HTMLHeaderTextSplitter

Description and motivation

Similar in concept to the

`MarkdownHeaderTextSplitter`, the `HTMLHeaderTextSplitter` is a “structure-aware” chunker that splits text at the element level and adds metadata for each header “relevant” to any given chunk. It can return chunks element by element or combine elements with the same metadata, with the objectives of (a) keeping related text grouped (more or less) semantically and (b) preserving context-rich information encoded in document structures. It can be used with other text splitters as part of a chunking pipeline.

Usage examples

1) With an HTML string:

```
from langchain.text_splitter import HTMLHeaderT

html_string = """
<!DOCTYPE html>
<html>
<body>
    <div>
        <h1>Foo</h1>
        <p>Some intro text about Foo.</p>
        <div>
            <h2>Bar main section</h2>
            <p>Some intro text about Bar.</p>
            <h3>Bar subsection 1</h3>
            <p>Some text about the first subtop
            <h3>Bar subsection 2</h3>
            <p>Some text about the second subto
        </div>
        <div>
            <h2>Baz</h2>
            <p>Some text about Baz</p>
        </div>
        <br>
        <p>Some concluding text about Foo</p>
    </div>
</body>
</html>
"""

headers_to_split_on = [
    ("h1", "Header 1"),
```

```
    ("h2", "Header 2"),
    ("h3", "Header 3"),
]

html_splitter =
HTMLHeaderTextSplitter(headers_to_split_on=head
html_header_splits = html_splitter.split_text(h
html_header_splits
```

```
[Document(page_content='Foo'),
 Document(page_content='Some intro text about
Foo.  \nBar main section Bar subsection 1 Bar
subsection 2', metadata={'Header 1': 'Foo'}),
 Document(page_content='Some intro text about
Bar.', metadata={'Header 1': 'Foo', 'Header
2': 'Bar main section'}),
 Document(page_content='Some text about the
first subtopic of Bar.', metadata={'Header
1': 'Foo', 'Header 2': 'Bar main section',
'Header 3': 'Bar subsection 1'}),
 Document(page_content='Some text about the
second subtopic of Bar.', metadata={'Header
1': 'Foo', 'Header 2': 'Bar main section',
'Header 3': 'Bar subsection 2'}),
 Document(page_content='Baz', metadata=
{'Header 1': 'Foo'}),
 Document(page_content='Some text about Baz',
metadata={'Header 1': 'Foo', 'Header 2':
'Baz'}),
```

```
 Document(page_content='Some concluding text
about Foo', metadata={'Header 1': 'Foo'})]
```

2) Pipelined to another splitter, with html loaded from a web URL:

```
from langchain.text_splitter import
RecursiveCharacterTextSplitter

url = "https://plato.stanford.edu/entries/goede

headers_to_split_on = [
    ("h1", "Header 1"),
    ("h2", "Header 2"),
    ("h3", "Header 3"),
    ("h4", "Header 4"),
]

html_splitter =
HTMLHeaderTextSplitter(headers_to_split_on=head

# for local file use
html_splitter.split_text_from_file(<path_to_fil
html_header_splits = html_splitter.split_text_f

chunk_size = 500
chunk_overlap = 30
text_splitter = RecursiveCharacterTextSplitter(
    chunk_size=chunk_size, chunk_overlap=chunk_
)
```

```
# Split
splits = text_splitter.split_documents(html_hea
splits[80:85]
```

```
[Document(page_content='We see that Gödel
first tried to reduce the consistency problem
for analysis to that of arithmetic. This
seemed to require a truth definition for
arithmetic, which in turn led to paradoxes,
such as the Liar paradox (“This sentence is
false”) and Berry’s paradox (“The least
number not defined by an expression
consisting of just fourteen English words”).
Gödel then noticed that such paradoxes would
not necessarily arise if truth were replaced
by provability. But this means that
arithmetic truth', metadata={'Header 1':
'Kurt Gödel', 'Header 2': '2. Gödel’s
Mathematical Work', 'Header 3': '2.2 The
Incompleteness Theorems', 'Header 4': '2.2.1
The First Incompleteness Theorem'}),
 Document(page_content='means that arithmetic
truth and arithmetic provability are not co-
extensive — whence the First Incompleteness
Theorem.', metadata={'Header 1': 'Kurt
Gödel', 'Header 2': '2. Gödel’s Mathematical
Work', 'Header 3': '2.2 The Incompleteness
Theorems', 'Header 4': '2.2.1 The First
Incompleteness Theorem'}),
 Document(page_content='This account of
```

```
Gödel’s discovery was told to Hao Wang very
much after the fact; but in Gödel’s
contemporary correspondence with Bernays and
Zermelo, essentially the same description of
his path to the theorems is given. (See Gödel
2003a and Gödel 2003b respectively.) From
those accounts we see that the undefinability
of truth in arithmetic, a result credited to
Tarski, was likely obtained in some form by
Gödel by 1931. But he neither publicized nor
published the result; the biases logicians',
metadata={'Header 1': 'Kurt Gödel', 'Header
2': '2. Gödel’s Mathematical Work', 'Header
3': '2.2 The Incompleteness Theorems',
'Header 4': '2.2.1 The First Incompleteness
Theorem'}),
 Document(page_content='result; the biases
logicians had expressed at the time
concerning the notion of truth, biases which
came vehemently to the fore when Tarski
announced his results on the undefinability
of truth in formal systems 1935, may have
served as a deterrent to Gödel’s publication
of that theorem.', metadata={'Header 1':
'Kurt Gödel', 'Header 2': '2. Gödel’s
Mathematical Work', 'Header 3': '2.2 The
Incompleteness Theorems', 'Header 4': '2.2.1
The First Incompleteness Theorem'}),
 Document(page_content='We now describe the
proof of the two theorems, formulating
Gödel’s results in Peano arithmetic. Gödel
```

```
himself used a system related to that defined
in Principia Mathematica, but containing
Peano arithmetic. In our presentation of the
First and Second Incompleteness Theorems we
refer to Peano arithmetic as P, following
Gödel’s notation.', metadata={'Header 1':
'Kurt Gödel', 'Header 2': '2. Gödel’s
Mathematical Work', 'Header 3': '2.2 The
Incompleteness Theorems', 'Header 4': '2.2.2
The proof of the First Incompleteness
Theorem'})]
```

Limitations

There can be quite a bit of structural variation from one HTML document to another, and while `HTMLHeaderTextSplitter` will attempt to attach all “relevant” headers to any given chunk, it can sometimes miss certain headers. For example, the algorithm assumes an informational hierarchy in which headers are always at nodes “above” associated text, i.e. prior siblings, ancestors, and combinations thereof. In the following news article (as of the writing of this document), the document is structured such that the text of the top-level headline, while tagged “h1”, is in a *distinct* subtree from the text elements that we’d expect it to be *“above”*—so we can observe that the “h1” element and its associated text do not show up in the chunk

metadata (but, where applicable, we do see “h2” and its associated text):

```
url = "https://www.cnn.com/2023/09/25/weather/e
us-climate/index.html"

headers_to_split_on = [
    ("h1", "Header 1"),
    ("h2", "Header 2"),
]

html_splitter =
HTMLHeaderTextSplitter(headers_to_split_on=head
html_header_splits = html_splitter.split_text_f
print(html_header_splits[1].page_content[:500])
```

```
No two El NiÃ±o winters are the same, but
many have temperature and precipitation
trends in common.
Average conditions during an El NiÃ±o winter
across the continental US.
One of the major reasons is the position of
the jet stream, which often shifts south
during an El NiÃ±o winter. This shift
typically brings wetter and cooler weather to
the South while the North becomes drier and
warmer, according to NOAA.
```

```
Because the jet stream is essentially a river of air that storms flow through, the
```