Markdown

> Markdown is a lightweight markup language for creating formatted text using a plain-text editor.

This covers how to load `Markdown` documents into a document format that we can use downstream.

```
# !pip install unstructured > /dev/null
```

```
from langchain_community.document_loaders
import UnstructuredMarkdownLoader
```

```
markdown_path = "../../../../../README.md"
loader =
UnstructuredMarkdownLoader(markdown_path)
```

```
data = loader.load()
```

```
data
```

```
[Document(page_content="ð\x9f¦\x9cï¸ \x8fð\x9f”\
LangChain\n\nâ\x9a¡ Building applications with
LLMs through composability â\x9a¡\n\nLooking fo
the JS/TS version? Check out
LangChain.js.\n\nProduction Support: As you mov
your LangChains into production, we'd love to
offer more comprehensive support.\nPlease fill
this form and we'll set up a dedicated support
Slack channel.\n\nQuick Install\n\npip install
langchain\nor\nconda install langchain -c conda
forge\n\nð\x9f¤” What is this?\n\nLarge languag
models (LLMs) are emerging as a transformative
technology, enabling developers to build
applications that they previously could not.
However, using these LLMs in isolation is often
insufficient for creating a truly powerful app
the real power comes when you can combine them
with other sources of computation or
knowledge.\n\nThis library aims to assist in th
development of those types of applications. Com
examples of these applications include:\n\nâ\x9
Question Answering over specific
documents\n\nDocumentation\n\nEnd-to-end Exampl
Question Answering over Notion Database\n\nð\x9
Chatbots\n\nDocumentation\n\nEnd-to-end Example
Chat-LangChain\n\nð\x9f¤\x96
```

```
Agents\n\nDocumentation\n\nEnd-to-end Example:
GPT+WolframAlpha\n\nð\x9f“\x96
Documentation\n\nPlease see here for full
documentation on:\n\nGetting started
(installation, setting up the environment, simp
examples)\n\nHow-To examples (demos, integratio
helper functions)\n\nReference (full API
docs)\n\nResources (high-level explanation of c
concepts)\n\nð\x9f\x9a\x80 What can this help
with?\n\nThere are six main areas that LangChai
is designed to help with.\nThese are, in
increasing order of complexity:\n\nð\x9f“\x83 L
and Prompts:\n\nThis includes prompt management
prompt optimization, a generic interface for al
LLMs, and common utilities for working with
LLMs.\n\nð\x9f”\x97 Chains:\n\nChains go beyond
single LLM call and involve sequences of calls
(whether to an LLM or a different utility).
LangChain provides a standard interface for
chains, lots of integrations with other tools, 
end-to-end chains for common
applications.\n\nð\x9f“\x9a Data Augmented
Generation:\n\nData Augmented Generation involv
specific types of chains that first interact wi
an external data source to fetch data for use i
the generation step. Examples include
summarization of long pieces of text and
question/answering over specific data
sources.\n\nð\x9f¤\x96 Agents:\n\nAgents involv
an LLM making decisions about which Actions to
take, taking that Action, seeing an Observation
```

and repeating that until done. LangChain provid
a standard interface for agents, a selection of
agents to choose from, and examples of end-to-e
agents.\n\nð\x9f§\xa0 Memory:\n\nMemory refers
persisting state between calls of a chain/agent
LangChain provides a standard interface for
memory, a collection of memory implementations,
and examples of chains/agents that use
memory.\n\nð\x9f§\x90 Evaluation:\n\n[BETA]
Generative models are notoriously hard to evalu
with traditional metrics. One new way of
evaluating them is using language models
themselves to do the evaluation. LangChain
provides some prompts/chains for assisting in
this.\n\nFor more information on these concepts
please see our full documentation.\n\nð\x9f’\x8
Contributing\n\nAs an open-source project in a
rapidly developing field, we are extremely open
contributions, whether it be in the form of a n
feature, improved infrastructure, or better
documentation.\n\nFor detailed information on h
to contribute, see here.", metadata={'source':
'../../../../../README.md'})]

Retain Elements

Under the hood, Unstructured creates different "elements" for different chunks of text. By default we combine those together,

but you can easily keep that separation by specifying `mode="elements"`.

```
loader = UnstructuredMarkdownLoader(markdown_path, mode="elements")
```

```
data = loader.load()
```

```
data[0]
```

```
Document(page_content='ð\x9f¦\x9cï¸\x8fð\x9f”\x
LangChain', metadata={'source':
'../../../../../README.md', 'page_number': 1,
'category': 'Title'})
```