```
!pip install -qU \langchain==0.0.292 \openai==0.28.0 \datasets==2.10.1 \pinecone-client==2.2.4 \tiktoken==0.5.1
!pip install cohere
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.7/1.7 MB 19.6 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 76.5/76.5 kB 7.8 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 469.0/469.0 kB 33.9 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 179.4/179.4 kB 15.2 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.0/2.0 MB 65.6 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 48.2/48.2 kB 4.8 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 110.5/110.5 kB 11.2 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 134.8/134.8 kB 12.1 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 62.5/62.5 kB 6.5 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 300.4/300.4 kB 25.7 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 49.4/49.4 kB 5.0 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 134.3/134.3 kB 12.3 MB/s eta 0:00:00
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source
llmx 0.0.15a0 requires cohere, which is not installed.
Collecting cohere
  Downloading cohere-4.37-py3-none-any.whl (48 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 48.9/48.9 kB 1.3 MB/s eta 0:00:00
Requirement already satisfied: aiohttp<4.0,>=3.0 in /usr/local/lib/python3.10/dist-packages (from cohere) (3.9.1)
Collecting backoff<3.0,>=2.0 (from cohere)
  Downloading backoff-2.2.1-py3-none-any.whl (15 kB)
Collecting fastavro<2.0,>=1.8 (from cohere)
  Downloading fastavro-1.9.1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (3.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.1/3.1 MB 32.1 MB/s eta 0:00:00
Requirement already satisfied: importlib_metadata<7.0,>=6.0 in /usr/local/lib/python3.10/dist-packages (from cohere) (6.8.0)
Requirement already satisfied: requests<3.0.0,>=2.25.0 in /usr/local/lib/python3.10/dist-packages (from cohere) (2.31.0)
Requirement already satisfied: urllib3<3,>=1.26 in /usr/local/lib/python3.10/dist-packages (from cohere) (2.0.7)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp<4.0,>=3.0->cohere) (23.1.0)
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.10/dist-packages (from aiohttp<4.0,>=3.0->cohere) (6.0.4)
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp<4.0,>=3.0->cohere) (1.9.3)
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from aiohttp<4.0,>=3.0->cohere) (1.4.0)
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.10/dist-packages (from aiohttp<4.0,>=3.0->cohere) (1.3.1)
Requirement already satisfied: async-timeout<5.0,>=4.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp<4.0,>=3.0->cohere) (4.0.
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.10/dist-packages (from importlib_metadata<7.0,>=6.0->cohere) (3.17.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests<3.0.0,>=2.25.0->cohere
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<3.0.0,>=2.25.0->cohere) (3.6)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3.0.0,>=2.25.0->cohere) (202
Installing collected packages: fastavro, backoff, cohere
Successfully installed backoff-2.2.1 cohere-4.37 fastavro-1.9.1
```

```
#Get API key
import os
from langchain.chat_models import ChatOpenAI

os.environ["OPENAI_API_KEY"] = "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"

chat = ChatOpenAI(
    openai_api_key=os.environ["OPENAI_API_KEY"],
    model='gpt-3.5-turbo'
)


#Importing the Data
from datasets import load_dataset

dataset = load_dataset(
    "jamescalam/llama-2-arxiv-papers-chunked",
    split="train"
)

dataset
```

```
Downloading readme: 100%  409/409 [00:00<00:00, 20.2kB/s]
Downloading and preparing dataset json/jamescalam--llama-2-arxiv-papers-chunked to /root/.cache/huggingface/datasets/jamescalam___json/j
Downloading data files: 100%  1/1 [00:05<00:00, 5.35s/it]
Downloading data: 100%  14.4M/14.4M [00:01<00:00, 12.7MB/s]
Extracting data files: 100%  1/1 [00:00<00:00, 43.26it/s]

Dataset json downloaded and prepared to /root/.cache/huggingface/datasets/jamescalam___json/jamescalam--llama-2-arxiv-papers-chunked-ea2
Dataset({
    features: ['doi', 'chunk-id', 'chunk', 'id', 'title', 'summary', 'source', 'authors', 'categories', 'comment', 'journal_ref',
'primary_category', 'published', 'updated', 'references'],
    num_rows: 4838
})
```

```
dataset[0]
```

```
{'doi': '1102.0183',
 'chunk-id': '0',
 'chunk': 'High-Performance Neural Networks\nfor Visual Object Classi\x0ccation\nDan C. Cire\x18 san, Ueli Meier, Jonathan Masci,\nLuca
M. Gambardella and J\x7f urgen Schmidhuber\nTechnical Report No. IDSIA-01-11\nJanuary 2011\nIDSIA / USI-SUPSI\nDalle Molle Institute
for Arti\x0ccial Intelligence\nGalleria 2, 6928 Manno, Switzerland\nIDSIA is a joint institute of both University of Lugano (USI) and
University of Applied Sciences of Southern Switzerland (SUPSI),\nand was founded in 1988 by the Dalle Molle Foundation which promoted
quality of life.\nThis work was partially supported by the Swiss Commission for Technology and Innovation (CTI), Project n. 9688.1
IFF:\nIntelligent Fill in Form.arXiv:1102.0183v1  [cs.AI]  1 Feb 2011\nTechnical Report No. IDSIA-01-11 1\nHigh-Performance Neural
Networks\nfor Visual Object Classi\x0ccation\nDan C. Cire\x18 san, Ueli Meier, Jonathan Masci,\nLuca M. Gambardella and J\x7f urgen
Schmidhuber\nJanuary 2011\nAbstract\nWe present a fast, fully parameterizable GPU implementation of Convolutional Neural\nNetwork
variants. Our feature extractors are neither carefully designed nor pre-wired, but',
 'id': '1102.0183',
 'title': 'High-Performance Neural Networks for Visual Object Classification',
 'summary': 'We present a fast, fully parameterizable GPU implementation of Convolutional\nNeural Network variants. Our feature
extractors are neither carefully designed\nnor pre-wired, but rather learned in a supervised way. Our deep hierarchical\narchitectures
achieve the best published results on benchmarks for object\nclassification (NORB, CIFAR10) and handwritten digit recognition (MNIST),
with\nerror rates of 2.53%, 19.51%, 0.35%, respectively. Deep nets trained by simple\nback-propagation perform better than more shallow
ones. Learning is\nsurprisingly rapid. NORB is completely trained within five epochs. Test error\nrates on MNIST drop to 2.42%, 0.97%
and 0.48% after 1, 3 and 17 epochs,\nrespectively.',
 'source': 'http://arxiv.org/pdf/1102.0183',
 'authors': ['Dan C. Cireşan',
  'Ueli Meier',
  'Jonathan Masci',
  'Luca M. Gambardella',
  'Jürgen Schmidhuber'],
 'categories': ['cs.AI', 'cs.NE'],
 'comment': '12 pages, 2 figures, 5 tables',
 'journal_ref': None,
 'primary_category': 'cs.AI',
 'published': '20110201',
 'updated': '20110201',
 'references': []}
```

```
#Building the Knowledge Base
import pinecone
#Get API key from app.pinecone.io and environment from console
pinecone.init(
    api_key='c86b72b1-4266-4b63-9b7e-8a7735663780',
    environment='gcp-starter'
)
```

```
#initialize the index
import time
index_name = 'llama-2-rag'
if index_name not in pinecone.list_indexes():
    pinecone.create_index(
        index_name,
        dimension=300,
        metric='cosine'
    )
    # wait for index to finish initialization
    while not pinecone.describe_index(index_name).status['ready']:
        time.sleep(1)

index = pinecone.Index(index_name)
```

```
# connect to the index
index.describe_index_stats()
```

```
{'dimension': 1536,
 'index_fullness': 0.004,
 'namespaces': {'': {'vector_count': 400}},
 'total_vector_count': 400}
```

```
#create Vector embeddings
from langchain.embeddings.openai import OpenAIEmbeddings

embed_model = OpenAIEmbeddings(model="text-embedding-ada-002")
```

```
#Looping through our dataset and embedding and inserting everything in batches.
from tqdm.auto import tqdm  # for progress bar
#This makes it easier to iterate over the dataset
data = dataset.to_pandas()
batch_size = 100
for i in tqdm(range(0, len(data), batch_size)):
    i_end = min(len(data), i+batch_size)
    # get batch of data
    batch = data.iloc[i:i_end]
    # generate unique ids for each chunk
    ids = [f"{x['doi']}-{x['chunk-id']}" for i, x in batch.iterrows()]
    # get text to embed
    texts = [x['chunk'] for _, x in batch.iterrows()]
    # embed text
    embeds = embed_model.embed_documents(texts)
    # get metadata to store in Pinecone
    metadata = [
        {'text': x['chunk'],
         'source': x['source'],
         'title': x['title']} for i, x in batch.iterrows()
    ]
    # add to Pinecone
    index.upsert(vectors=zip(ids, embeds, metadata))
```

10% 5/49 [01:32<10:24, 14.18s/it]

```
WARNING:langchain.embeddings.openai:Retrying langchain.embeddings.openai.embed_with_retry.<locals>._embed_with_retry in 4.0 seconds as i
WARNING:langchain.embeddings.openai:Retrying langchain.embeddings.openai.embed_with_retry.<locals>._embed_with_retry in 4.0 seconds as i
WARNING:langchain.embeddings.openai:Retrying langchain.embeddings.openai.embed_with_retry.<locals>._embed_with_retry in 4.0 seconds as i
WARNING:langchain.embeddings.openai:Retrying langchain.embeddings.openai.embed_with_retry.<locals>._embed_with_retry in 4.0 seconds as i
WARNING:langchain.embeddings.openai:Retrying langchain.embeddings.openai.embed_with_retry.<locals>._embed_with_retry in 8.0 seconds as i
WARNING:langchain.embeddings.openai:Retrying langchain.embeddings.openai.embed_with_retry.<locals>._embed_with_retry in 4.0 seconds as i
WARNING:langchain.embeddings.openai:Retrying langchain.embeddings.openai.embed_with_retry.<locals>._embed_with_retry in 4.0 seconds as i
WARNING:langchain.embeddings.openai:Retrying langchain.embeddings.openai.embed_with_retry.<locals>._embed_with_retry in 4.0 seconds as i
WARNING:langchain.embeddings.openai:Retrying langchain.embeddings.openai.embed_with_retry.<locals>._embed_with_retry in 4.0 seconds as i
WARNING:langchain.embeddings.openai:Retrying langchain.embeddings.openai.embed_with_retry.<locals>._embed_with_retry in 4.0 seconds as i
WARNING:langchain.embeddings.openai:Retrying langchain.embeddings.openai.embed_with_retry.<locals>._embed_with_retry in 8.0 seconds as i
WARNING:langchain.embeddings.openai:Retrying langchain.embeddings.openai.embed_with_retry.<locals>._embed_with_retry in 10.0 seconds as
---------------------------------------------------------------------------
RateLimitError                            Traceback (most recent call last)
<ipython-input-22-2b8c7ccfb5d2> in <cell line: 7>()
     14     texts = [x['chunk'] for _, x in batch.iterrows()]
     15     # embed text
---> 16     embeds = embed_model.embed_documents(texts)
     17     # get metadata to store in Pinecone
     18     metadata = [
```

15 frames

```
/usr/local/lib/python3.10/dist-packages/openai/api_requestor.py in _interpret_response_line(self, rbody, rcode, rheaders, stream)
    763         stream_error = stream and "error" in resp.data
    764         if stream_error or not 200 <= rcode < 300:
--> 765             raise self.handle_error_response(
    766                 rbody, rcode, resp.data, rheaders, stream_error=stream_error
    767             )

RateLimitError: Rate limit reached for text-embedding-ada-002 in organization org-5gt5T65OTMSdgS0rsWTQaE0G on requests per min (RPM):
Limit 3, Used 3, Requested 1. Please try again in 20s. Visit https://platform.openai.com/account/rate-limits to learn more. You can
increase your rate limit by adding a payment method to your account at https://platform.openai.com/account/billing.
```

```
#connect to the index to check vectors
index.describe_index_stats()
```

```
{'dimension': 1536,
 'index_fullness': 0.005,
 'namespaces': {'': {'vector_count': 500}},
 'total_vector_count': 500}
```

```
#Vector database
from langchain.vectorstores import Pinecone

text_field = "text"  # the metadata field that contains our text

# initialize the vector store object
vectorstore = Pinecone(
    index, embed_model.embed_query, text_field
)
```

```
/usr/local/lib/python3.10/dist-packages/langchain/vectorstores/pinecone.py:59: UserWarning: Passing in `embedding` as a Callable is depr
  warnings.warn(
```

```
#Query the index and see if we have any relevant information given our question
query = "What is so special about Llama 2?"
vectorstore.similarity_search(query, k=2)
```

```
[Document(page_content='for these high-level abstractions.\n\x0fMemes Divide-and-Conquer Hypothesis: Linguistic exchange, individual
learning\nand the recombination of memes constitute an e\x0ecient evolutionary recombination operator in the meme-space. This helps
human learners to collectively build better internal\nrepresentations of their environment, including fairly high-level
abstractions.\nThis paper is focused on \\ Point 1 " and testing the \\ Guided Learning Hypothesis ", using\nmachine learning
algorithms to provide experimental evidence. The experiments performed\nalso provide evidence in favor of the \\ Deeper Harder
Hypothesis " and associated \\ Abstractions\nHarder Hypothesis ". Machine Learning is still far beyond the current capabilities of
humans,\nand it is important to tackle the remaining obstacles to approach AI. For this purpose, the\nquestion to be answered is why
tasks that humans learn e\x0bortlessly from very few examples,\nwhile machine learning algorithms fail miserably?\n2. Recent work
showed that rather deep feedforward networks can be very successfully trained when large\nquantities of labeled data are available
(Ciresan et al., 2010; Glorot et al., 2011a; Krizhevsky et al., 2012).\nNonetheless, the experiments reported here suggest that it all
depends on the task being considered, since\neven with very large quantities of labeled examples, the deep networks trained here were
unsuccessful.\n4', metadata={'source': 'http://arxiv.org/pdf/1301.4083', 'title': 'Knowledge Matters: Importance of Prior Information
for Optimization'}),
 Document(page_content='et al., 2010) machine learning libraries. We have selected 2 hidden layers, the recti\x0cer
activation\nfunction, and 2048 hidden units per layer. We cross-validated three hyper-parameters of the\nmodel using random-search,
sampling the learning rates \x0fin log-domain, and selecting L1\nandL2 regularization penalty coe\x0ecients in sets of \x0cxed values,
evaluating 64 hyperparameter\nvalues. The range of the hyperparameter values are
\x0f2[0:0001;1],L12f0:;1e\x006;1e\x005;1e\x004g\nandL22f0;1e\x006;1e\x005g. As a result, the following were selected: L1 = 1e\x006,L2 =
1e\x005\nand\x0f= 0:05.\n5.2.4 Random Forests\nWe used scikit-learn\'s implementation of \\Random Forests" decision tree learning. The
Random Forests algorithm creates an ensemble of decision trees by randomly selecting for each tree\na subset of features and applying
bagging to combine the individual decision trees (Breiman,\n2001). We have used grid-search and cross-validated the maxdepth ,minsplit,
and number', metadata={'source': 'http://arxiv.org/pdf/1301.4083', 'title': 'Knowledge Matters: Importance of Prior Information for
Optimization'})]
```

```
#Connect the output from our vectorstore to our chat chatbot, LLM will be able to parse this informatio
def augment_prompt(query: str):
    # get top 3 results from knowledge base
    results = vectorstore.similarity_search(query, k=2)
    # get the text from the results
    source_knowledge = "\n".join([x.page_content for x in results])
    # feed into an augmented prompt
    augmented_prompt = f"""Using the contexts below, answer the query.

    Contexts:
    {source_knowledge}

    Query: {query}"""
    return augmented_prompt
```

```
#
print(augment_prompt(query))
```

```
Using the contexts below, answer the query.

    Contexts:
    for these high-level abstractions.
Memes Divide-and-Conquer Hypothesis: Linguistic exchange, individual learning
and the recombination of memes constitute an e cient evolutionary recombination operator in the meme-space. This helps human learners to
representations of their environment, including fairly high-level abstractions.
This paper is focused on \ Point 1 " and testing the \ Guided Learning Hypothesis ", using
machine learning algorithms to provide experimental evidence. The experiments performed
also provide evidence in favor of the \ Deeper Harder Hypothesis " and associated \ Abstractions
Harder Hypothesis ". Machine Learning is still far beyond the current capabilities of humans,
and it is important to tackle the remaining obstacles to approach AI. For this purpose, the
question to be answered is why tasks that humans learn e ortlessly from very few examples,
```

```
while machine learning algorithms fail miserably?
2. Recent work showed that rather deep feedforward networks can be very successfully trained when large
quantities of labeled data are available (Ciresan et al., 2010; Glorot et al., 2011a; Krizhevsky et al., 2012).
Nonetheless, the experiments reported here suggest that it all depends on the task being considered, since
even with very large quantities of labeled examples, the deep networks trained here were unsuccessful.
4
et al., 2010) machine learning libraries. We have selected 2 hidden layers, the rectier activation
function, and 2048 hidden units per layer. We cross-validated three hyper-parameters of the
model using random-search, sampling the learning rates in log-domain, and selecting L1
andL2 regularization penalty coecients in sets of xed values, evaluating 64 hyperparameter
values. The range of the hyperparameter values are 2[0:0001;1],L12f0:;1e6;1e5;1e4g
andL22f0;1e6;1e5g. As a result, the following were selected: L1 = 1e6,L2 = 1e5
and= 0:05.
5.2.4 Random Forests
We used scikit-learn's implementation of \Random Forests" decision tree learning. The Random Forests algorithm creates an ensemble of de
a subset of features and applying bagging to combine the individual decision trees (Breiman,
2001). We have used grid-search and cross-validated the maxdepth ,minsplit, and number

    Query: What is so special about Llama 2?
```

```
from langchain.schema import (
    SystemMessage,
    HumanMessage,
    AIMessage
)

messages = [
    SystemMessage(content="You are a helpful assistant."),
    HumanMessage(content="Hi AI, how are you today?"),
    AIMessage(content="I'm great thank you. How can I help you?"),
    HumanMessage(content="I'd like to understand string theory.")
]
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
<ipython-input-4-6df580d7aa35> in <cell line: 1>()
----> 1 from langchain.schema import (
      2     SystemMessage,
      3     HumanMessage,
      4     AIMessage
      5 )

ModuleNotFoundError: No module named 'langchain'

---------------------------------------------------------------------------
NOTE: If your import is failing due to a missing package, you can
manually install dependencies using either !pip or !apt.

To view examples of installing some common dependencies, click the
"Open Examples" button below.
---------------------------------------------------------------------------
```

OPEN EXAMPLES

```
# create a new user prompt
prompt = HumanMessage(
    content=augment_prompt(query)
)
# add to messages
messages.append(prompt)

res = chat(messages)

print(res.content)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-3-db20c7b23178> in <cell line: 2>()
      1 # create a new user prompt
----> 2 prompt = HumanMessage(
      3     content=augment_prompt(query)
      4 )
      5 # add to messages

NameError: name 'HumanMessage' is not defined
```

```
#without RAG
prompt = HumanMessage(
    content="what safety measures were used in the development of llama 2?"
)

res = chat(messages + [prompt])
print(res.content)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-2-0cf55540186c> in <cell line: 1>()
----> 1 prompt = HumanMessage(
      2     content="what safety measures were used in the development of llama 2?"
      3 )
      4
      5 res = chat(messages + [prompt])

NameError: name 'HumanMessage' is not defined
```

```
#with RAG
prompt = HumanMessage(
    content=augment_prompt(
        "what safety measures were used in the development of llama 2?"
    )
)

res = chat(messages + [prompt])
print(res.content)
```

```
I'm sorry, but I couldn't find any information about the development of "Llama 2" in the provided contexts. It seems that the informatic
```