Markdown

Contents

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

```
from langchain.document_loaders import UnstructuredMarkdownLoader
```

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

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

```
data
```

```
[Document(page_content="ð\x9f¦\x9cï¸\x8fð\x9f”\x97 LangChain\n\nâ\x9a¡ Building
applications with LLMs through composability â\x9a¡\n\nProduction Support: As you
move your LangChains into production, we'd love to offer more comprehensive
support.\nPlease fill out this form and we'll set up a dedicated support Slack
channel.\n\nQuick Install\n\npip install langchain\n\nð\x9f¤” What is this?
\n\nLarge language models (LLMs) are emerging as a transformative technology,
enabling\ndevelopers to build applications that they previously could not.\nBut
using these LLMs in isolation is often not enough to\ncreate a truly powerful app
- the real power comes when you can combine them with other sources of computation
or knowledge.\n\nThis library is aimed at assisting in the development of those
types of applications. Common examples of these types of applications
include:\n\nâ\x9d“ Question Answering over specific
documents\n\nDocumentation\n\nEnd-to-end Example: Question Answering over Notion
Database\n\nð\x9f’¬ 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,
simple examples)\n\nHow-To examples (demos, integrations, helper
functions)\n\nReference (full API docs)\n  Resources (high-level explanation of
```

Skip to main content

```
areas that LangChain is designed to help with.\nThese are, in increasing order of
complexity:\n\nð\x9f“\x83 LLMs and Prompts:\n\nThis includes prompt management,
prompt optimization, generic interface for all LLMs, and common utilities for
working with LLMs.\n\nð\x9f”\x97 Chains:\n\nChains go beyond just a single LLM
call, and are sequences of calls (whether to an LLM or a different utility).
LangChain provides a standard interface for chains, lots of integrations with
other tools, and end-to-end chains for common applications.\n\nð\x9f“\x9a Data
Augmented Generation:\n\nData Augmented Generation involves specific types of
chains that first interact with an external datasource to fetch data to use in the
generation step. Examples of this include summarization of long pieces of text and
question/answering over specific data sources.\n\nð\x9f¤\x96 Agents:\n\nAgents
involve an LLM making decisions about which Actions to take, taking that Action,
seeing an Observation, and repeating that until done. LangChain provides a
standard interface for agents, a selection of agents to choose from, and examples
of end to end agents.\n\nð\x9f§\xa0 Memory:\n\nMemory is the concept of 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 evaluate 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’\x81 Contributing\n\nAs an open source
project in a rapidly developing field, we are extremely open to contributions,
whether it be in the form of a new feature, improved infra, or better
documentation.\n\nFor detailed information on how to contribute, see here.",
lookup_str='', metadata={'source': '../../../../README.md'}, lookup_index=0)]
```

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("../../../../README.md", mode="elements")
```

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

```
data[0]
```

```
Document(page_content='ð\x9f¦\x9cï¸\x8fð\x9f”\x97 LangChain', lookup_str='',
metadata={'source': '../../../../README.md', 'page_number': 1, 'category':
'UncategorizedText'}, lookup_index=0)
```