第四章 向量数据库与词向量(Vectorstores and Embeddings)

让我们一起回顾一下检索增强生成（RAG）的整体工作流程：

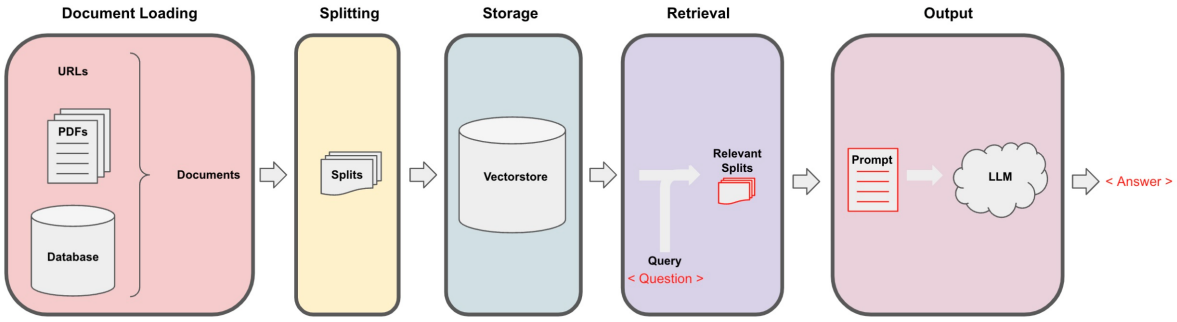

图 4.4 检索增强生成整体流程

前两节课我们讨论了`Document Loading`（文档加载）和`Splitting`（分割）。

下面我们将使用前两节课的知识对文档进行加载分割。

一、读取文档

下面文档是 datawhale 官方开源的 matplotlib 教程链接 https://datawhalechina.github.io/fantastic-matplotlib/index.html ，可在该网站上下载对应的教程。

注意，本章节需要安装第三方库`pypdf`、`chromadb`

```
from langchain.document_loaders import PyPDFLoader

# 加载 PDF
loaders_chinese = [
    # 故意添加重复文档，使数据混乱
    PyPDFLoader("docs/matplotlib/第一回：Matplotlib初相识.pdf"),
    PyPDFLoader("docs/matplotlib/第一回：Matplotlib初相识.pdf"),
    PyPDFLoader("docs/matplotlib/第二回：艺术画笔见乾坤.pdf"),
    PyPDFLoader("docs/matplotlib/第三回：布局格式定方圆.pdf")
]
docs = []
for loader in loaders_chinese:
    docs.extend(loader.load())
```

在文档加载后，我们可以使用`RecursiveCharacterTextSplitter`(递归字符文本拆分器)来创建块。

```
# 分割文本
from langchain.text_splitter import RecursiveCharacterTextSplitter
text_splitter = RecursiveCharacterTextSplitter(
    chunk_size = 1500,  # 每个文本块的大小。这意味着每次切分文本时，会尽量使每个块包含 1500 个字符。
    chunk_overlap = 150  # 每个文本块之间的重叠部分。
)

splits = text_splitter.split_documents(docs)

print(len(splits))
```

```
27
```

二、Embeddings

什么是`Embeddings`?

在机器学习和自然语言处理（NLP）中，`Embeddings`（嵌入）是一种将类别数据，如单词、句子或者整个文档，转化为实数向量的技术。这些实数向量可以被计算机更好地理解和处理。嵌入背后的主要想法是，相似或相关的对象在嵌入空间中的距离应该很近。

举个例子，我们可以使用词嵌入（word embeddings）来表示文本数据。在词嵌入中，每个单词被转换为一个向量，这个向量捕获了这个单词的语义信息。例如，"king" 和 "queen" 这两个单词在嵌入空间中的位置将会非常接近，因为它们的含义相似。而 "apple" 和 "orange" 也会很接近，因为它们都是水果。而 "king" 和 "apple" 这两个单词在嵌入空间中的距离就会比较远，因为它们的含义不同。

让我们取出我们的切分部分并对它们进行`Embedding`处理。

```
from langchain.embeddings.openai import OpenAIEmbeddings
embedding = OpenAIEmbeddings()
```

在使用真实文档数据的例子之前，让我们用几个测试案例的句子来试试，以便了解`embedding`。

下面有几个示例句子，其中前两个非常相似，第三个与之无关。然后我们可以使用`embedding`类为每个句子创建一个`embedding`。

```
sentence1_chinese = "我喜欢狗"
sentence2_chinese = "我喜欢犬科动物"
sentence3_chinese = "外面的天气很糟糕"

embedding1_chinese = embedding.embed_query(sentence1_chinese)
embedding2_chinese = embedding.embed_query(sentence2_chinese)
embedding3_chinese = embedding.embed_query(sentence3_chinese)
```

然后我们可以使用`numpy`来比较它们，看看哪些最相似。

我们期望前两个句子应该非常相似。

然后，第一和第二个与第三个相比应该相差很大。

我们将使用点积来比较两个嵌入。

如果你不知道什么是点积，没关系。你只需要知道的重要一点是，分数越高句子越相似。

```
import numpy as np

np.dot(embedding1_chinese, embedding2_chinese)
```

```
0.9440614936689298
```

我们可以看到前两个`embedding`的分数相当高，为0.94。

```
np.dot(embedding1_chinese, embedding3_chinese)
```

```
0.792186975021313
```

如果我们将第一个`embedding`与第三个`embedding`进行比较，我们可以看到它明显较低，约为0.79。

```
np.dot(embedding2_chinese, embedding3_chinese)
```

```
0.7804109942586283
```

我们将第二个`embedding`和第三个`embedding`进行比较，我们可以看到它的分数大约为0.78。

三、Vectorstores

3.1 初始化Chroma

Langchain集成了超过30个不同的向量存储库。我们选择Chroma是因为它轻量级且数据存储在内存中，这使得它非常容易启动和开始使用。

首先我们指定一个持久化路径：

```
from langchain.vectorstores import Chroma

persist_directory_chinese = 'docs/chroma/matplotlib/'
```

如果该路径存在旧的数据库文件，可以通过以下命令删除：

```
!rm -rf './docs/chroma/matplotlib'  # 删除旧的数据库文件（如果文件夹中有文件的话）
```

接着从已加载的文档中创建一个向量数据库：

```
vectordb_chinese = Chroma.from_documents(
    documents=splits,
    embedding=embedding,
    persist_directory=persist_directory_chinese  # 允许我们将persist_directory目录保存到磁盘上
)
```

```
100%|██████████| 1/1 [00:01<00:00,  1.64s/it]
```

可以看到数据库长度也是30，这与我们之前的切分数量是一样的。现在让我们开始使用它。

```
print(vectordb_chinese._collection.count())
```

```
27
```

3.2 相似性搜索(Similarity Search)

首先我们定义一个需要检索答案的问题：

```
question_chinese = "Matplotlib是什么？"
```

接着调用已加载的向量数据库根据相似性检索答案：

```
docs_chinese = vectordb_chinese.similarity_search(question_chinese,k=3)
```

查看检索答案数量：

```
len(docs_chinese)
```

```
3
```

打印其 page_content 属性可以看到检索答案的文本：

```
print(docs_chinese[0].page_content)
```

```
第一回：Matplotlib 初相识
一、认识matplotlib
Matplotlib 是一个 Python 2D 绘图库，能够以多种硬拷贝格式和跨平台的交互式环境生成出版物质量的图形，用来绘制各种静态，动态，
交互式的图表。
Matplotlib 可用于 Python 脚本， Python 和 IPython Shell 、 Jupyter notebook ， Web 应用程序服务器和各种图形用户界面工具包等。
Matplotlib 是 Python 数据可视化库中的泰斗，它已经成为 python 中公认的数据可视化工具，我们所熟知的 pandas 和 seaborn 的绘图接口
其实也是基于 matplotlib 所作的高级封装。
为了对matplotlib 有更好的理解，让我们从一些最基本的概念开始认识它，再逐渐过渡到一些高级技巧中。
二、一个最简单的绘图例子
Matplotlib 的图像是画在 figure （如 windows ， jupyter 窗体）上的，每一个 figure 又包含了一个或多个 axes （一个可以指定坐标系的子区
域）。最简单的创建 figure 以及 axes 的方式是通过 pyplot.subplots命令，创建 axes 以后，可以使用 Axes.plot绘制最简易的折线图。
import matplotlib.pyplot as plt
import matplotlib as mpl
import numpy as np
fig, ax = plt.subplots()  # 创建一个包含一个 axes 的 figure
ax.plot([1, 2, 3, 4], [1, 4, 2, 3]);  # 绘制图像
Trick： 在jupyter notebook 中使用 matplotlib 时会发现，代码运行后自动打印出类似
<matplotlib.lines.Line2D at 0x23155916dc0>
这样一段话，这是因为 matplotlib 的绘图代码默认打印出最后一个对象。如果不想显示这句话，有以下三种方法，在本章节的代码示例
中你能找到这三种方法的使用。
•. 在代码块最后加一个分号 ；
```

•. 在代码块最后加一句 plt.show()
•. 在绘图时将绘图对象显式赋值给一个变量，如将 plt.plot([1, 2, 3, 4]) 改成 line =plt.plot([1, 2, 3, 4])
和MATLAB 命令类似，你还可以通过一种更简单的方式绘制图像， matplotlib.pyplot方法能够直接在当前 axes 上绘制图像，如果用户
未指定axes ， matplotlib 会帮你自动创建一个。所以上面的例子也可以简化为以下这一行代码。
line =plt.plot([1, 2, 3, 4], [1, 4, 2, 3])
三、Figure 的组成
现在我们来深入看一下 figure 的组成。通过一张 figure 解剖图，我们可以看到一个完整的 matplotlib 图像通常会包括以下四个层级，这些
层级也被称为容器（ container ），下一节会详细介绍。在 matplotlib 的世界中，我们将通过各种命令方法来操纵图像中的每一个部分，
从而达到数据可视化的最终效果，一副完整的图像实际上是各类子元素的集合。
Figure：顶层级，用来容纳所有绘图元素

在此之后，我们要确保通过运行vectordb.persist来持久化向量数据库，以便我们在未来的课程中使用。

```
vectordb_chinese.persist()
```

四、失败的情况(Failure modes)

这看起来很好，基本的相似性搜索很容易就能让你完成80%的工作。但是，可能会出现一些相似性搜索失败的情况。这里有一些可能出现的边缘情况————我们将在下一章节中修复它们。

4.1 重复块

```
question_chinese = "Matplotlib是什么？"

docs_chinese = vectordb_chinese.similarity_search(question_chinese,k=5)
```

请注意，我们得到了重复的块（因为索引中有重复的 `第一回：Matplotlib初相识.pdf`）。

语义搜索获取所有相似的文档，但不强制多样性。

`docs[0]` 和 `docs[1]` 是完全相同的。

```
print("docs[0]")
print(docs_chinese[0])

print("docs[1]")
print(docs_chinese[1])
```

docs[0]
page_content='第一回：Matplotlib 初相识\n一、认识matplotlib\nMatplotlib 是一个 Python 2D 绘图库，能够以多种硬拷贝格式和跨平台的交互式环境生成出版物质量的图形，用来绘制各种静态，动态，\n交互式的图表。\nMatplotlib 可用于 Python 脚本， Python 和 IPython Shell 、 Jupyter notebook ， Web 应用程序服务器和各种图形用户界面工具包等。\nMatplotlib 是 Python 数据可视化库中的泰斗，它已经成为 python 中公认的数据可视化工具，我们所熟知的 pandas 和 seaborn 的绘图接口\n其实也是基于 matplotlib 所作的高级封装。\n为了对matplotlib 有更好的理解，让我们从一些最基本的概念开始认识它，再逐渐过渡到一些高级技巧中。\n二、一个最简单的绘图例子\nMatplotlib 的图像是画在 figure （如 windows ， jupyter 窗体）上的，每一个 figure 又包含了一个或多个 axes （一个可以指定坐标系的子区\n域）。最简单的创建 figure 以及 axes 的方式是通过 pyplot.subplots命令，创建 axes 以后，可以使用 Axes.plot绘制最简易的折线图。\nimport matplotlib.pyplot as plt\nimport matplotlib as mpl\nimport numpy as np\nfig, ax = plt.subplots() # 创建一个包含一个 axes 的 figure\nax.plot([1, 2, 3, 4], [1, 4, 2, 3]); # 绘制图像\nTrick： 在jupyter notebook 中使用 matplotlib 时会发现，代码运行后自动打印出类似 <matplotlib.lines.Line2D at 0x23155916dc0>\n这样一段话，这是因为 matplotlib 的绘图代码默认打印出最后一个对象。如果不想显示这句话，有以下三种方法，在本章节的代码示例\n中你能找到这三种方法的使用。\n\x00. 在代码块最后加一个分号 ;\n\x00. 在代码块最后加一句 plt.show()\n\x00. 在绘图时将绘图对象显式赋值给一个变量，如将 plt.plot([1, 2, 3, 4]) 改成 line =plt.plot([1, 2, 3, 4])\n和MATLAB 命令类似，你还可以通过一种更简单的方式绘制图像，matplotlib.pyplot方法能够直接在当前 axes 上绘制图像，如果用户\n未指定axes ， matplotlib 会帮你自动创建一个。所以上面的例子也可以简化为以下这一行代码。\nline =plt.plot([1, 2, 3, 4], [1, 4, 2, 3]) \n三、Figure 的组成\n现在我们来深入看一下 figure 的组成。通过一张 figure 解剖图，我们可以看到一个完整的 matplotlib 图像通常会包括以下四个层级，这些\n层级也被称为容器（ container ），下一节会详细介绍。在 matplotlib 的世界中，我们将通过各种命令方法来操纵图像中的每一个部分，\n从而达到数据可视化的最终效果，一副完整的图像实际上是各类子元素的集合。\nFigure：顶层级，用来容纳所有绘图元素' metadata={'source': 'docs/matplotlib/第一回：Matplotlib初相识.pdf', 'page': 0}
docs[1]
page_content='第一回：Matplotlib 初相识\n一、认识matplotlib\nMatplotlib 是一个 Python 2D 绘图库，能够以多种硬拷贝格式和跨平台的交互式环境生成出版物质量的图形，用来绘制各种静态，动态，\n交互式的图表。\nMatplotlib 可用于 Python 脚本， Python 和 IPython Shell 、 Jupyter notebook ， Web 应用程序服务器和各种图形用户界面工具包等。\nMatplotlib 是 Python 数据可视化库中的泰斗，它已经成为 python 中公认的数据可视化工具，我们所熟知的 pandas 和 seaborn 的绘图接口\n其实也是基于 matplotlib 所作的高级封装。\n为了对matplotlib 有更好的理解，让我们从一些最基本的概念开始认识它，再逐渐过渡到一些高级技巧中。\n二、一个最简单的绘图例子\nMatplotlib 的图像是画在 figure （如 windows ， jupyter 窗体）上的，每一个 figure 又包含了一个或多个 axes （一个可以指定坐标系的子区\n域）。最简单的创建 figure 以及 axes 的方式是通过 pyplot.subplots命令，创建 axes 以后，可以使用 Axes.plot绘制最简易的折线图。\nimport matplotlib.pyplot as plt\nimport matplotlib as mpl\nimport numpy as np\nfig, ax = plt.subplots() # 创建一个包含一个 axes 的 figure\nax.plot([1, 2, 3, 4], [1, 4, 2, 3]); # 绘制图像\nTrick： 在jupyter notebook 中使用 matplotlib 时会发现，代码运行后自动打印出类似 <matplotlib.lines.Line2D at 0x23155916dc0>\n这样一段话，这是因为 matplotlib 的绘图代码默认打印出最后一个对象。如果不想显示这句话，有以下三种方法，在本章节的代码示例\n中你能找到这三种方法的使用。\n\x00. 在代码块最后加一个分号 ;\n\x00. 在代码块最后加一句 plt.show()\n\x00. 在绘图时将绘图对象显式赋值给一个变量，如将 plt.plot([1, 2, 3, 4]) 改成 line =plt.plot([1, 2, 3, 4])\n和MATLAB 命令类似，你还可以通过一种更简单的方式绘制图像，matplotlib.pyplot方法能够直接在当前 axes 上绘制图像，如果用户\n未指定axes ， matplotlib 会帮你自动创建一个。所以上面的例子也可以简化为以下这一行代码。\nline =plt.plot([1, 2, 3, 4], [1, 4, 2, 3]) \n三、Figure 的组成\n现在我们来深入看一下 figure 的组成。通过一张 figure 解剖图，我们可以看到一个完整的 matplotlib 图像通常会包括以下四个层级，这些\n层级也被称为容器（ container ），下一节会详细介绍。在 matplotlib 的世界中，我们将通过各种命令方法来操纵图像中的每一个部分，\n从而达到数据可视化的最终效果，一副完整的图像实际上是各类子元素的集合。\nFigure：顶层级，用来容纳所有绘图元素' metadata={'source': 'docs/matplotlib/第一回：Matplotlib初相识.pdf', 'page': 0}

4.2 检索错误答案

我们可以看到一种新的失败的情况。

下面的问题询问了关于第二讲的问题，但也包括了来自其他讲的结果。

```
question_chinese = "他们在第二讲中对Figure说了些什么？"
docs_chinese = vectordb_chinese.similarity_search(question_chinese,k=5)

for doc_chinese in docs_chinese:
    print(doc_chinese.metadata)
```

```
{'source': 'docs/matplotlib/第一回：Matplotlib初相识.pdf', 'page': 0}
{'source': 'docs/matplotlib/第一回：Matplotlib初相识.pdf', 'page': 0}
{'source': 'docs/matplotlib/第二回：艺术画笔见乾坤.pdf', 'page': 9}
{'source': 'docs/matplotlib/第二回：艺术画笔见乾坤.pdf', 'page': 10}
{'source': 'docs/matplotlib/第一回：Matplotlib初相识.pdf', 'page': 1}
```

可见，虽然我们询问的问题是第二讲，但第一个出现的答案却是第一讲的内容。而第三个答案才是我们想要的正确回答。

```
print(docs_chinese[2].page_content)
```

```
三、对象容器 - Object container
容器会包含一些 primitives，并且容器还有它自身的属性。
比如Axes Artist，它是一种容器，它包含了很多 primitives，比如Line2D，Text；同时，它也有自身的属性，比如 xscal，用来控制
X轴是linear还是log的。
1. Figure容器
matplotlib.figure.Figure是Artist最顶层的 container对象容器，它包含了图表中的所有元素。一张图表的背景就是在
Figure.patch的一个矩形 Rectangle。
当我们向图表添加 Figure.add_subplot()或者Figure.add_axes()元素时，这些都会被添加到
Figure.axes列表中。
fig = plt.figure()
ax1 = fig.add_subplot(211) # 作一幅2*1 的图，选择第 1 个子图
ax2 = fig.add_axes([0.1, 0.1, 0.7, 0.3]) # 位置参数，四个数分别代表了
(left,bottom,width,height)
print(ax1)
print(fig.axes) # fig.axes 中包含了 subplot 和 axes 两个实例 ，刚刚添加的
AxesSubplot(0.125,0.536818;0.775x0.343182)
[<AxesSubplot:>, <Axes:>]
由于Figure维持了current axes，因此你不应该手动的从 Figure.axes列表中添加删除元素，而是要通过 Figure.add_subplot()、
Figure.add_axes()来添加元素，通过 Figure.delaxes()来删除元素。但是你可以迭代或者访问
Figure.axes中的Axes，然后修改这个
Axes的属性。
比如下面的遍历 axes 里的内容，并且添加网格线：
fig = plt.figure()
ax1 = fig.add_subplot(211)
for ax in fig.axes:
    ax.grid(True)
```

```
Figure也有它自己的 text、line 、 patch 、 image。你可以直接通过 add primitive语句直接添加。但是注意 Figure默认的坐标系是以像
素为单位，你可能需要转换成 figure 坐标系： (0,0) 表示左下点， (1,1) 表示右上点。
Figure容器的常见属性：
Figure.patch属性：Figure 的背景矩形
Figure.axes属性：一个 Axes 实例的列表（包括 Subplot）
Figure.images属性：一个 FigureImages patch 列表
Figure.lines属性：一个 Line2D 实例的列表（很少使用）
Figure.legends属性：一个 Figure Legend 实例列表（不同于 Axes.legends）
Figure.texts属性：一个 Figure Text 实例列表
```

在接下来的章节中，我们将探讨的方法能够有效地解答这两个问题！

五、英文版

1.1 读取文档

```
from langchain.document_loaders import PyPDFLoader

# 加载 PDF
loaders = [
    # 故意添加重复文档，使数据混乱
    PyPDFLoader("docs/cs229_lectures/MachineLearning-Lecture01.pdf"),
    PyPDFLoader("docs/cs229_lectures/MachineLearning-Lecture01.pdf"),
    PyPDFLoader("docs/cs229_lectures/MachineLearning-Lecture02.pdf"),
    PyPDFLoader("docs/cs229_lectures/MachineLearning-Lecture03.pdf")
]
docs = []
for loader in loaders:
    docs.extend(loader.load())
```

进行分割

```
# 分割文本
from langchain.text_splitter import RecursiveCharacterTextSplitter
text_splitter = RecursiveCharacterTextSplitter(
    chunk_size = 1500,  # 每个文本块的大小。这意味着每次切分文本时，会尽量使每个块包含 1500 个字符。
    chunk_overlap = 150  # 每个文本块之间的重叠部分。
)

splits = text_splitter.split_documents(docs)

print(len(splits))
```

```
209
```

2.1 Embedding

```
from langchain.embeddings.openai import OpenAIEmbeddings
import numpy as np

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)

print("Sentence 1 VS setence 2")
print(np.dot(embedding1, embedding2))
print("Sentence 1 VS setence 3")
print(np.dot(embedding1, embedding3))
print("Sentence 2 VS sentence 3")
print(np.dot(embedding2, embedding3))
```

```
Sentence 1 VS setence 2
0.9632026347895142
Sentence 1 VS setence 3
0.7711302839662464
Sentence 2 VS sentence 3
0.759699788340627
```

3.1 初始化 Chroma

```
from langchain.vectorstores import Chroma

persist_directory = 'docs/chroma/cs229_lectures/'

vectordb = Chroma.from_documents(
    documents=splits,
    embedding=embedding,
    persist_directory=persist_directory  # 允许我们将persist_directory目录保存到磁盘
上
)

print(vectordb._collection.count())
```

```
100%|██████████| 1/1 [00:02<00:00,  2.62s/it]
```

```
209
```

3.2 相似性检索

```
question = "is there an email i can ask for help"  # "有我可以寻求帮助的电子邮件吗"

docs = vectordb.similarity_search(question,k=3)

print("Length of docs: ", len(docs))
print("Page content:")
print(docs[0].page_content)
```

```
Length of docs:  3
Page content:
cs229-qa@cs.stanford.edu. This goes to an acc ount that's read by all the TAs and
me. So
rather than sending us email individually, if you send email to this account, it
will
actually let us get back to you maximally quickly with answers to your questions.

If you're asking questions about homework probl ems, please say in the subject
line which
assignment and which question the email refers to, since that will also help us
to route
your question to the appropriate TA or to me  appropriately and get the response
back to
you quickly.
Let's see. Skipping ahead — let's see — for homework, one midterm, one open and
term
project. Notice on the honor code. So one thi ng that I think will help you to
succeed and
do well in this class and even help you to enjoy this cla ss more is if you form
a study
group.
So start looking around where you' re sitting now or at the end of class today,
mingle a
little bit and get to know your classmates. I strongly encourage you to form
study groups
and sort of have a group of people to study with and have a group of your fellow
students
to talk over these concepts with. You can also  post on the class news group if
you want to
use that to try to form a study group.
But some of the problems sets in this cla ss are reasonably difficult.  People
that have
taken the class before may tell you they were very difficult. And just I bet it
would be
more fun for you, and you'd probably have a be tter learning experience if you
form a
```

持久化数据库

```
vectordb.persist()
```

4.1 重复块

```
question = "what did they say about matlab?"  # "他们对 matlab 有何评价？"

docs = vectordb.similarity_search(question,k=5)

print("docs[0]")
print(docs[0])

print("docs[1]")
print(docs[1])
```

```
docs[0]
page_content='those homeworks will be done in either MATLA B or in Octave, which is sort of – I \nknow some people call it a free ve rsion of MATLAB, which it sort  of is, sort of isn\'t.  \nSo I guess for those of you that haven\'t s een MATLAB before, and I know most of you \nhave, MATLAB is I guess part of the programming language that makes it very easy to write codes using matrices, to write code for numerical routines, to move data around, to \nplot data. And it\'s sort of an extremely easy to  learn tool to use for implementing a lot of \nlearning algorithms.  \nAnd in case some of you want to work on your  own home computer or something if you \ndon\'t have a MATLAB license, for the purposes of this class, there\'s also – [inaudible] \nwrite that down [inaudible] MATLAB – there\' s also a software package called Octave \nthat you can download for free off the Internet. And it has somewhat fewer features than MATLAB, but it\'s free, and for the purposes of  this class, it will work for just about \neverything. \nSo actually I, well, so yeah, just a side comment for those of you that haven\'t seen \nMATLAB before I guess, once a colleague of mine at a different university, not at \nStanford, actually teaches another machine l earning course. He\'s taught it for many years. \nSo one day, he was in his office, and an old student of his from, lik e, ten years ago came \ninto his office and he said, "Oh, professo r, professor, thank you so much for your' metadata={'source': 'docs/cs229_lectures/MachineLearning-Lecture01.pdf', 'page': 8}
docs[1]
page_content='those homeworks will be done in either MATLA B or in Octave, which is sort of – I \nknow some people call it a free ve rsion of MATLAB, which it sort  of is, sort of isn\'t.  \nSo I guess for those of you that haven\'t s een MATLAB before, and I know most of you \nhave, MATLAB is I guess part of the programming language that makes it very easy to write codes using matrices, to write code for numerical routines, to move data around, to \nplot data. And it\'s sort of an extremely easy to  learn tool to use for implementing a lot of \nlearning algorithms.  \nAnd in case some of you want to work on your  own home computer or something if you \ndon\'t have a MATLAB license, for the purposes of this class, there\'s also – [inaudible] \nwrite that down [inaudible] MATLAB – there\' s also a software package called Octave \nthat you can download for free off the Internet. And it has somewhat fewer features than MATLAB, but it\'s free, and for the purposes of  this class, it will work for just about \neverything. \nSo actually I, well, so yeah, just a side comment for those of you that haven\'t seen \nMATLAB before I guess, once a colleague of mine at a different university, not at \nStanford, actually teaches another machine l earning course. He\'s taught it for many years. \nSo one day, he was in his office, and an old student of his from, lik e, ten years ago came \ninto his office and he said, "Oh, professo r, professor, thank you so much for your' metadata={'source': 'docs/cs229_lectures/MachineLearning-Lecture01.pdf', 'page': 8}
```

4.2 检索错误答案

```python
question = "what did they say about regression in the third lecture?"  # "他们在第三讲中是怎么谈论回归的？"

docs = vectordb.similarity_search(question,k=5)

for doc in docs:
    print(doc.metadata)

print("docs-4:")
print(docs[4].page_content)
```

```
{'source': 'docs/cs229_lectures/MachineLearning-Lecture03.pdf', 'page': 0}
{'source': 'docs/cs229_lectures/MachineLearning-Lecture03.pdf', 'page': 14}
{'source': 'docs/cs229_lectures/MachineLearning-Lecture02.pdf', 'page': 0}
{'source': 'docs/cs229_lectures/MachineLearning-Lecture03.pdf', 'page': 6}
{'source': 'docs/cs229_lectures/MachineLearning-Lecture01.pdf', 'page': 8}
docs-4:
into his office and he said, "Oh, professo r, professor, thank you so much for
your
machine learning class. I learned so much from it. There's this stuff that I
learned in your
class, and I now use every day. And it's help ed me make lots of money, and
here's a
picture of my big house."
So my friend was very excited. He said, "W ow. That's great. I'm glad to hear
this
machine learning stuff was actually useful. So what was it that you learned? Was
it
logistic regression? Was it the PCA? Was it the data ne tworks? What was it that
you
learned that was so helpful?" And the student said, "Oh, it was the MATLAB."
So for those of you that don't know MATLAB yet, I hope you do learn it. It's not
hard,
and we'll actually have a short MATLAB tutori al in one of the discussion
sections for
those of you that don't know it.
Okay. The very last piece of logistical th ing is the discussion s ections. So
discussion
sections will be taught by the TAs, and atte ndance at discussion sections is
optional,
although they'll also be recorded and televi sed. And we'll use the discussion
sections
mainly for two things. For the next two or th ree weeks, we'll use the discussion
sections
to go over the prerequisites to this class or if some of you haven't seen
probability or
statistics for a while or maybe algebra, we'll go over those in the discussion
sections as a
refresher for those of you that want one.
```