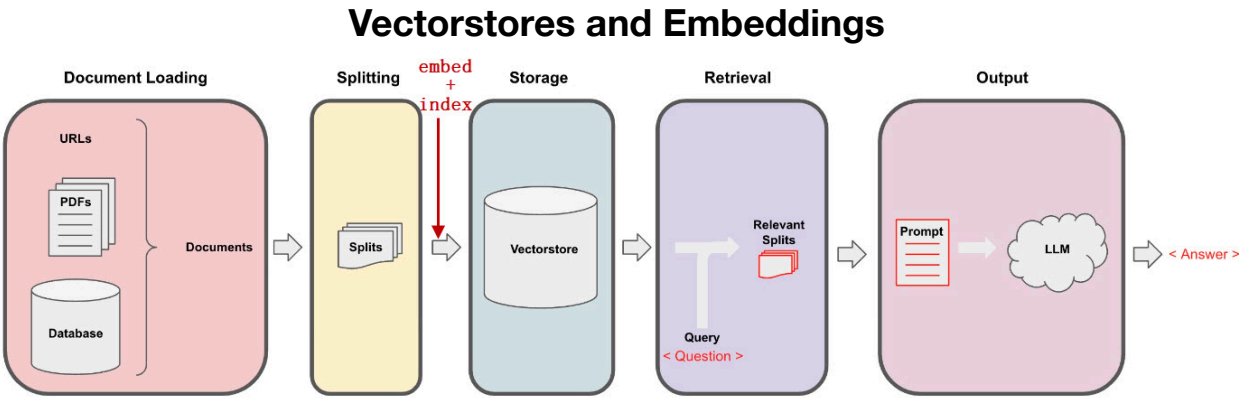
19953 Zhiyu Zhang

DS565 Week 7 Homework 2

Nov 3rd, 2024



Step 1: Set up API

```
import os
import openai
import sys
sys.path.append('../..')

from dotenv import load_dotenv, find_dotenv
_ = load_dotenv(find_dotenv()) # read local .env file
openai.api_key  = os.environ['OPENAI_API_KEY']
```

Step 2: Load and split documents

Make sure to use the most updated module names. You can check all the modules here. We're loading a file twice as messy data to create one of the failure cases later. You can download the sample files from the links I inserted below or use your own files.

```
from langchain_community.document_loaders import PyPDFLoader

# Load PDF
loaders = [
    PyPDFLoader("docs/MachineLearning-Lecture01.pdf"),
    PyPDFLoader("docs/MachineLearning-Lecture01.pdf"),
    PyPDFLoader("docs/MachineLearning-Lecture02.pdf"),
    PyPDFLoader("docs/MachineLearning-Lecture03.pdf")]
docs = []
for loader in loaders:
    docs.extend(loader.load())

# Split
from langchain_text_splitters.character import RecursiveCharacterTextSplitter

text_splitter = RecursiveCharacterTextSplitter(
    chunk_size = 1500,
    chunk_overlap = 150)

splits = text_splitter.split_documents(docs)
```

Result:

```
[7]: splits = text_splitter.split_documents(docs)

[8]: len(splits)

[8]: 208
```

Step 3: Embed the splits and create indexes

Use OpenAI API to embed the data, then calculate the dot product of two embeddings to represent their semantic similarity.

```
# !pip install langchain_openai
from langchain_openai import OpenAIEmbeddings

embedding = OpenAIEmbeddings()

sentence1 = "i like dogs"
sentence2 = "i like canines"
sentence3 = "the weather is ugly outside"

embedding1 = embedding.embed_query(sentence1)
embedding2 = embedding.embed_query(sentence2)
embedding3 = embedding.embed_query(sentence3)

import numpy as np

np.dot(embedding1, embedding2)
np.dot(embedding1, embedding3)
np.dot(embedding2, embedding3)
```

Result:

```
[15]: np.dot(embedding1, embedding2)

[15]: 0.9630397143104906

[16]: np.dot(embedding1, embedding3)

[16]: 0.7702742223497947

[17]: np.dot(embedding2, embedding3)

[17]: 0.7590147808716895
```

Step 4: Store the embeddings in a vectorstore

```
# !pip install langchain_chroma
from langchain_chroma import Chroma
```

19953 Zhiyu Zhang

DS565 Week 7 Homework 2

Nov 3rd, 2024

```
persist_directory = 'docs/chroma/'

# !rm -rf ./docs/chroma  # remove old database files if any

vectordb = Chroma.from_documents(
    documents=splits,
    embedding=embedding,
    persist_directory=persist_directory)
```

Result:

```
[24]: print(vectordb._collection.count())
      416
```

Step 5: Similarity search

Using similarity search to retrieve documents relevant to a question.

```
question = "is there an email i can ask for help"
docs = vectordb.similarity_search(question,k=3)
```

Result:

```
[26]: question = "is there an email i can ask for help"

[27]: docs = vectordb.similarity_search(question,k=3)

[28]: len(docs)

[28]: 3

[29]: docs[0].page_content

[29]: "cs229-qa@cs.stanford.edu. This goes to an account that's read by all the TAs and me. So \nrather than sending u
      s email individually, if you send email to this account, it will \nactually let us get back to you maximally qui
      ckly with answers to your questions.  \nIf you're asking questions about homework problems, please say in the su
      bject line which \nassignment and which question the email refers to, since that will also help us to route \nyo
      ur question to the appropriate TA or to me appropriately and get the response back to \nyou quickly.  \nLet's se
      e. Skipping ahead — let's see — for homework, one midterm, one open and term \nproject. Notice on the honor cod
      e. So one thing that I think will help you to succeed and \ndo well in this class and even help you to enjoy thi
      s class more is if you form a study \ngroup.  \nSo start looking around where you're sitting now or at the end o
      f class today, mingle a \nlittle bit and get to know your classmates. I strongly encourage you to form study gro
      ups \nand sort of have a group of people to study with and have a group of your fellow students \nto talk over t
      hese concepts with. You can also post on the class newsgroup if you want to \nuse that to try to form a study gr
      oup.  \nBut some of the problems sets in this class are reasonably difficult. People that have \ntaken the class
      before may tell you they were very difficult. And just I bet it would be \nmore fun for you, and you'd probably
      have a better learning experience if you form a"
```

Step 6: Failure modes and solutions

6.1 Edge Case 1: Diversity Failure

Because of the duplicate MachineLearning-Lecture01.pdf in the index, we're getting duplicate chunks. Semantic search fetches all similar documents, but does not enforce diversity. Thus, docs[0] and docs[1] are identical.

```
question = "what did they say about matlab?"
docs = vectordb.similarity_search(question,k=5)
```

Result:

```
[33]: question = "what did they say about matlab?"

[34]: docs = vectordb.similarity_search(question,k=5)

[81]: docs[0].page_content[:100]

[81]: 'MachineLearning-Lecture03  \nInstructor (Andrew Ng):Okay. Good morning and welcome back to the third '

[83]: docs[1].page_content[:100]

[83]: 'MachineLearning-Lecture03  \nInstructor (Andrew Ng):Okay. Good morning and welcome back to the third '
```

Solution: Use maximum marginal relevance to achieve both relevance to the query and diversity among the results.

```
docs_mmr = vectordb.max_marginal_relevance_search(
              question,k=3)
```

```
[75]: docs_mmr = vectordb.max_marginal_relevance_search(
                 question,k=3)

[77]: docs_mmr[0].page_content[:100]

[77]: 'MachineLearning-Lecture03  \nInstructor (Andrew Ng):Okay. Good morning and welcome back to the third '

[79]: docs_mmr[1].page_content[:100]

[79]: 'Instructor (Andrew Ng):All right, so who thought driving could be that dramatic, right? \nSwitch back'
```

6.2 Edge Case 2: Specificity Failure

The question below asks a question about the third lecture, but the search result includes other lectures as well.

```
question = "what did they say about regression in the third lecture?"
docs = vectordb.similarity_search(question,k=5)
```

Result:

19953 Zhiyu Zhang

DS565 Week 7 Homework 2

Nov 3rd, 2024

[38]:
```
question = "what did they say about regression in the third lecture?"
```

[39]:
```
docs = vectordb.similarity_search(question,k=5)
```

[40]:
```
for doc in docs:
    print(doc.metadata)
```

```
{'page': 0, 'source': 'docs/MachineLearning-Lecture03.pdf'}
{'page': 0, 'source': 'docs/MachineLearning-Lecture03.pdf'}
{'page': 0, 'source': 'docs/MachineLearning-Lecture03.pdf'}
{'page': 2, 'source': 'docs/MachineLearning-Lecture02.pdf'}
{'page': 2, 'source': 'docs/MachineLearning-Lecture02.pdf'}
```

[41]:
```
print(docs[4].page_content)
```

```
Instructor (Andrew Ng):All right, so who thought driving could be that dramatic, right?
Switch back to the chalkboard, please. I should say, this work was done about 15 years
ago and autonomous driving has come a long way. So many of you will have heard of the
DARPA Grand Challenge, where one of my colleagues, Sebastian Thrun, the winning
team's drive a car across a desert by itself.
So Alvin was, I think, absolutely amazing work for its time, but autonomous driving has
obviously come a long way since then. So what you just saw was an example, again, of
supervised learning, and in particular it was an example of what they call the regression
problem, because the vehicle is trying to predict a continuous value variables of a
continuous value steering directions, we call the regression problem.
And what I want to do today is talk about our first supervised learning algorithm, and it
will also be to a regression task. So for the running example that I'm going to use
throughout today's lecture, you're going to return to the example of trying to predict
housing prices. So here's actually a dataset collected by TA, Dan Ramage, on housing
prices in Portland, Oregon.
So here's a dataset of a number of houses of different sizes, and here are their asking
prices in thousands of dollars, $200,000. And so we can take this data and plot it, square
feet, best price, and so you make your other dataset like that. And the question is, given a
```

Solution: Use metadata to address the specificity.

```
docs1 = vectordb.similarity_search(
    question,
    k=3,
    filter={"source":
     "docs/MachineLearning-Lecture03.pdf"}
)
```

[102]:
```
docs1 = vectordb.similarity_search(
    question,
    k=3,
    filter={"source":
     "docs/MachineLearning-Lecture03.pdf"}
)
```

[104]:
```
for d in docs1:
    print(d.metadata)
```

```
{'page': 0, 'source': 'docs/MachineLearning-Lecture03.pdf'}
{'page': 0, 'source': 'docs/MachineLearning-Lecture03.pdf'}
{'page': 0, 'source': 'docs/MachineLearning-Lecture03.pdf'}
```