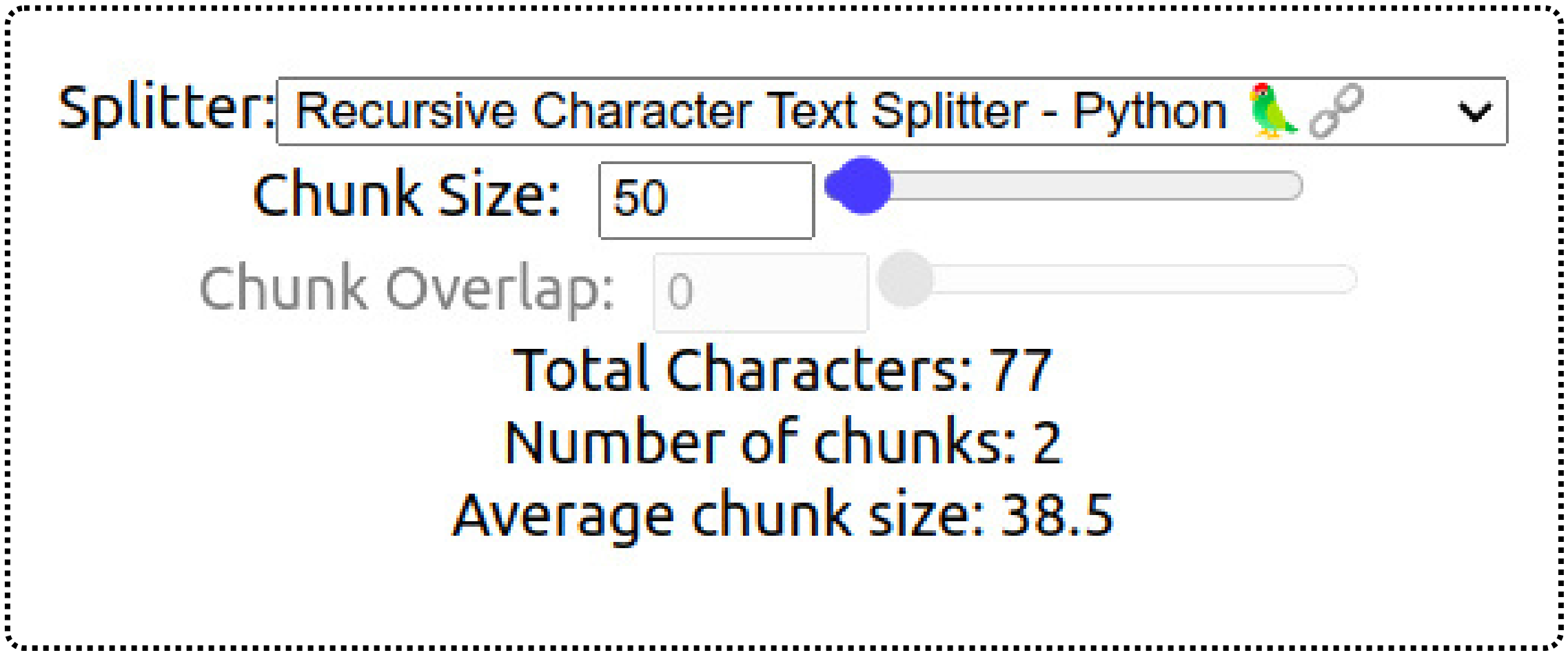
Analytics Vidhya

Mastering RAG

Document Specific Chunking Using LangChain




```
%pip install -qU langchain-text-splitters
from langchain_text_splitters import
(Language,RecursiveCharacterTextSplitter,)
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])
```

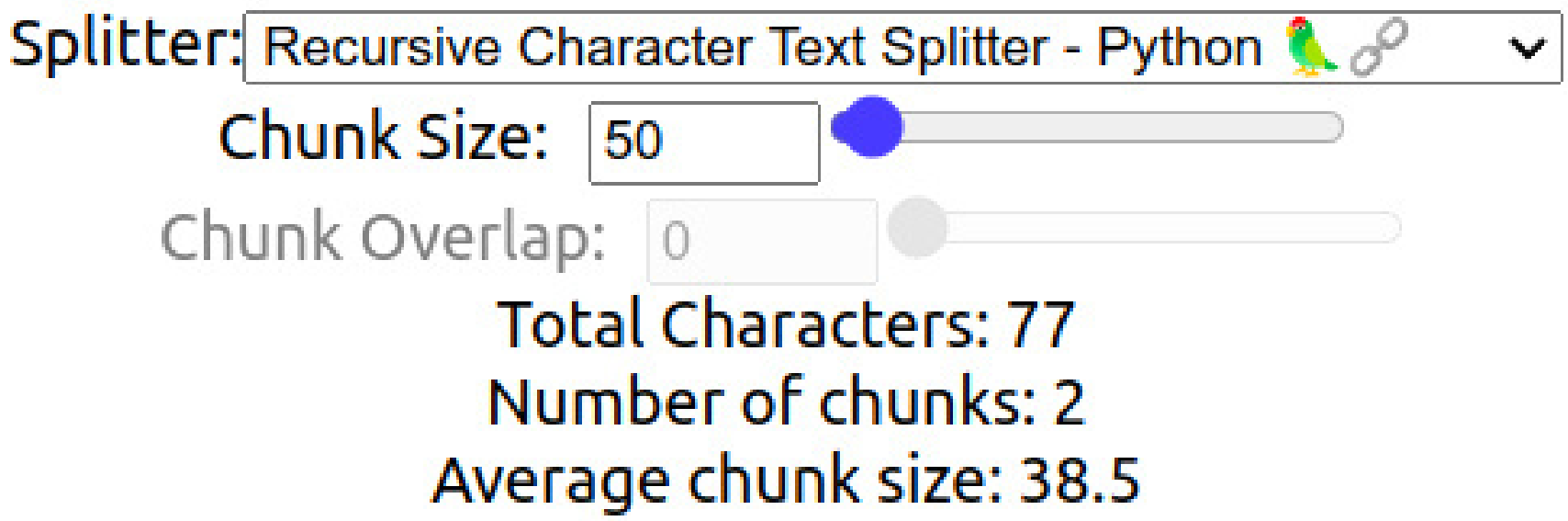
- Document-specific chunking is a strategy designed to tailor text-splitting methods to fit different data formats such as images, PDFs, or code snippets.

- Unlike generic chunking methods, which may not work effectively across various content types, document-specific chunking takes into account the unique structure and characteristics of each format to ensure meaningful segmentation.

- For instance, when dealing with Markdown, Python, or JavaScript files, chunking methods are adapted to use format-specific separators, such as headers in Markdown, function definitions in Python, or code blocks in JavaScript.

- This approach allows for more accurate and context-aware chunking, ensuring that key elements of the content remain intact and understandable.

- By adopting document-specific chunking, organizations and developers can efficiently process diverse data types while maintaining logical segmentation, and improving downstream tasks such as search, summarization, and analysis.



1. Python

```
%pip install -qU langchain-text-splitters
from langchain_text_splitters import
(Language,RecursiveCharacterTextSplitter,)
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])
```

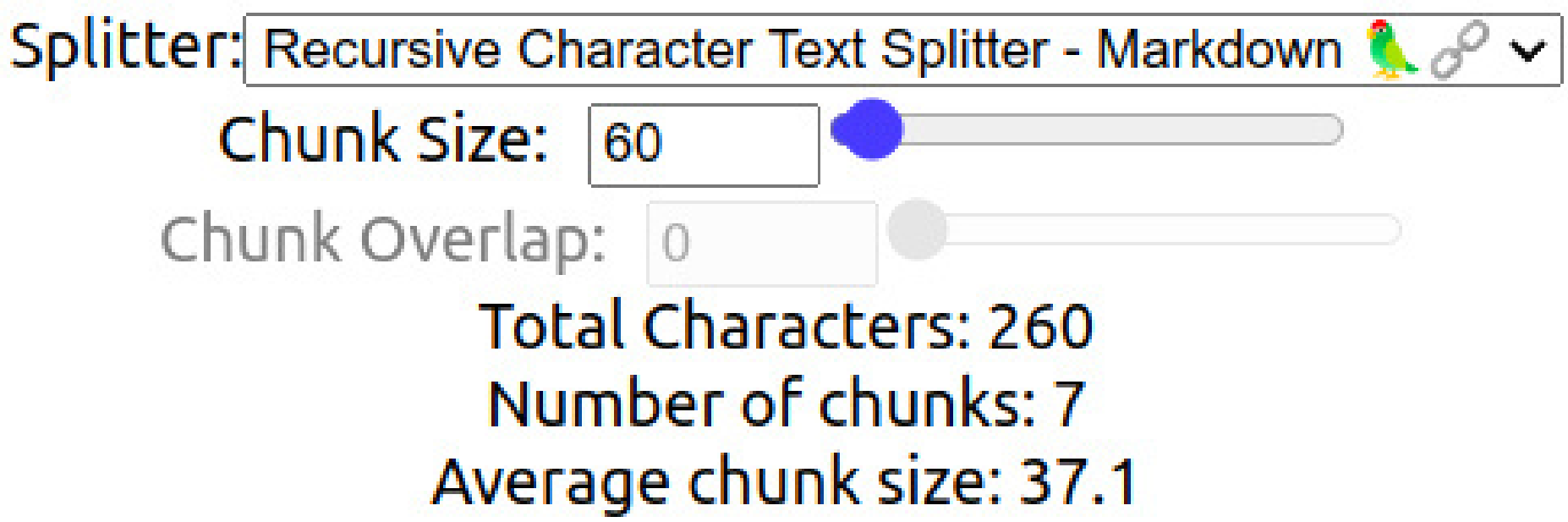
```
python_docs
```

Output

```
[Document(metadata={}, page_content='def
hello_world():\n    print("Hello,
 World!")'),

 Document(metadata={}, page_content='# Call the
function\nhello_world()')]
```

2. Markdown




```
# 🦜🔗 LangChain⚡ Building applications with LLMs through composability ⚡## What is
LangChain?# Hopefully this code block isn't splitLangChain is a framework for...As an open-
source project in a rapidly developing field, weare extremely open to contributions.
```

```
%pip install -qU langchain-text-splitters
from langchain_text_splitters
import(Language,RecursiveCharacterTextSplitter)

markdown_text = """# 🦜🔗 LangChain
⚡ Building applications with LLMs through composability ⚡
## What is LangChain?
# Hopefully this code block isn't split
LangChain is a framework for...
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
```

Output

[Document(metadata={}, page_content='# 🦜🔗 LangChain'),

Document(metadata={}, page_content='⚡ Building applications with LLMs through composability ⚡'),

Document(metadata={}, page_content='## What is LangChain?'),

Document(metadata={}, page_content="# Hopefully this code block isn't split"),

Document(metadata={}, page_content='LangChain is a framework for...'),

Document(metadata={}, page_content='As an open-source project in a rapidly developing field, we'),

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