Split code

CodeTextSplitter allows you to split your code with multiple language support. Import enum `Language` and specify the language.

```
from langchain.text_splitter import (
    RecursiveCharacterTextSplitter,
    Language,
)
```

```
# Full list of support languages
[e.value for e in Language]
```

```
    ['cpp',
     'go',
     'java',
     'js',
     'php',
     'proto',
     'python',
     'rst',
     'ruby',
     'rust',
     'scala',
     'swift',
     'markdown',
     'latex',
     'html',
     'sol',]
```

```
# You can also see the separators used for a given language
RecursiveCharacterTextSplitter.get_separators_for_language(Language.PYTHON
```

```
    ['\nclass ', '\ndef ', '\n\tdef ', '\n\n', '\n', ' ', '']
```

Python

Here's an example using the PythonTextSplitter

```
PYTHON_CODE = """
def hello_world():
    print("Hello, World!")

# Call the function
hello_world()
"""
python_splitter = RecursiveCharacterTextSplitter.from_language(
    language=Language.PYTHON, chunk_size=50, chunk_overlap=0
)
python_docs = python_splitter.create_documents([PYTHON_CODE])
python_docs
```

```
    [Document(page_content='def hello_world():\n    print("Hello,
World!")', metadata={}),
     Document(page_content='# Call the function\nhello_world()',
metadata={})]
```

JS

Here's an example using the JS text splitter

```
JS_CODE = """
function helloWorld() {
  console.log("Hello, World!");
}

// Call the function
helloWorld();
"""

js_splitter = RecursiveCharacterTextSplitter.from_language(
    language=Language.JS, chunk_size=60, chunk_overlap=0
)
js_docs = js_splitter.create_documents([JS_CODE])
js_docs
```

```
    [Document(page_content='function helloWorld() {\n
console.log("Hello, World!");\n}', metadata={}),
     Document(page_content='// Call the function\nhelloWorld();',
metadata={})]
```

Markdown

Here's an example using the Markdown text splitter.

~~~
markdown_text = """
# 🦜🔗 LangChain

⚡ Building applications with LLMs through composability ⚡

## Quick Install

```bash
# Hopefully this code block isn't split
pip install langchain
```

As an open source project in a rapidly developing field, we are
extremely open to contributions.
"""
~~~

```
md_splitter = RecursiveCharacterTextSplitter.from_language(
    language=Language.MARKDOWN, chunk_size=60, chunk_overlap=0
)
md_docs = md_splitter.create_documents([markdown_text])
md_docs
```

```
    [Document(page_content='# 🦜🔗 LangChain', metadata={}),
     Document(page_content='⚡ Building applications with LLMs through
composability ⚡', metadata={}),
     Document(page_content='## Quick Install', metadata={}),
     Document(page_content="```bash\n# Hopefully this code block isn't
split", metadata={}),
     Document(page_content='pip install langchain', metadata={}),
     Document(page_content='```', metadata={}),
     Document(page_content='As an open source project in a rapidly
developing field, we', metadata={}),
```

```
     Document(page_content='are extremely open to contributions.',
metadata={})]
```

Latex

Here's an example on Latex text

```
latex_text = """
\documentclass{article}

\begin{document}

\maketitle

\section{Introduction}
Large language models (LLMs) are a type of machine learning model that
can be trained on vast amounts of text data to generate human-like
language. In recent years, LLMs have made significant advances in a
variety of natural language processing tasks, including language
translation, text generation, and sentiment analysis.

\subsection{History of LLMs}
The earliest LLMs were developed in the 1980s and 1990s, but they were
limited by the amount of data that could be processed and the
computational power available at the time. In the past decade, however,
advances in hardware and software have made it possible to train LLMs
on massive datasets, leading to significant improvements in
performance.

\subsection{Applications of LLMs}
LLMs have many applications in industry, including chatbots, content
creation, and virtual assistants. They can also be used in academia for
research in linguistics, psychology, and computational linguistics.

\end{document}
"""
```

```
latex_splitter = RecursiveCharacterTextSplitter.from_language(
    language=Language.MARKDOWN, chunk_size=60, chunk_overlap=0
)
latex_docs = latex_splitter.create_documents([latex_text])
latex_docs
```

```
[Document(page_content='\\documentclass{article}\n\n\x08egin{document}\n\r
metadata={}),
     Document(page_content='\\section{Introduction}', metadata={}),
     Document(page_content='Large language models (LLMs) are a type of mac
learning', metadata={}),
     Document(page_content='model that can be trained on vast amounts of t
metadata={}),
     Document(page_content='generate human-like language. In recent years,
metadata={}),
     Document(page_content='made significant advances in a variety of natu
language', metadata={}),
     Document(page_content='processing tasks, including language translati
metadata={}),
     Document(page_content='generation, and sentiment analysis.', metadata
     Document(page_content='\\subsection{History of LLMs}', metadata={}),
     Document(page_content='The earliest LLMs were developed in the 1980s
metadata={}),
     Document(page_content='but they were limited by the amount of data th
metadata={}),
     Document(page_content='processed and the computational power availab
metadata={}),
     Document(page_content='time. In the past decade, however, advances ir
and', metadata={}),
     Document(page_content='software have made it possible to train LLMs c
metadata={}),
     Document(page_content='datasets, leading to significant improvements
metadata={}),
     Document(page_content='performance.', metadata={}),
     Document(page_content='\\subsection{Applications of LLMs}', metadata=
     Document(page_content='LLMs have many applications in industry, inclu
metadata={}),
     Document(page_content='chatbots, content creation, and virtual assist
metadata={}),
     Document(page_content='can also be used in academia for research in l
metadata={}),
     Document(page_content='psychology, and computational linguistics.', r
     Document(page_content='\\end{document}', metadata={})]
```

HTML

Here's an example using an HTML text splitter

```
html_text = """
<!DOCTYPE html>
<html>
    <head>
        <title>🦜️🔗 LangChain</title>
        <style>
            body {
                font-family: Arial, sans-serif;
            }
            h1 {
                color: darkblue;
            }
        </style>
    </head>
    <body>
        <div>
            <h1>🦜️🔗 LangChain</h1>
            <p>⚡ Building applications with LLMs through composability
⚡</p>
        </div>
        <div>
            As an open source project in a rapidly developing field, we
are extremely open to contributions.
        </div>
    </body>
</html>
"""
```

```
html_splitter = RecursiveCharacterTextSplitter.from_language(
    language=Language.HTML, chunk_size=60, chunk_overlap=0
)
html_docs = html_splitter.create_documents([html_text])
html_docs
```

```
    [Document(page_content='<!DOCTYPE html>\n<html>', metadata={}),
     Document(page_content='<head>\n        <title>🦜️🔗
LangChain</title>', metadata={}),
     Document(page_content='<style>\n            body {\n
font-family: Aria', metadata={}),
     Document(page_content='l, sans-serif;\n            }\n
h1 {', metadata={}),
     Document(page_content='color: darkblue;\n            }\n
</style>\n    </head', metadata={}),
     Document(page_content='>', metadata={}),
     Document(page_content='<body>', metadata={}),
     Document(page_content='<div>\n           <h1>🦜️🔗
LangChain</h1>', metadata={}),
     Document(page_content='<p>⚡ Building applications with LLMs
through composability ⚡', metadata={}),
     Document(page_content='</p>\n       </div>', metadata={}),
     Document(page_content='<div>\n           As an open source
project in a rapidly dev', metadata={}),
     Document(page_content='eloping field, we are extremely open to
contributions.', metadata={}),
     Document(page_content='</div>\n    </body>\n</html>', metadata=
{})]
```

Solidity

Here's an example using the Solidity text splitter

```
SOL_CODE = """
pragma solidity ^0.8.20;
contract HelloWorld {
   function add(uint a, uint b) pure public returns(uint) {
       return a + b;
   }
}
"""

sol_splitter = RecursiveCharacterTextSplitter.from_language(
    language=Language.SOL, chunk_size=128, chunk_overlap=0
)
sol_docs = sol_splitter.create_documents([SOL_CODE])
sol_docs
```

```
[
    Document(page_content='pragma solidity ^0.8.20;', metadata={}),
    Document(page_content='contract HelloWorld {\n   function add(uint
a, uint b) pure public returns(uint) {\n       return a + b;\n   }\n}',
metadata={})
]
```