Document Splitting

```
In [1]: 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']
```

```
In [2]: from langchain.text_splitter import RecursiveCharacterTextSplitter, Characte
```

```
In [3]: chunk_size =26
        chunk_overlap = 4
```

```
In [4]: r_splitter = RecursiveCharacterTextSplitter(
            chunk_size=chunk_size,
            chunk_overlap=chunk_overlap
        )
        c_splitter = CharacterTextSplitter(
            chunk_size=chunk_size,
            chunk_overlap=chunk_overlap
        )
```

Why doesn't this split the string below?

```
In [5]: text1 = 'abcdefghijklmnopqrstuvwxyz'
```

```
In [6]: r_splitter.split_text(text1)
```

```
['abcdefghijklmnopqrstuvwxyz']
```

```
In [7]: text2 = 'abcdefghijklmnopqrstuvwxyzabcdefg'
```

```
In [8]: r_splitter.split_text(text2)
```

```
['abcdefghijklmnopqrstuvwxyz', 'wxyzabcdefg']
```

Ok, this splits the string but we have an overlap specified as 5, but it looks like 3? (try an even number)

```
In [9]: text3 = "a b c d e f g h i j k l m n o p q r s t u v w x y z"
In [10]: r_splitter.split_text(text3)
```

```
['a b c d e f g h i j k l m', 'l m n o p q r s t u v w x', 'w x y z']
```

```
In [11]: c_splitter.split_text(text3)
```

```
['a b c d e f g h i j k l m n o p q r s t u v w x y z']
```

```
In [12]: c_splitter = CharacterTextSplitter(
             chunk_size=chunk_size,
             chunk_overlap=chunk_overlap,
             separator = ' '
         )
         c_splitter.split_text(text3)
```

```
['a b c d e f g h i j k l m', 'l m n o p q r s t u v w x', 'w x y z']
```

Try your own examples!

Recursive splitting details

`RecursiveCharacterTextSplitter` is recommended for generic text.

```
In [13]: some_text = """When writing documents, writers will use document structure t
         This can convey to the reader, which idea's are related. For example, closel
         are in sentences. Similar ideas are in paragraphs. Paragraphs form a documer
         Paragraphs are often delimited with a carriage return or two carriage returr
         Carriage returns are the "backslash n" you see embedded in this string. \
         Sentences have a period at the end, but also, have a space.\
         and words are separated by space."""
```

```
In [14]: len(some_text)
```

```
496
```

```
In [15]: c_splitter = CharacterTextSplitter(
             chunk_size=450,
             chunk_overlap=0,
             separator = ' '
         )
         r_splitter = RecursiveCharacterTextSplitter(
             chunk_size=450,
             chunk_overlap=0,
             separators=["\n\n", "\n", " ", ""]
         )
```

```
In [16]: c_splitter.split_text(some_text)
```

```
['When writing documents, writers will use document structure to group conten
t. This can convey to the reader, which idea\'s are related. For example, clo
sely related ideas are in sentances. Similar ideas are in paragraphs. Paragra
phs form a document. \n\n Paragraphs are often delimited with a carriage retu
rn or two carriage returns. Carriage returns are the "backslash n" you see em
bedded in this string. Sentences have a period at the end, but also,',
 'have a space.and words are separated by space.']
```

```
In [17]: r_splitter.split_text(some_text)
```

```
["When writing documents, writers will use document structure to group conten
t. This can convey to the reader, which idea's are related. For example, clos
ely related ideas are in sentances. Similar ideas are in paragraphs. Paragrap
hs form a document.",
 'Paragraphs are often delimited with a carriage return or two carriage retur
ns. Carriage returns are the "backslash n" you see embedded in this string. S
entences have a period at the end, but also, have a space.and words are separ
ated by space.']
```

Let's reduce the chunk size a bit and add a period to our separators:

```
In [18]: r_splitter = RecursiveCharacterTextSplitter(
             chunk_size=150,
             chunk_overlap=0,
             separators=["\n\n", "\n", "\. ", " ", ""]
         )
         r_splitter.split_text(some_text)
```

```
["When writing documents, writers will use document structure to group conten
t. This can convey to the reader, which idea's are related",
 '. For example, closely related ideas are in sentances. Similar ideas are in
paragraphs. Paragraphs form a document.',
 'Paragraphs are often delimited with a carriage return or two carriage retur
ns',
 '. Carriage returns are the "backslash n" you see embedded in this string',
 '. Sentences have a period at the end, but also, have a space.and words are
separated by space.']
```

In [19]:
```
r_splitter = RecursiveCharacterTextSplitter(
    chunk_size=150,
    chunk_overlap=0,
    separators=["\n\n", "\n", "(?<=\. )", " ", ""]
)
r_splitter.split_text(some_text)
```

```
["When writing documents, writers will use document structure to group conten
t. This can convey to the reader, which idea's are related.",
 'For example, closely related ideas are in sentances. Similar ideas are in p
aragraphs. Paragraphs form a document.',
 'Paragraphs are often delimited with a carriage return or two carriage retur
ns.',
 'Carriage returns are the "backslash n" you see embedded in this string.',
 'Sentences have a period at the end, but also, have a space.and words are se
parated by space.']
```

In [20]:
```
from langchain.document_loaders import PyPDFLoader
loader = PyPDFLoader("docs/cs229_lectures/MachineLearning-Lecture01.pdf")
pages = loader.load()
```

In [21]:
```
from langchain.text_splitter import CharacterTextSplitter
text_splitter = CharacterTextSplitter(
    separator="\n",
    chunk_size=1000,
    chunk_overlap=150,
    length_function=len
)
```

In [22]:
```
docs = text_splitter.split_documents(pages)
```

In [23]:
```
len(docs)
```

```
77
```

In [24]:
```
len(pages)
```

```
22
```

In [25]:
```
from langchain.document_loaders import NotionDirectoryLoader
loader = NotionDirectoryLoader("docs/Notion_DB")
notion_db = loader.load()
```

In [26]:
```
docs = text_splitter.split_documents(notion_db)
```

In [27]:
```
len(notion_db)
```

```
52
```

```
In [28]: len(docs)
```

```
353
```

Token splitting

We can also split on token count explicity, if we want.

This can be useful because LLMs often have context windows designated in tokens.

Tokens are often ~4 characters.

```
In [29]: from langchain.text_splitter import TokenTextSplitter
```

```
In [30]: text_splitter = TokenTextSplitter(chunk_size=1, chunk_overlap=0)
```

```
In [31]: text1 = "foo bar bazzyfoo"
```

```
In [32]: text_splitter.split_text(text1)
```

```
['foo', ' bar', ' b', 'az', 'zy', 'foo']
```

```
In [33]: text_splitter = TokenTextSplitter(chunk_size=10, chunk_overlap=0)
```

```
In [34]: docs = text_splitter.split_documents(pages)
```

```
In [35]: docs[0]
```

```
Document(page_content='MachineLearning-Lecture01  \n', metadata={'source': 'd
ocs/cs229_lectures/MachineLearning-Lecture01.pdf', 'page': 0})
```

```
In [36]: pages[0].metadata
```

```
{'source': 'docs/cs229_lectures/MachineLearning-Lecture01.pdf', 'page': 0}
```

Context aware splitting

Chunking aims to keep text with common context together.

A text splitting often uses sentences or other delimiters to keep related text together but many documents (such as Markdown) have structure (headers) that can be explicitly used in splitting.

We can use `MarkdownHeaderTextSplitter` to preserve header metadata in our chunks, as show below.

```
In [37]: from langchain.document_loaders import NotionDirectoryLoader
         from langchain.text_splitter import MarkdownHeaderTextSplitter
```

In [38]:
```
markdown_document = """# Title\n\n \
## Chapter 1\n\n \
Hi this is Jim\n\n Hi this is Joe\n\n \
### Section \n\n \
Hi this is Lance \n\n
## Chapter 2\n\n \
Hi this is Molly"""
```

In [39]:
```
headers_to_split_on = [
    ("#", "Header 1"),
    ("##", "Header 2"),
    ("###", "Header 3"),
]
```

In [40]:
```
markdown_splitter = MarkdownHeaderTextSplitter(
    headers_to_split_on=headers_to_split_on
)
md_header_splits = markdown_splitter.split_text(markdown_document)
```

In [41]:
```
md_header_splits[0]
```

```
Document(page_content='Hi this is Jim  \nHi this is Joe', metadata={'Header
1': 'Title', 'Header 2': 'Chapter 1'})
```

In [42]:
```
md_header_splits[1]
```

```
Document(page_content='Hi this is Lance', metadata={'Header 1': 'Title', 'Hea
der 2': 'Chapter 1', 'Header 3': 'Section'})
```

Try on a real Markdown file, like a Notion database.

In [43]:
```
loader = NotionDirectoryLoader("docs/Notion_DB")
docs = loader.load()
txt = ' '.join([d.page_content for d in docs])
```

In [44]:
```
headers_to_split_on = [
    ("#", "Header 1"),
    ("##", "Header 2"),
]
markdown_splitter = MarkdownHeaderTextSplitter(
    headers_to_split_on=headers_to_split_on
)
```

In [45]:
```
md_header_splits = markdown_splitter.split_text(txt)
```

In [46]:
```
md_header_splits[0]
```

```
Document(page_content="This is a living document with everything we've learne
d working with people while running a startup. And, of course, we continue to
learn. Therefore it's a document that will continue to change.  \n**Everythin
g related to working at Blendle and the people of Blendle, made public.**  \n
These are the lessons from three years of working with the people of Blendle.
It contains everything from [how our leaders lead](https://www.notion.so/ecfb
7e647136468a9a0a32f1771a8f52?pvs=21) to [how we increase salaries](https://ww
w.notion.so/Salary-Review-e11b6161c6d34f5c9568bb3e83ed96b6?pvs=21), from [how
we hire](https://www.notion.so/Hiring-451bbcfe8d9b49438c0633326bb7af0a?pvs=2
1) and [fire](https://www.notion.so/Firing-5567687a2000496b8412e53cd58eed9d?p
vs=21) to [how we think people should give each other feedback](https://www.n
otion.so/Our-Feedback-Process-eb64f1de796b4350aeab3bc068e3801f?pvs=21) — and
much more.  \nWe've made this document public because we want to learn from y
ou. We're very much interested in your feedback (including weeding out typo's
and Dunglish ;)). Email us at hr@blendle.com. If you're starting your own com
pany or if you're curious as to how we do things at Blendle, we hope that our
employee handbook inspires you.  \nIf you want to work at Blendle you can che
ck our [job ads here](https://blendle.homerun.co/). If you want to be kept in
the loop about Blendle, you can sign up for [our behind the scenes newslette
r](https://blendle.homerun.co/yes-keep-me-posted/tr/apply?token=8092d4128c306
003d97dd3821bad06f2).", metadata={'Header 1': "Blendle's Employee Handbook"})
```

In [ ]: