第二章 文档加载

用户个人数据可以以多种形式呈现：PDF 文档、视频、网页等。基于 LangChain 提供给 LLM 访问用户个人数据的能力，首先要加载并处理用户的多样化、非结构化个人数据。在本章，我们首先介绍如何加载文档（包括文档、视频、网页等），这是访问个人数据的第一步。

让我们先从 PDF 文档开始。

一、PDF 文档

首先，我们将从以下链接加载一个PDF文档。这是 DataWhale 提供的开源教程，名为《Fantastic Matplotlib》。值得注意的是，在英文版本中，吴恩达教授使用的是他2009年的机器学习课程字幕文件作为示例。为了更适合中文读者，我们选择了上述中文教程作为示例。不过，在英文原版代码中，你仍可以找到吴恩达教授的机器学习课程文件作为参考。后续的代码实践也会遵循这一调整。

注意，要运行以下代码，你需要安装第三方库 pypdf：

```
!pip install -q pypdf
```

1.1 加载PDF文档

首先，我们将利用 `PyPDFLoader` 来对 PDF 文件进行读取和加载。

```
from langchain.document_loaders import PyPDFLoader

# 创建一个 PyPDFLoader Class 实例，输入为待加载的pdf文档路径
loader = PyPDFLoader("docs/matplotlib/第一回：Matplotlib初相识.pdf")

# 调用 PyPDFLoader Class 的函数 load对pdf文件进行加载
pages = loader.load()
```

1.2 探索加载的数据

一旦文档被加载，它会被存储在名为 `pages` 的变量里。此外，`pages` 的数据结构是一个 `List` 类型。为了确认其类型，我们可以借助Python内建的 `type` 函数来查看 `pages` 的确切数据类型。

```
print(type(pages))
```

```
<class 'list'>
```

通过输出 `pages` 的长度，我们可以轻松地了解该PDF文件包含的总页数。

```
print(len(pages))
```

```
3
```

在 `page` 变量中，每一个元素都代表一个文档，它们的数据类型是 `langchain.schema.Document`。

```
page = pages[0]
print(type(page))
```

```
<class 'langchain.schema.document.Document'>
```

`langchain.schema.Document`类型包含两个属性：

1. `page_content`：包含该文档页面的内容。

```
print(page.page_content[0:500])
```

```
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
```

2. `meta_data`：为文档页面相关的描述性数据。

```
print(page.metadata)
```

```
{'source': 'docs/matplotlib/第一回：Matplotlib初相识.pdf', 'page': 0}
```

二、YouTube音频

在第一部分的内容，我们已经探讨了如何加载 PDF 文档。在这部分的内容中，对于给定的 YouTube 视频链接，我们会详细讨论：

- 利用`langchain`加载工具，为指定的 YouTube 视频链接下载对应的音频至本地
- 通过`OpenAIWhisperPaser`工具，将这些音频文件转化为可读的文本内容

注意，要运行以下代码，你需要安装如下两个第三方库：

```
!pip -q install yt_dlp
!pip -q install pydub
```

2.1 加载Youtube音频文档

首先，我们将构建一个 `GenericLoader` 实例来对 Youtube 视频的下载到本地并加载。

```
from langchain.document_loaders.generic import GenericLoader
from langchain.document_loaders.parsers import OpenAIWhisperParser
from langchain.document_loaders.blob_loaders.youtube_audio import
YoutubeAudioLoader

url="https://www.youtube.com/watch?v=_PHdzsQaDgw"
save_dir="docs/youtube-zh/"

# 创建一个 GenericLoader Class 实例
loader = GenericLoader(
    #将链接url中的Youtube视频的音频下载下来,存在本地路径save_dir
    YoutubeAudioLoader([url],save_dir),

    #使用OpenAIWhisperPaser解析器将音频转化为文本
    OpenAIWhisperParser()
)

# 调用 GenericLoader Class 的函数 load对视频的音频文件进行加载
pages = loader.load()
```

```
[youtube] Extracting URL: https://www.youtube.com/watch?v=_PHdzsQaDgw
[youtube] _PHdzsQaDgw: Downloading webpage
[youtube] _PHdzsQaDgw: Downloading ios player API JSON
[youtube] _PHdzsQaDgw: Downloading android player API JSON
[youtube] _PHdzsQaDgw: Downloading m3u8 information
WARNING: [youtube] Failed to download m3u8 information: HTTP Error 429: Too Many
Requests
[info] _PHdzsQaDgw: Downloading 1 format(s): 140
[download] docs/youtube-zh//【2023年7月最新】ChatGPT注册教程，国内详细注册流程，支持中文
使用，chatgpt 中国怎么用？.m4a has already been downloaded
[download] 100% of    7.72MiB
[ExtractAudio] Not converting audio docs/youtube-zh//【2023年7月最新】ChatGPT注册教
程，国内详细注册流程，支持中文使用，chatgpt 中国怎么用？.m4a; file is already in target
format m4a
Transcribing part 1!
```

2.2 探索加载的数据

Youtube 音频文件加载得到的变量同上文类似，此处不再一一解释，通过类似代码可以展示加载数据：

```
print("Type of pages: ", type(pages))
print("Length of pages: ", len(pages))

page = pages[0]
print("Type of page: ", type(page))
print("Page_content: ", page.page_content[:500])
print("Meta Data: ", page.metadata)
```

```
Type of pages:  <class 'list'>
Length of pages:  1
Type of page:  <class 'langchain.schema.document.Document'>
Page_content:  大家好,欢迎来到我的频道 今天我们来介绍如何注册ChetGBT账号 之前我有介绍过一期如何注册ChetGBT账号 但是还是会有一些朋友在注册过程当中 遇到了一些问题 今天我们再来详细介绍最新的注册方法 我们先打开这个网站 这个网站的网址我会放到视频下方的评论区 大家可以直接点击打开 这个网站是需要翻墙才能打开 建议使用全局模式翻墙打开 可以选择台湾,新加坡,日本,美国节点 不要选择香港节点 我这里使用的是台湾节点 这个翻墙软件如果大家需要的话 我也会共享在视频的下方 另外浏览器需要开启无痕模式打开 这个就是打开新的无痕模式窗口 我们可以按快捷键,Ctrl键加Shift键加N 可以打开新的无痕模式窗口 然后用无痕模式窗口来打开这个网站 然后点击这里 然后会出现这个登录注册界面 如果没有显示这个界面 显示的是拒绝访问 那么就表示你使用的节点可能有问题 我们需要切换其他的节点 我们可以这样切换其他的节点 能够正常打开这个页面 表示节点是没问题的 我们可以点击注册 这里需要填一个邮箱 然后点击继续 然后需要输入密码 再点击继续 然后会出现这个提示 我们需要去收一封邮件 刷新一下 邮件已经收到了
Meta Data:  {'source': 'docs/youtube-zh/【2023年7月最新】ChatGPT注册教程，国内详细注册流程，支持中文使用，chatgpt 中国怎么用？.m4a', 'chunk': 0}
```

三、网页文档

在第二部分，我们对于给定的 YouTube 视频链接 (URL)，使用 LangChain 加载器将视频的音频下载到本地，然后使用 OpenAIWhisperPaser 解析器将音频转化为文本。

本部分，我们将研究如何处理网页链接（URLs）。为此，我们会以 GitHub 上的一个markdown格式文档为例，学习如何对其进行加载。

3.1 加载网页文档

首先，我们将构建一个`WebBaseLoader`实例来对网页进行加载。

```
from langchain.document_loaders import WebBaseLoader


# 创建一个 WebBaseLoader Class 实例
url = "https://github.com/datawhalechina/d2l-ai-solutions-manual/blob/master/docs/README.md"
header = {'User-Agent': 'python-requests/2.27.1',
          'Accept-Encoding': 'gzip, deflate, br',
          'Accept': '*/*',
          'Connection': 'keep-alive'}
loader = WebBaseLoader(web_path=url,header_template=header)

# 调用 WebBaseLoader Class 的函数 load对文件进行加载
pages = loader.load()
```

3.2 探索加载的数据

同理我们通过上文代码可以展示加载数据：

```
print("Type of pages: ", type(pages))
print("Length of pages: ", len(pages))

page = pages[0]
print("Type of page: ", type(page))
print("Page_content: ", page.page_content[:500])
print("Meta Data: ", page.metadata)
```

```
Type of pages:  <class 'list'>
Length of pages:  1
Type of page:  <class 'langchain.schema.document.Document'>
Page_content:  {"payload":{"allShortcutsEnabled":false,"fileTree":{"docs":
{"items":[{"name":"ch02","path":"docs/ch02","contentType":"directory"},
{"name":"ch03","path":"docs/ch03","contentType":"directory"},
{"name":"ch05","path":"docs/ch05","contentType":"directory"},
{"name":"ch06","path":"docs/ch06","contentType":"directory"},
{"name":"ch08","path":"docs/ch08","contentType":"directory"},
{"name":"ch09","path":"docs/ch09","contentType":"directory"},
{"name":"ch10","path":"docs/ch10","contentType":"directory"},{"na
Meta Data:  {'source': 'https://github.com/datawhalechina/d2l-ai-solutions-
manual/blob/master/docs/README.md'}
```

可以看到上面的文档内容包含许多冗余的信息。通常来讲，我们需要进行对这种数据进行进一步处理(Post Processing)。

```
import json
convert_to_json = json.loads(page.page_content)
extracted_markdow = convert_to_json['payload']['blob']['richText']
print(extracted_markdow)
```

```
动手学深度学习习题解答 {docsify-ignore-all}
    李沐老师的《动手学深度学习》是入门深度学习的经典书籍，这本书基于深度学习框架来介绍深度学习，书中代码可以做到“所学即所用”。对于一般的初学者来说想要把书中课后习题部分独立解答还是比较困难。本项目对《动手学深度学习》习题部分进行解答，作为该书的习题手册，帮助初学者快速理解书中内容。
使用说明
    动手学深度学习习题解答，主要完成了该书的所有习题，并提供代码和运行之后的截图，里面的内容是以深度学习的内容为前置知识，该习题解答的最佳使用方法是以李沐老师的《动手学深度学习》为主线，并尝试完成课后习题，如果遇到不会的，再来查阅习题解答。
    如果觉得解答不详细，可以点击这里提交你希望补充推导或者习题编号，我们看到后会尽快进行补充。
选用的《动手学深度学习》版本
书名：动手学深度学习（PyTorch版）
著者：阿斯顿·张、[美]扎卡里 C. 立顿、李沐、[德]亚历山大·J.斯莫拉
译者：何孝霆、瑞潮儿·胡
出版社：人民邮电出版社
版次：2023年2月第1版

项目结构
codes----------------------------------------------习题代码
docs-----------------------------------------------习题解答
notebook-------------------------------------------习题解答JupyterNotebook格式
requirements.txt-----------------------------------运行环境依赖包

关注我们
```

扫描下方二维码关注公众号：Datawhale

Datawhale，一个专注于AI领域的学习圈子。初衷是for the learner，和学习者一起成长。目前加入学习社群的人数已经数千人，组织了机器学习，深度学习，数据分析，数据挖掘，爬虫，编程，统计学，Mysql，数据竞赛等多个领域的内容学习，微信搜索公众号Datawhale可以加入我们。

四、Notion文档

- 点击Notion示例文档(https://yolospace.notion.site/Blendle-s-Employee-Handbook-e31bff7da17346ee99f531087d8b133f)右上方复制按钮(Duplicate)，复制文档到你的Notion空间
- 点击右上方 ··· 按钮，选择导出为Mardown&CSV。导出的文件将为zip文件夹
- 解压并保存mardown文档到本地路径`docs/Notion_DB/`

4.1 加载Notion Markdown文档

首先，我们将使用`NotionDirectoryLoader`来对Notion Markdown文档进行加载。

```
from langchain.document_loaders import NotionDirectoryLoader
loader = NotionDirectoryLoader("docs/Notion_DB")
pages = loader.load()
```

4.2 探索加载的数据

同理，使用上文代码：

```
print("Type of pages: ", type(pages))
print("Length of pages: ", len(pages))

page = pages[0]
print("Type of page: ", type(page))
print("Page_content: ", page.page_content[:500])
print("Meta Data: ", page.metadata)
```

```
Type of pages:  <class 'list'>
Length of pages:  51
Type of page:  <class 'langchain.schema.document.Document'>
Page_content:  # #letstalkaboutstress

Let’s talk about stress. Too much stress.

We know this can be a topic.

So let’s get this conversation going.

[Intro: two things you should know]
(#letstalkaboutstress%2064040a0733074994976118bbe0acc7fb/Intro%20two%20things%20you%20should%20know%20b5fd0c5393a9498b93396e79fe71e8bf.md)
```

```
[What is stress]
(#letstalkaboutstress%2064040a0733074994976118bbe0acc7fb/what%20is%20stress%20b19
8b685ed6a474ab14f6fafff7004b6.md)

[When is there too much stress?](#letstalkaboutstress%2
Meta Data:  {'source': 'docs/Notion_DB/#letstalkaboutstress
64040a0733074994976118bbe0acc7fb.md'}
```

五、英文版

1.1 加载 PDF 文档

```
from langchain.document_loaders import PyPDFLoader

# 创建一个 PyPDFLoader Class 实例，输入为待加载的pdf文档路径
loader = PyPDFLoader("docs/cs229_lectures/MachineLearning-Lecture01.pdf")

# 调用 PyPDFLoader Class 的函数 load对pdf文件进行加载
pages = loader.load()
```

1.2 探索加载的数据

```
print("Type of pages: ", type(pages))
print("Length of pages: ", len(pages))

page = pages[0]
print("Type of page: ", type(page))
print("Page_content: ", page.page_content[:500])
print("Meta Data: ", page.metadata)
```

```
Type of pages:  <class 'list'>
Length of pages:  22
Type of page:  <class 'langchain.schema.document.Document'>
Page_content:  MachineLearning-Lecture01
Instructor (Andrew Ng):  Okay. Good morning. Welcome to CS229, the machine
learning class. So what I wanna do today is ju st spend a little time going over
the logistics
of the class, and then we'll start to  talk a bit about machine learning.
By way of introduction, my name's  Andrew Ng and I'll be instru ctor for this
class. And so
I personally work in machine learning, and I' ve worked on it for about 15 years
now, and
I actually think that machine learning i
Meta Data:  {'source': 'docs/cs229_lectures/MachineLearning-Lecture01.pdf',
'page': 0}
```

2.1 加载 Youtube 音频

```
# 注：由于该视频较长，容易出现网络问题，此处没有运行，读者可自行运行探索

from langchain.document_loaders.generic import GenericLoader
from langchain.document_loaders.parsers import OpenAIWhisperParser
```

```
from langchain.document_loaders.blob_loaders.youtube_audio import
YoutubeAudioLoader

url="https://www.youtube.com/watch?v=jGwO_UgTS7I"
save_dir="docs/youtube/"

# 创建一个 GenericLoader Class 实例
loader = GenericLoader(
    #将链接url中的Youtube视频的音频下载下来,存在本地路径save_dir
    YoutubeAudioLoader([url],save_dir),

    #使用OpenAIWhisperPaser解析器将音频转化为文本
    OpenAIWhisperParser()
)

# 调用 GenericLoader Class 的函数 load对视频的音频文件进行加载
docs = loader.load()
```

2.2 探索加载的数据

```
print("Type of pages: ", type(pages))
print("Length of pages: ", len(pages))

page = pages[0]
print("Type of page: ", type(page))
print("Page_content: ", page.page_content[:500])
print("Meta Data: ", page.metadata)
```

3.1 加载网页文档

```
from langchain.document_loaders import WebBaseLoader


# 创建一个 WebBaseLoader Class 实例
url = "https://github.com/basecamp/handbook/blob/master/37signals-is-you.md"
header = {'User-Agent': 'python-requests/2.27.1',
          'Accept-Encoding': 'gzip, deflate, br',
          'Accept': '*/*',
          'Connection': 'keep-alive'}
loader = WebBaseLoader(web_path=url,header_template=header)

# 调用 WebBaseLoader Class 的函数 load对文件进行加载
pages = loader.load()
```

3.2 探索加载的数据

```
print("Type of pages: ", type(pages))
print("Length of pages: ", len(pages))

page = pages[0]
print("Type of page: ", type(page))
print("Page_content: ", page.page_content[:500])
print("Meta Data: ", page.metadata)
```

```
Type of pages:  <class 'list'>
Length of pages:  1
Type of page:  <class 'langchain.schema.document.Document'>
Page_content:  {"payload":{"allShortcutsEnabled":false,"fileTree":{"":{"items":
[{"name":"37signals-is-you.md","path":"37signals-is-
you.md","contentType":"file"},
{"name":"LICENSE.md","path":"LICENSE.md","contentType":"file"},
{"name":"README.md","path":"README.md","contentType":"file"},{"name":"benefits-
and-perks.md","path":"benefits-and-perks.md","contentType":"file"},{"name":"code-
of-conduct.md","path":"code-of-conduct.md","contentType":"file"},
{"name":"faq.md","path":"faq.md","contentType":"file"},{"name":"ge
Meta Data:  {'source':
'https://github.com/basecamp/handbook/blob/master/37signals-is-you.md'}
```

进行进一步处理

```
import json
convert_to_json = json.loads(page.page_content)
extracted_markdow = convert_to_json['payload']['blob']['richText']
print(extracted_markdow)
```

```
37signals Is You
Everyone working at 37signals represents 37signals. When a customer gets a
response from Merissa on support, Merissa is 37signals. When a customer reads a
tweet by Eron that our systems are down, Eron is 37signals. In those situations,
all the other stuff we do to cultivate our best image is secondary. What's right
in front of someone in a time of need is what they'll remember.
That's what we mean when we say marketing is everyone's responsibility, and that
it pays to spend the time to recognize that. This means avoiding the bullshit of
outage language and bending our policies, not just lending your ears. It means
taking the time to get the writing right and consider how you'd feel if you were
on the other side of the interaction.
The vast majority of our customers come from word of mouth and much of that word
comes from people in our audience. This is an audience we've been educating and
entertaining for 20 years and counting, and your voice is part of us now, whether
you like it or not! Tell us and our audience what you have to say!
This goes for tools and techniques as much as it goes for prose. 37signals not
only tries to out-teach the competition, but also out-share and out-collaborate.
We're prolific open source contributors through Ruby on Rails, Trix, Turbolinks,
Stimulus, and many other projects. Extracting the common infrastructure that
others could use as well is satisfying, important work, and we should continue to
do that.
It's also worth mentioning that joining 37signals can be all-consuming. We've
seen it happen. You dig 37signals, so you feel pressure to contribute, maybe
overwhelmingly so. The people who work here are some of the best and brightest in
our industry, so the self-imposed burden to be exceptional is real. But here's
the thing: stop it. Settle in. We're glad you love this job because we all do
too, but at the end of the day it's a job. Do your best work, collaborate with
your team, write, read, learn, and then turn off your computer and play with your
dog. We'll all be better for it.
```

4.1 加载 Notion 文档

```
from langchain.document_loaders import NotionDirectoryLoader
loader = NotionDirectoryLoader("docs/Notion_DB")
pages = loader.load()
```

4.2 探索加载的数据

```
print("Type of pages: ", type(pages))
print("Length of pages: ", len(pages))

page = pages[0]
print("Type of page: ", type(page))
print("Page_content: ", page.page_content[:500])
print("Meta Data: ", page.metadata)
```

```
Type of pages:  <class 'list'>
Length of pages:  51
Type of page:  <class 'langchain.schema.document.Document'>
Page_content:  # #letstalkaboutstress

Let’s talk about stress. Too much stress.

We know this can be a topic.

So let’s get this conversation going.

[Intro: two things you should know]
(#letstalkaboutstress%2064040a0733074994976118bbe0acc7fb/Intro%20two%20things%20y
ou%20should%20know%20b5fd0c5393a9498b93396e79fe71e8bf.md)

[What is stress]
(#letstalkaboutstress%2064040a0733074994976118bbe0acc7fb/what%20is%20stress%20b19
8b685ed6a474ab14f6fafff7004b6.md)

[When is there too much stress?](#letstalkaboutstress%2
Meta Data:  {'source': 'docs/Notion_DB/#letstalkaboutstress
64040a0733074994976118bbe0acc7fb.md'}
```