in your environment or defined here
PROJECT_ID = "ai-dl-vertexai-examples" # Replace with your Project ID if not using env var
GCS_BUCKET_NAME = "adta5770-bucket-1" # Replace with your bucket name (e.g., the value of GCS_BUCKET_DOCS)
FOLDER_PREFIX = "docs/pdfs/" # Replace with the folder prefix within the bucket (e.g., the value of folder_prefix)
# Ensure FOLDER_PREFIX ends with a '/' if it's targeting a directory structure
# --- Initialization ---
print(f"Processing documents from gs://{GCS_BUCKET_NAME}/{FOLDER_PREFIX}")
storage_client = storage.Client(project=PROJECT_ID)
bucket = storage_client.bucket(GCS_BUCKET_NAME)
# --- Load Documents ---
all_documents = []
blobs = bucket.list_blobs(prefix=FOLDER_PREFIX) # List blobs matching the prefix
for blob in blobs:
print(str(blob))
# Skip directories/folders if represented as blobs, and ensure it's a PDF
if blob.name.endswith("/") or not blob.name.lower().endswith(".pdf"):
continue
print(f" Loading document: {blob.name}")
Search

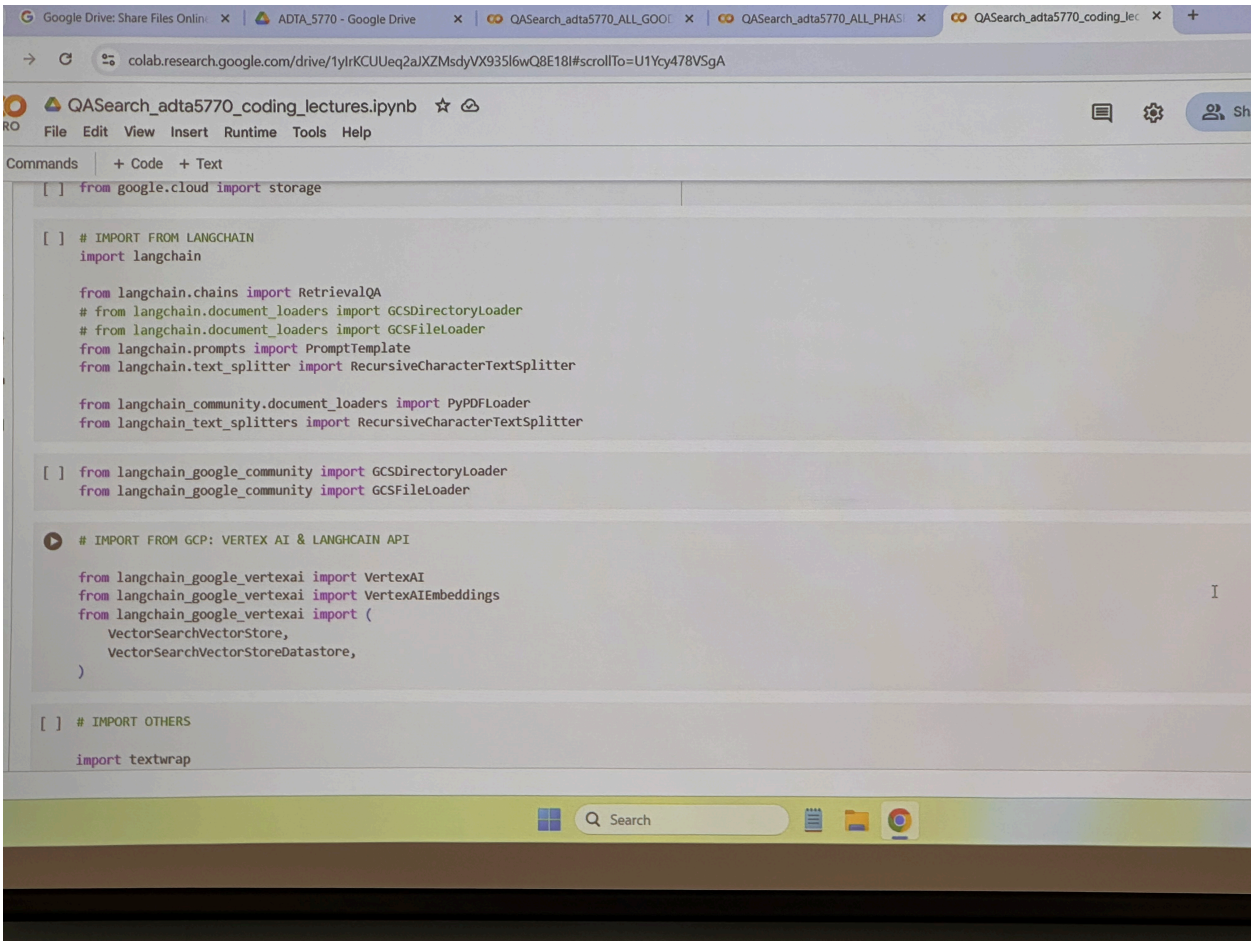

Google Drive: Share Files Onlin
ADTA_5770 - Google Drive
QASearch_adta5770_ALL_GOO
QASearch_adta5770_ALL_PHAS
QASearch_adta5770_coding_lec
colab.research.google.com/drive/1ylrKCUUeq2aJXZMsdyVX935l6wQ8E18l#scrollTo=U1Ycy478VSgA
QASearch_adta5770_coding_lectures.ipynb
File Edit View Insert Runtime Tools Help
Share
Commands
+ Code + Text
# Skip directories/folders if represented as blobs, and ensure it's a PDF
if blob.name.endswith("/") or not blob.name.lower().endswith(".pdf"):
continue
print(f" Loading document: {blob.name}")
# Use GCSFileLoader for each PDF blob
loader = GCSFileLoader(
project_name=PROJECT_ID, bucket=GCS_BUCKET_NAME, blob=blob.name
)
# Load documents (GCSFileLoader often returns one Document per page)
documents_from_blob = loader.load()
#====================== NEW NEW NEW
# --- Metadata Enhancement (similar to original logic) ---
# Derive document name from the blob name
document_name = blob.name.split("/")[-1]
# Derive doc source prefix and suffix to match original logic
# Note: The original code's 'source' derivation seemed to point to the
# *folder* rather than the file. This replicates that.
# A more common approach might be to use the blob's full gs:// path
# as the source.
doc_source_prefix = f"gs://{GCS_BUCKET_NAME}"
# Get the directory path containing the blob
doc_source_suffix = "/".join(blob.name.split("/")[0:-1])
source = f"{doc_source_prefix}/{doc_source_suffix}" # Folder path as source
Search

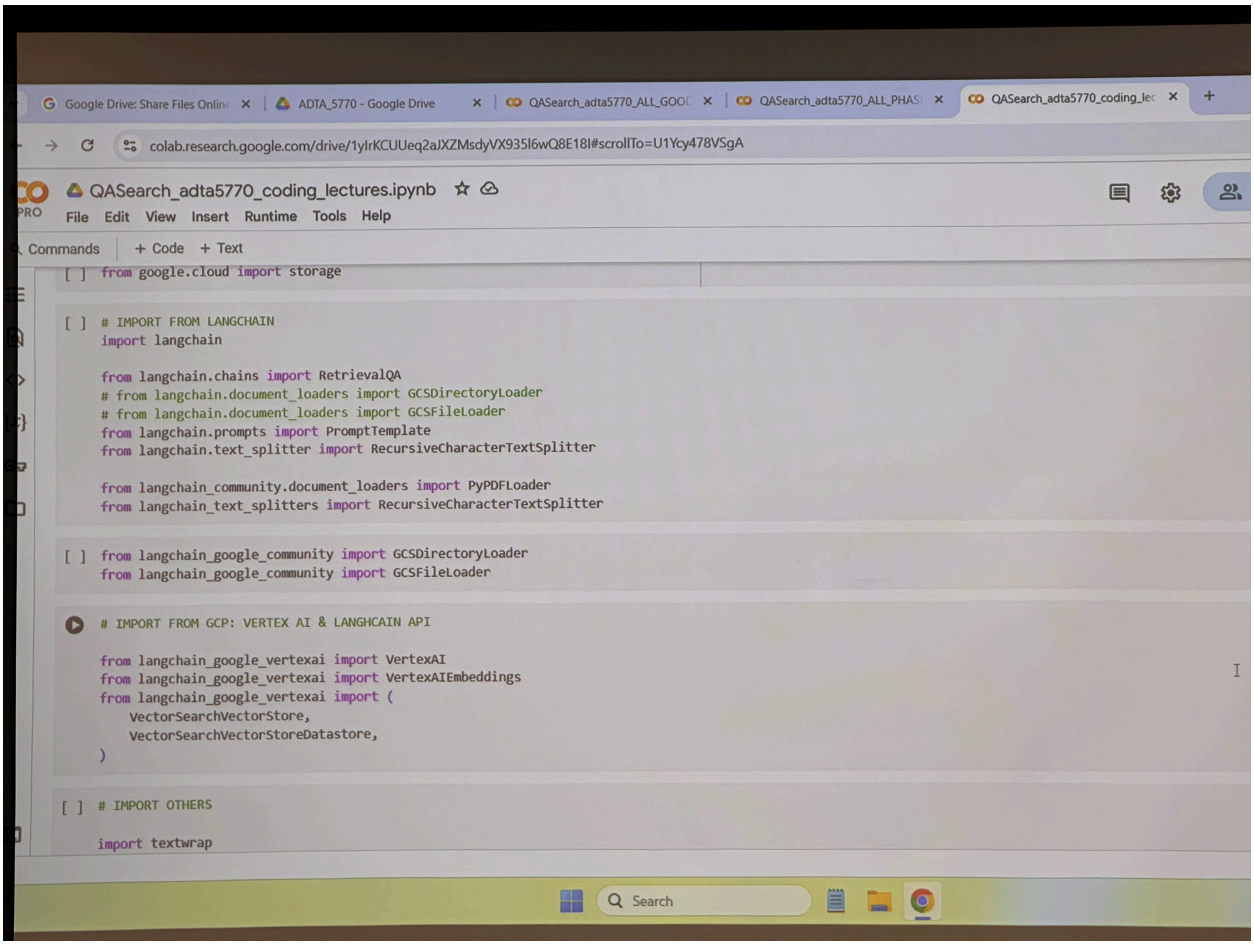

Google Drive: Share Files Onlin
ADTA_5770 - Google Drive
QASearch_adta5770_ALL_GOOD
QASearch_adta5770_ALL_PHAS
QASearch_adta5770_coding_lec
colab.research.google.com/drive/1yIrKCUUeq2aJXZMsdyVX935I6wQ8E18I#scrollTo=U1Ycy478VSgA
QASearch_adta5770_coding_lectures.ipynb
File Edit View Insert Runtime Tools Help
Share
+ Code + Text
doc_source_prefix = f"gs://{GCS_BUCKET_NAME}"
# Get the directory path containing the blob
doc_source_suffix = "/".join(blob.name.split("/")[0:-1])
source = f"{doc_source_prefix}/{doc_source_suffix}" # Folder path as source
for document in documents_from_blob:
# Add derived metadata to each document (page)
document.metadata["source"] = source
document.metadata["document_name"] = document_name
# GCSFileLoader might add other useful metadata like 'page' automatically
all_documents.extend(documents_from_blob)
# The 'all_documents' list now contains LangChain Document objects,
# likely one per page from all loaded PDFs.
print(f"# of document pages loaded (pre-chunking) = {len(all_documents)}")
Loading document: docs/pdfs/development-of-a-ml-model-for-sonographic-assessment-of-gestational-age.pdf
<Blob: adta5770-bucket-1, docs/pdfs/differentially-private-continual-releases-of-streaming-freq-moment-estimations.pdf, 1743568051399383>
Loading document: docs/pdfs/differentially-private-continual-releases-of-streaming-freq-moment-estimations.pdf
WARNING:pdfminer.pdfpage:CropBox missing from /Page, defaulting to MediaBox
Search

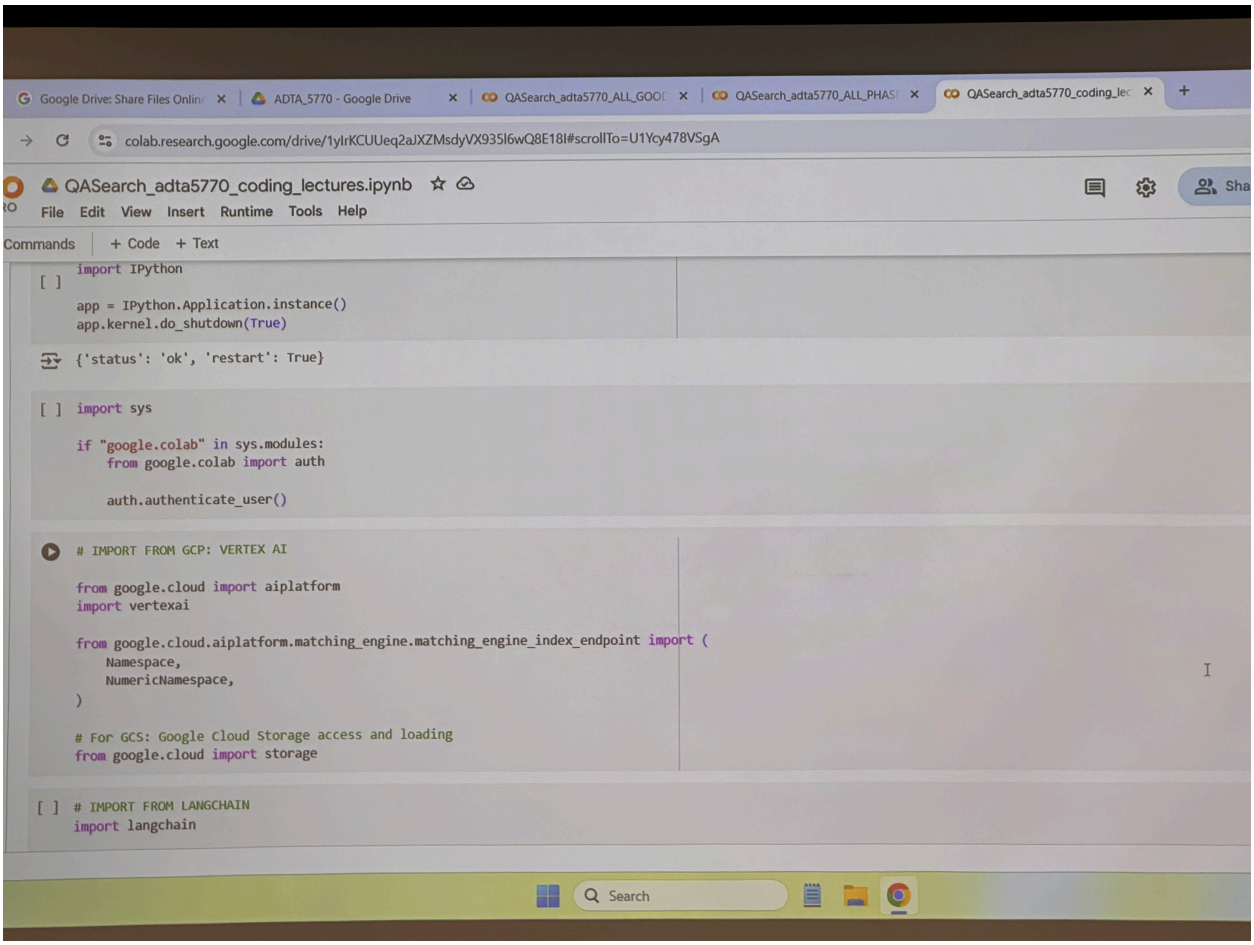

Google Drive: Share Files Onlin
ADTA_5770 - Google Drive
QASearch_adta5770_ALL_GOOD
QASearch_adta5770_ALL_PHAS
QASearch_adta5770_coding_lec
colab.research.google.com/drive/1yIrKCUUeq2aJXZMsdyVX935l6wQ8E18I#scrollTo=U1Ycy478VSgA
QASearch_adta5770_coding_lectures.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text
resource name: projects/446492160130/locations/us-central1/indexEndpoints/2235188392012808192
INDEX ENDPOINT ID or name: 2235188392012808192
PHASE 3 START
Add Document Embeddings to Matching Engine - Vector Store
Ingest PDF files
VIP NOTES:
–) A bucket has been created in GCS
–) PDFs have been uploaded into the bucket
# For GCS: Google Cloud Storage access and loading
# MUST import again here!
from google.cloud import storage
client = storage.Client()
for blob in client.list_blobs('adta5770-bucket-1', prefix='docs/pdfs'):
print(str(blob))
<Blob: adta5770-bucket-1, docs/pdfs/, 1743567844880661>
<Blob: adta5770-bucket-1, docs/pdfs/a-human-centered-approach-to-developer-productivity.pdf, 1743568042270002>
<Blob: adta5770-bucket-1, docs/pdfs/a-mixed-methods-approach-to-understanding-user-trust-after-voice-assistant-failures.pdf, 1743568043092578>
Search

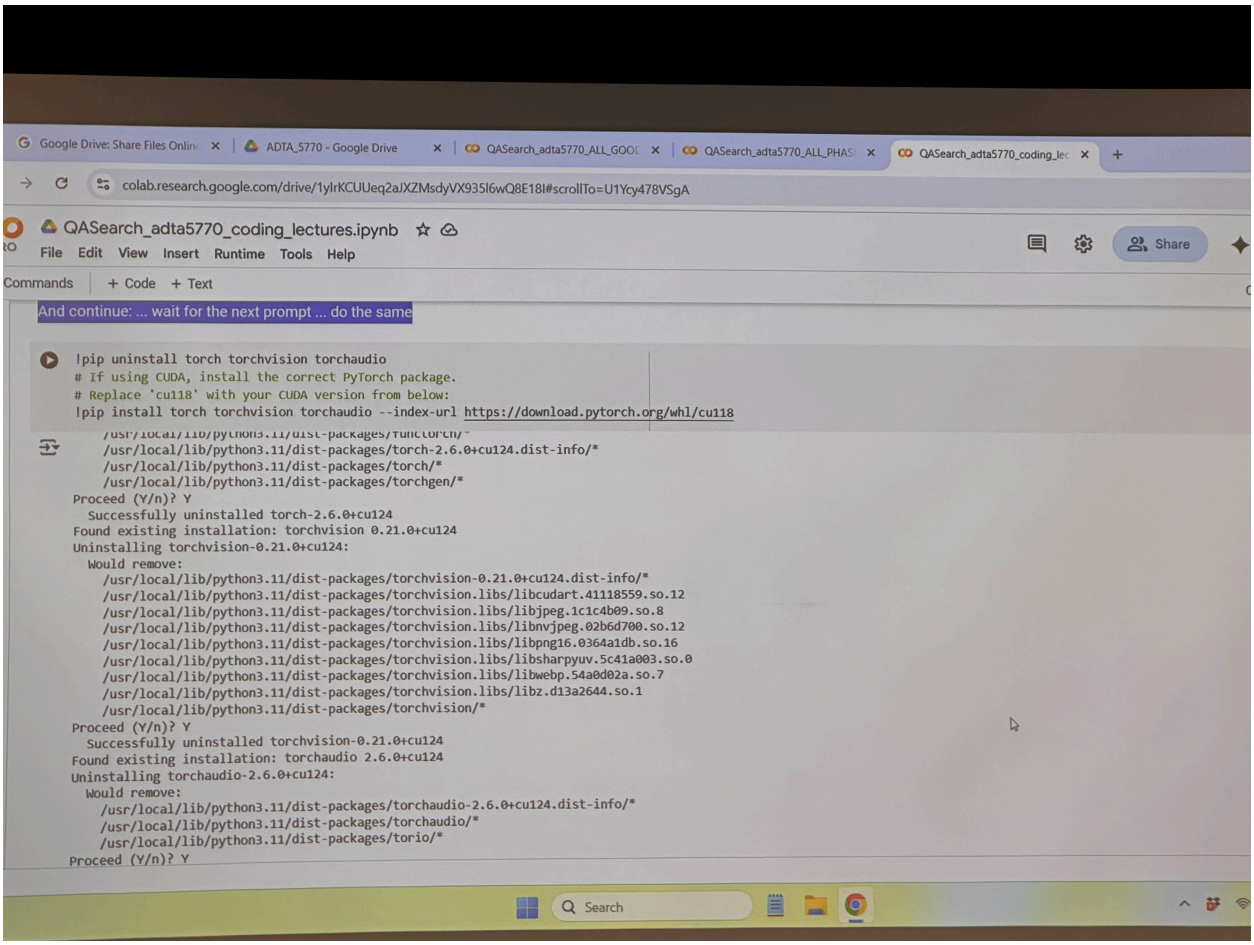

Google Drive: Share Files Onlin
ADTA_5770 - Google Drive
QASearch_adta5770_ALL_GOOI
QASearch_adta5770_ALL_PHAS
QASearch_adta5770_coding_lec
colab.research.google.com/drive/1yIrKCUUeq2aJXZMsdyVX935l6wQ8E18l#scrollTo=U1Ycy478VSgA
QASearch_adta5770_coding_lectures.ipynb
File Edit View Insert Runtime Tools Help
Share
Commands
+ Code
+ Text
[ ] # Access/Get the index using index_id
# Index name or ID: "3457218600471887872"
index_id = "3457218600471887872"
index = aiplatform.MatchingEngineIndex(index_name=index_id)
print(f"INDEX: {index}")
print(f"INDEX ID or name: {index.name}")
INDEX: <google.cloud.aiplatform.matching_engine.matching_engine_index.MatchingEngineIndex object at 0x7c7e98129550>
resource name: projects/446492160130/locations/us-central1/indexes/3457218600471887872
INDEX ID or name: 3457218600471887872
# Access/Get end point using end point ID
# End point name or ID: "2235188392012808192"
index_end_point_id = "2235188392012808192"
index_endpoint = aiplatform.MatchingEngineIndexEndpoint(
index_endpoint_name=index_end_point_id
)
print(f"INDEX ENDPOINT: {index_endpoint}")
print(f"INDEX ENDPOINT ID or name: {index_endpoint.name}")
Search

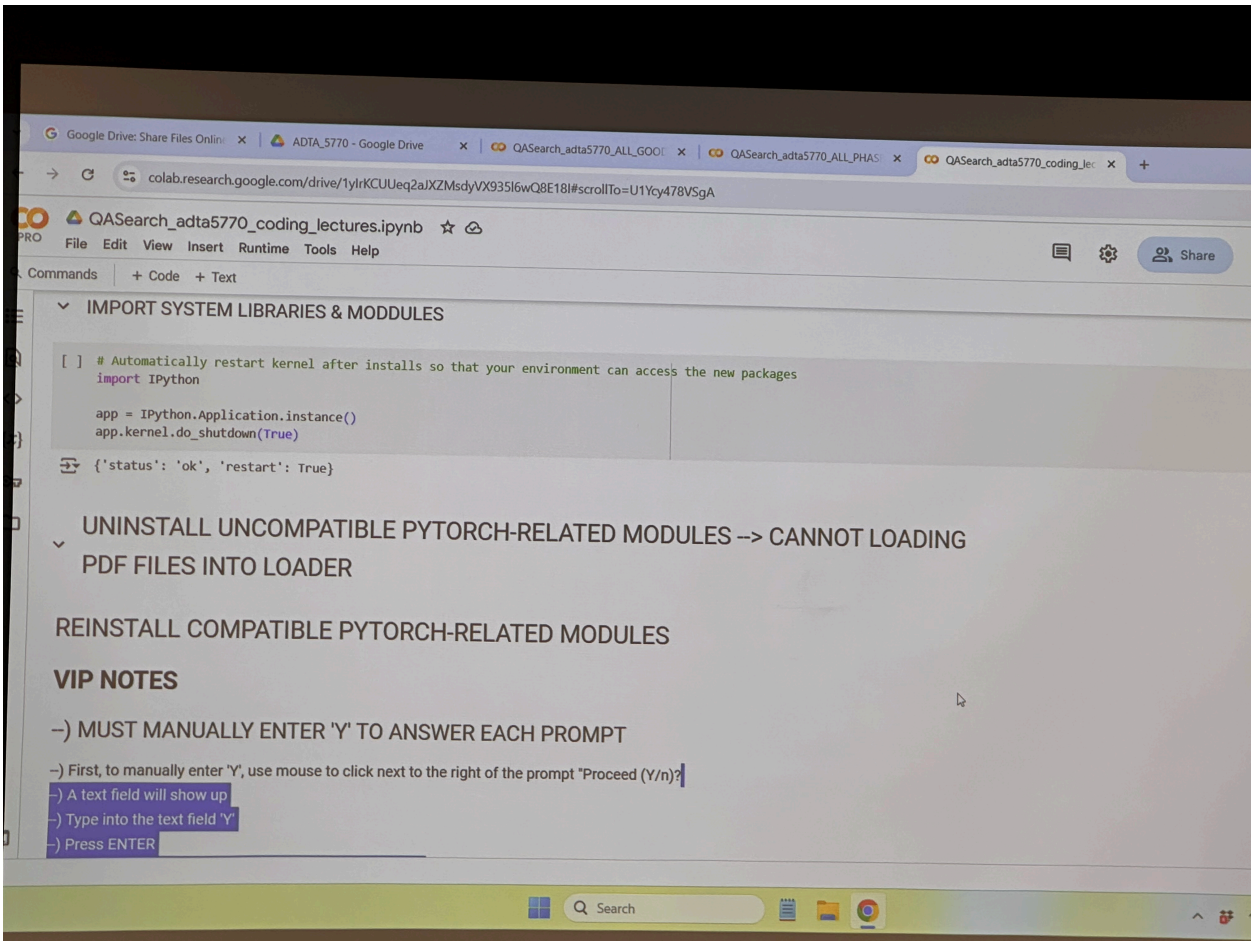

Google Drive: Share Files Onlin
ADTA_5770 - Google Drive
QASearch_adta5770_ALL_GOOI
QASearch_adta5770_ALL_PHAS
QASearch_adta5770_coding_lec
colab.research.google.com/drive/1yIrKCUUeq2aJXZMsdyVX935l6wQ8E18I#scrollTo=U1Ycy478VSgA
QASearch_adta5770_coding_lectures.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text
GET & SET index_id and end_point_id
NOTES: To be used in code for convenience
–) GET INDEX NAME, a.k.a. INDEX ID, that has been created.
–) GET END POINT ID of the END POINT of the DEPLOYED INDEX
[ ] # First, get the list of indexes that has been created for the vector search system
list_indexes = aiplatform.MatchingEngineIndex.list()
print(f"List of indexes: {list_indexes}")
List of indexes: [<google.cloud.aiplatform.matching_engine.matching_engine_index.MatchingEngineIndex object at 0x7c7e99433210>
resource name: projects/446492160130/locations/us-central1/indexes/3457218600471887872]
NOTES:
A shown above, only one index has been created for the system. Index:
Index name, i.e., ID, is 3457218600471887872.
We use the index name or ID to access the index.
# Access/Get the index using index_id
# Index name or ID: "3457218600471887872"
index_id = "3457218600471887872"
Search

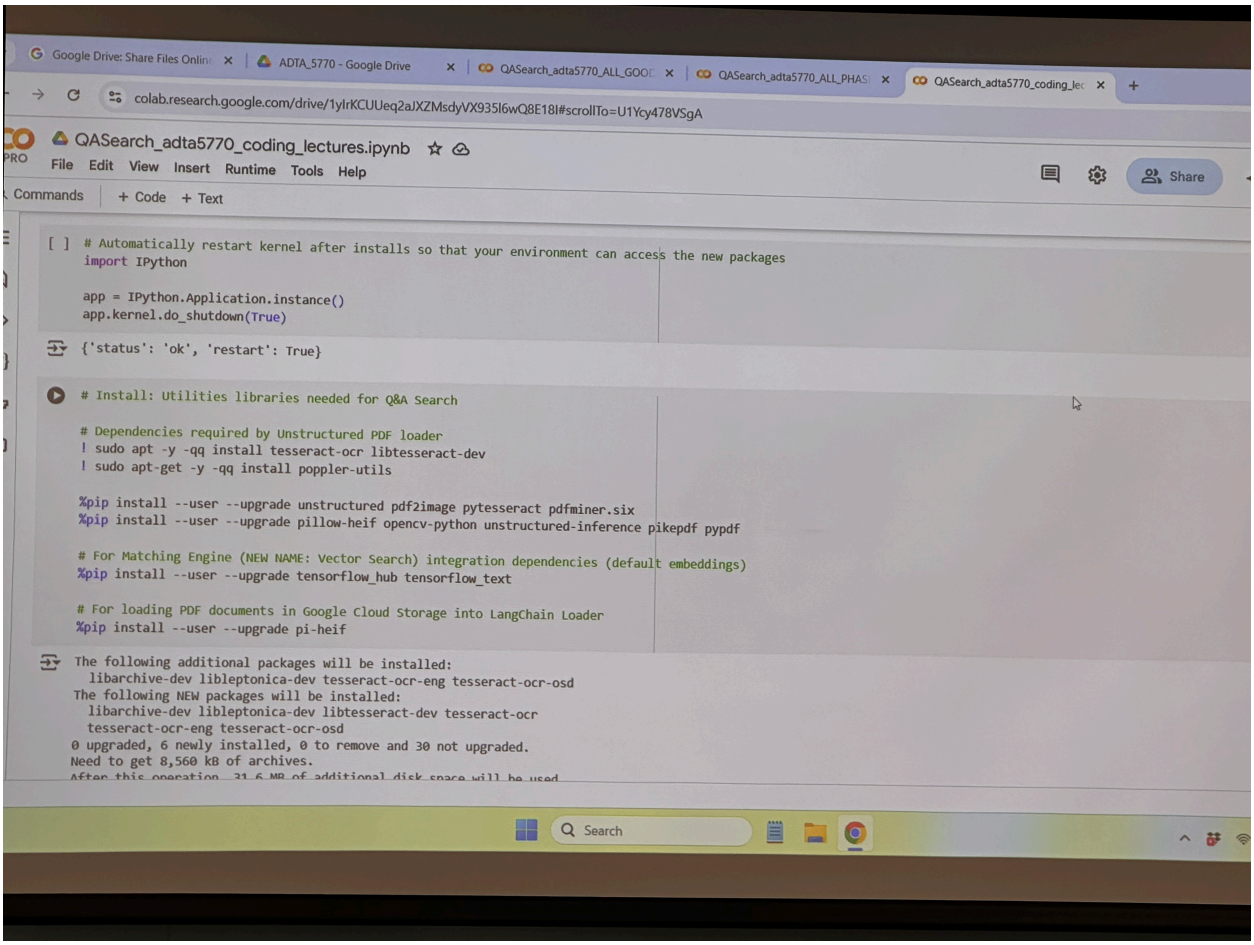

Google Drive: Share Files Onlin
ADTA_5770 - Google Drive
QASearch_adta5770_ALL_GOO
QASearch_adta5770_ALL_PHAS
QASearch_adta5770_coding_lec
colab.research.google.com/drive/1yIrKCUUeq2aJXZMsdyVX935l6wQ8E18I#scrollTo=U1Ycy478VSgA
QASearch_adta5770_coding_lectures.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text
[ ] from google.cloud import storage
[ ] # IMPORT FROM LANGCHAIN
import langchain
from langchain.chains import RetrievalQA
# from langchain.document_loaders import GCSDirectoryLoader
# from langchain.document_loaders import GCSFileLoader
from langchain.prompts import PromptTemplate
from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain_community.document_loaders import PyPDFLoader
from langchain_text_splitters import RecursiveCharacterTextSplitter
[ ] from langchain_google_community import GCSDirectoryLoader
from langchain_google_community import GCSFileLoader
# IMPORT FROM GCP: VERTEX AI & LANGHCAIN API
from langchain_google_vertexai import VertexAI
from langchain_google_vertexai import VertexAIEmbeddings
from langchain_google_vertexai import (
VectorSearchVectorStore,
VectorSearchVectorStoreDatastore,
)
[ ] # IMPORT OTHERS
import textwrap
Search


Google Drive: Share Files Onlin
ADTA_5770 - Google Drive
QASearch_adta5770_ALL_GOOD
QASearch_adta5770_ALL_PHAS
QASearch_adta5770_coding_lec
colab.research.google.com/drive/1yIrKCUUeq2aJXZMsdyVX935l6wQ8E18I#scrollTo=U1Ycy478VSgA
QASearch_adta5770_coding_lectures.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text
[ ] from google.cloud import storage
[ ] # IMPORT FROM LANGCHAIN
import langchain
from langchain.chains import RetrievalQA
# from langchain.document_loaders import GCSDirectoryLoader
# from langchain.document_loaders import GCSFileLoader
from langchain.prompts import PromptTemplate
from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain_community.document_loaders import PyPDFLoader
from langchain_text_splitters import RecursiveCharacterTextSplitter
[ ] from langchain_google_community import GCSDirectoryLoader
from langchain_google_community import GCSFileLoader
# IMPORT FROM GCP: VERTEX AI & LANGHCAIN API
from langchain_google_vertexai import VertexAI
from langchain_google_vertexai import VertexAIEmbeddings
from langchain_google_vertexai import (
VectorSearchVectorStore,
VectorSearchVectorStoreDatastore,
)
[ ] # IMPORT OTHERS
import textwrap
Search


Google Drive: Share Files Onlin
ADTA_5770 - Google Drive
QASearch_adta5770_ALL_GOOD
QASearch_adta5770_ALL_PHAS
QASearch_adta5770_coding_lec
colab.research.google.com/drive/1yIrKCUUeq2aJXZMsdyVX935l6wQ8E18I#scrollTo=U1Ycy478VSgA
QASearch_adta5770_coding_lectures.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text
[ ]
import IPython
app = IPython.Application.instance()
app.kernel.do_shutdown(True)
{'status': 'ok', 'restart': True}
[ ] import sys
if "google.colab" in sys.modules:
from google.colab import auth
auth.authenticate_user()
# IMPORT FROM GCP: VERTEX AI
from google.cloud import aiplatform
import vertexai
from google.cloud.aiplatform.matching_engine.matching_engine_index_endpoint import (
Namespace,
NumericNamespace,
)
# For GCS: Google Cloud Storage access and loading
from google.cloud import storage
[ ] # IMPORT FROM LANGCHAIN
import langchain
Search


Google Drive: Share Files Onlin
ADTA_5770 - Google Drive
QASearch_adta5770_ALL_GOOD
QASearch_adta5770_ALL_PHAS
QASearch_adta5770_coding_lec
colab.research.google.com/drive/1yIrKCUUeq2aJXZMsdyVX935l6wQ8E18I#scrollTo=U1Ycy478VSgA
QASearch_adta5770_coding_lectures.ipynb
File Edit View Insert Runtime Tools Help
Share
Commands + Code + Text
And continue: ... wait for the next prompt ... do the same
!pip uninstall torch torchvision torchaudio
If using CUDA, install the correct PyTorch package.
Replace 'cu118' with your CUDA version from below:
!pip install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cu118
/usr/local/lib/python3.11/dist-packages/torch-2.6.0+cu124.dist-info/*
/usr/local/lib/python3.11/dist-packages/torch/*
/usr/local/lib/python3.11/dist-packages/torchgen/*
Proceed (Y/n)? Y
Successfully uninstalled torch-2.6.0+cu124
Found existing installation: torchvision 0.21.0+cu124
Uninstalling torchvision-0.21.0+cu124:
Would remove:
/usr/local/lib/python3.11/dist-packages/torchvision-0.21.0+cu124.dist-info/*
/usr/local/lib/python3.11/dist-packages/torchvision.libs/libcudart.41118559.so.12
/usr/local/lib/python3.11/dist-packages/torchvision.libs/libjpeg.1c1c4b09.so.8
/usr/local/lib/python3.11/dist-packages/torchvision.libs/libnvjpeg.02b6d700.so.12
/usr/local/lib/python3.11/dist-packages/torchvision.libs/libpng16.0364a1db.so.16
/usr/local/lib/python3.11/dist-packages/torchvision.libs/libsharpyuv.5c41a003.so.0
/usr/local/lib/python3.11/dist-packages/torchvision.libs/libwebp.54a0d02a.so.7
/usr/local/lib/python3.11/dist-packages/torchvision.libs/libz.d13a2644.so.1
/usr/local/lib/python3.11/dist-packages/torchvision/*
Proceed (Y/n)? Y
Successfully uninstalled torchvision-0.21.0+cu124
Found existing installation: torchaudio 2.6.0+cu124
Uninstalling torchaudio-2.6.0+cu124:
Would remove:
/usr/local/lib/python3.11/dist-packages/torchaudio-2.6.0+cu124.dist-info/*
/usr/local/lib/python3.11/dist-packages/torchaudio/*
/usr/local/lib/python3.11/dist-packages/torio/*
Proceed (Y/n)? Y
Search


Google Drive: Share Files Onlin
ADTA_5770 - Google Drive
QASearch_adta5770_ALL_GOOG
QASearch_adta5770_ALL_PHAS
QASearch_adta5770_coding_lec
colab.research.google.com/drive/1yIrKCUUeq2aJXZMsdyVX935l6wQ8E18l#scrollTo=U1Ycy478VSgA
QASearch_adta5770_coding_lectures.ipynb
File Edit View Insert Runtime Tools Help
Share
Commands
+ Code + Text
IMPORT SYSTEM LIBRARIES & MODDULES
[ ] # Automatically restart kernel after installs so that your environment can access the new packages
import IPython
app = IPython.Application.instance()
app.kernel.do_shutdown(True)
{'status': 'ok', 'restart': True}
UNINSTALL UNCOMPATIBLE PYTORCH-RELATED MODULES --> CANNOT LOADING PDF FILES INTO LOADER
REINSTALL COMPATIBLE PYTORCH-RELATED MODULES
VIP NOTES
–) MUST MANUALLY ENTER 'Y' TO ANSWER EACH PROMPT
–) First, to manually enter 'Y', use mouse to click next to the right of the prompt "Proceed (Y/n)?
–) A text field will show up
–) Type into the text field 'Y'
–) Press ENTER
Search


Google Drive: Share Files Onlin
ADTA_5770 - Google Drive
QASearch_adta5770_ALL_GOOD
QASearch_adta5770_ALL_PHAS
QASearch_adta5770_coding_lec
colab.research.google.com/drive/1ylrKCUUeq2aJXZMsdyVX935l6wQ8E18l#scrollTo=U1Ycy478VSgA
QASearch_adta5770_coding_lectures.ipynb
File Edit View Insert Runtime Tools Help
Share
Commands
+ Code + Text
[ ] # Automatically restart kernel after installs so that your environment can access the new packages
import IPython
app = IPython.Application.instance()
app.kernel.do_shutdown(True)
{'status': 'ok', 'restart': True}
Install: Utilities libraries needed for Q&A Search
Dependencies required by Unstructured PDF loader
! sudo apt -y -qq install tesseract-ocr libtesseract-dev
! sudo apt-get -y -qq install poppler-utils
%pip install --user --upgrade unstructured pdf2image pytesseract pdfminer.six
%pip install --user --upgrade pillow-heif opencv-python unstructured-inference pikepdf pypdf
For Matching Engine (NEW NAME: Vector Search) integration dependencies (default embeddings)
%pip install --user --upgrade tensorflow_hub tensorflow_text
For loading PDF documents in Google Cloud Storage into LangChain Loader
%pip install --user --upgrade pi-heif
The following additional packages will be installed:
libarchive-dev libleptonica-dev tesseract-ocr-eng tesseract-ocr-osd
The following NEW packages will be installed:
libarchive-dev libleptonica-dev libtesseract-dev tesseract-ocr
tesseract-ocr-eng tesseract-ocr-osd
0 upgraded, 6 newly installed, 0 to remove and 30 not upgraded.
Need to get 8,560 kB of archives.
Search