
- 출처: LangChain 공식 문서 또는 해당 교재명
- 원본 URL: https://smith.langchain.com/hub/teddynote/summary-stuff-documents

---

`HTML 헤더 텍스트 분할 (HTMLHeaderTextSplitter)`

`HTMLHeaderTextSplitter`

- **`HTMLHeaderTextSplitter`**
  - `MarkdownHeaderTextSplitter`와 개념적으로 유사
  - ** `텍스트`를 `요소 수준`에서 `분할`하고 각 `헤더`에 대한 `메타데이터`를 `추가`하는 **`구조 인식` 청크 생성기**
  - 각 청크와 **`관련된`** 메타데이터 추가
- **`HTMLHeaderTextSplitter`**: `요소별`로 `청크`를 `반환`하거나 `동일한 메타데이터`를 가진 `요소`를 `결합` 가능
- 목표
  - **`관련 텍스트`를 `의미론적으로` (대략적으로) `그룹화`**
  - **`문서 구조`에 `인코딩된 컨텍스트` 풍부한 정보를 `보존`하는 것**

1) `HTML` 문자열을 사용하는 경우

- 사전에 `VS Code` 터미널에 설치할 것

```
pip install -qU langchain-text-splitters
```

- `headers_to_split_on` 리스트 = `분할 기준`이 되는 `헤더 태그`와 `해당 헤더`의 `이름`을 **`튜플 형태`로 지정**
- `HTMLHeaderTextSplitter` 객체 생성 → `headers_to_split_on` 매개변수에 `분할 기준 헤더 리스트` 전달

```python
from langchain_text_splitters import HTMLHeaderTextSplitter

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 subtopic of Bar.</p>
            <h3>Bar subsection 2</h3>
            <p>Some text about the second subtopic of Bar.</p>
        </div>
        <div>
            <h2>Baz</h2>
            <p>Some text about Baz</p>
        </div>
        <br>
        <p>Some concluding text about Foo</p>
    </div>
</body>
</html>
"""
```

```python
# 분할할 헤더 태그와 해당 헤더의 이름 지정하기

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

```python
# 지정된 헤더를 기준으로 HTML 텍스트를 분할하는 HTMLHeaderTextSplitter 객체 생성하기

html_splitter = HTMLHeaderTextSplitter(headers_to_split_on=headers_to_split_on)
```

```python
# HTML 문자열을 분할하여 결과를 html_header_splits 변수에 저장하기

html_header_splits = html_splitter.split_text(html_string)
```

```python
# 분할된 결과 출력하기

for header in html_header_splits:
    print(f"{header.page_content}")
    print(f"{header.metadata}", end="\n=====================\n")
```

- 셀 출력

```
Foo
{'Header 1': 'Foo'}
=====================
Some intro text about Foo.
{'Header 1': 'Foo'}
=====================
Bar main section
{'Header 1': 'Foo', 'Header 2': 'Bar main section'}
=====================
Some intro text about Bar.
{'Header 1': 'Foo', 'Header 2': 'Bar main section'}
=====================
Bar subsection 1
{'Header 1': 'Foo', 'Header 2': 'Bar main section', 'Header 3': 'Bar subsection
=====================
Some text about the first subtopic of Bar.
{'Header 1': 'Foo', 'Header 2': 'Bar main section', 'Header 3': 'Bar subsection
=====================
Bar subsection 2
{'Header 1': 'Foo', 'Header 2': 'Bar main section', 'Header 3': 'Bar subsection
=====================
Some text about the second subtopic of Bar.
{'Header 1': 'Foo', 'Header 2': 'Bar main section', 'Header 3': 'Bar subsection
=====================
Baz
{'Header 1': 'Foo', 'Header 2': 'Baz'}
=====================
Some text about Baz
Some concluding text about Foo
{'Header 1': 'Foo'}
=====================
```

---

2) 다른 splitter와 파이프라인으로 연결 → 웹 URL에서 HTML을 로드하는 경우

- 웹 URL로부터 HTML 콘텐츠를 로드 ⟶ 이를 다른 splitter와 파이프라인으로 연결하여 처리하는 과정

```
from langchain_text_splitters import RecursiveCharacterTextSplitter

# 분할할 텍스트의 URL 지정하기
url = "https://plato.stanford.edu/entries/goedel/"

# 분할할 HTML 헤더 태그와 해당 헤더의 이름 지정하기
headers_to_split_on = [
```

```python
        ("h2", "Header 2"),
        ("h3", "Header 3"),
        ("h4", "Header 4"),
    ]

    # HTML 헤더를 기준으로 텍스트를 분할하는 HTMLHeaderTextSplitter 객체 생성하기
    html_splitter = HTMLHeaderTextSplitter(headers_to_split_on=headers_to_split_on)

    # URL에서 텍스트를 가져와 HTML 헤더를 기준으로 분할하기
    html_header_splits = html_splitter.split_text_from_url(url)


    # 매개변수 설정하기
    chunk_size = 500                        # 텍스트를 분할할 청크의 크기 지정하기
    chunk_overlap = 30                      # 분할된 청크 간의 중복되는 문자 수 지정하기


    # 텍스트를 재귀적으로 분할하는 RecursiveCharacterTextSplitter 객체 생성하기
    text_splitter = RecursiveCharacterTextSplitter(
        chunk_size=chunk_size,              # 사전에 설정한 매개변수 가져오기
        chunk_overlap=chunk_overlap
    )

    # HTML 헤더로 분할된 텍스트를 다시 청크 크기에 맞게 분할하기
    splits = text_splitter.split_documents(html_header_splits)

    # 분할된 텍스트 중 80번째부터 85번째까지의 청크 출력해보기
    for header in splits[80:85]:
        print(f"{header.page_content}")
        print(f"{header.metadata}", end="\n=====================\n")
```

```
Incompleteness Theorem: \(P\) is if
\(P \vdash \neg \phi(\underline{n})\) for all \(n\)
implies \(P \not\vdash \exists x\phi(x)\).
Naturally this implies consistency and follows from the assumption
that the natural numbers satisfy the axioms of Peano arithmetic.
\(\omega\)-consistent
One of the main technical tools used in the proof is , a mechanism which assigns r
formulas of our formal theory \(P\). There are different ways of
{'Header 2': '2. Gödel’s Mathematical Work', 'Header 3': '2.2 The Incompleteness T
=====================
doing this. The most common is based on the unique representation of
natural numbers as products of powers of primes. Each symbol
\(s\) of number theory is assigned a positive natural number
\(\num(s)\) in a fixed but arbitrary way, e.g.
Gödel
numbering
The natural number corresponding to a sequence \(w = \lt w_0 ,\ldots ,w_k \gt\)
of symbols is
where \(p_k\) is the \(k+1\)st prime. It
is called its Gödel number and denoted by
\(\ulcorner w\urcorner\). In this way we can
{'Header 2': '2. Gödel’s Mathematical Work', 'Header 3': '2.2 The Incompleteness T
=====================
assign Gödel numbers to formulas, sequences of formulas (once a
method for distinguishing when one formula ends and another begins has
been adopted), and most notably, proofs.
An essential point here is that when a formula is construed as a
natural number, then the numeral corresponding to that natural number
```

```
can occur as the argument of a formula, thus enabling the syntax to
“refer” to itself, so to speak (i.e., when a numeral is
{'Header 2': '2. Gödel’s Mathematical Work', 'Header 3': '2.2 The Incompleteness T
=====================
substituted into a formula the Gödel number of which the numeral
represents). This will eventually allow Gödel to formalize the
Liar paradox (with “provability” in place of
“truth”) by substituting into the formula which says,
‘the formula, whose code is \(x\), is unprovable,’
its own natural number code (or more precisely the corresponding
numeral).
Another concept required to carry out the formalization is the concept
of numeralwise expressibility of number theoretic predicates. A
{'Header 2': '2. Gödel’s Mathematical Work', 'Header 3': '2.2 The Incompleteness T
=====================
number-theoretic formula \(\phi(n_1 , \ldots ,n_k)\) is in
\(P\) if for each tuple of natural numbers
\((n_1 , \ldots ,n_k)\):
numeralwise expressible
where \(\underline{n}\) is the formal term which denotes the natural
number \(n\). (In \(P\), this is
\(S(S(\ldots S\)(0)…), where \(n\)
is the number of iterations of the successor function applied to the
constant symbol 0.) One of the principal goals is to numeralwise
express the predicate
\(\Prf(x, y)\): ‘the sequence with Gödel
{'Header 2': '2. Gödel’s Mathematical Work', 'Header 3': '2.2 The Incompleteness T
=====================
```

- 셀 출력

```
Incompleteness Theorem: \(P\) is if
\(P \vdash \neg \phi(\underline{n})\) for all \(n\)
implies \(P \not\vdash \exists x\phi(x)\).
Naturally this implies consistency and follows from the assumption
that the natural numbers satisfy the axioms of Peano arithmetic.
\(\omega\)-consistent
One of the main technical tools used in the proof is , a mechanism which assign
formulas of our formal theory \(P\). There are different ways of
{'Header 2': '2. Gödel’s Mathematical Work', 'Header 3': '2.2 The Incompletenes
=====================
doing this. The most common is based on the unique representation of
natural numbers as products of powers of primes. Each symbol
\(s\) of number theory is assigned a positive natural number
\(\num(s)\) in a fixed but arbitrary way, e.g.
Gödel
numbering
The natural number corresponding to a sequence \(w = \lt w_0 ,\ldots ,w_k \gt\)
of symbols is
where \(p_k\) is the \(k+1\)st prime. It
is called its Gödel number and denoted by
\(\ulcorner w\urcorner\). In this way we can
{'Header 2': '2. Gödel’s Mathematical Work', 'Header 3': '2.2 The Incompletenes
```

```
=====================
assign Gödel numbers to formulas, sequences of formulas (once a
method for distinguishing when one formula ends and another begins has
been adopted), and most notably, proofs.
An essential point here is that when a formula is construed as a
natural number, then the numeral corresponding to that natural number
can occur as the argument of a formula, thus enabling the syntax to
“refer” to itself, so to speak (i.e., when a numeral is
{'Header 2': '2. Gödel’s Mathematical Work', 'Header 3': '2.2 The Incompletenes
=====================
substituted into a formula the Gödel number of which the numeral
represents). This will eventually allow Gödel to formalize the
Liar paradox (with “provability” in place of
“truth”) by substituting into the formula which says,
‘the formula, whose code is \(x\), is unprovable,’
its own natural number code (or more precisely the corresponding
numeral).
Another concept required to carry out the formalization is the concept
of numeralwise expressibility of number theoretic predicates. A
{'Header 2': '2. Gödel’s Mathematical Work', 'Header 3': '2.2 The Incompletenes
=====================
number-theoretic formula \(\phi(n_1 , \ldots ,n_k)\) is in
\(P\) if for each tuple of natural numbers
\((n_1 , \ldots ,n_k)\):
numeralwise expressible
where \(\underline{n}\) is the formal term which denotes the natural
number \(n\). (In \(P\), this is
\(S(S(\ldots S\)(0)…), where \(n\)
is the number of iterations of the successor function applied to the
constant symbol 0.) One of the principal goals is to numeralwise
express the predicate
\(\Prf(x, y)\): ‘the sequence with Gödel
{'Header 2': '2. Gödel’s Mathematical Work', 'Header 3': '2.2 The Incompletenes
=====================
```

3) `한계`

- **`HTMLHeaderTextSplitter` = 때로는 `특정 헤더`를 `누락`할 수 있음**
  - 예시: 아래의 코드는 헤더가 항상 관련 텍스트보다 `위` 에 있는 노드 (**`=이전 형제 노드`**), 조상 노드 및 이들의 조합에 위치한다고 가정
  - 다음 뉴스 기사 (아래 url): 최상위 헤드라인의 텍스트의 태그 = `h1`
    - `but` 우리의 예상과는 다르게 **텍스트 요소와 `별개의 하위 트리`에 있음**

- `h1` 관련 텍스트 ≠ 청크 메타데이터
- 관련되는 경우: `h2`, 관련 텍스트는 볼 수 있음

```
from langchain_text_splitters import RecursiveCharacterTextSplitter

# 분할할 HTML 페이지의 URL 지정하기
url = "https://www.cnn.com/2023/09/25/weather/el-nino-winter-us-climate/index.htm


# 분할할 헤더 태그와 해당 헤더의 이름 지정하기
headers_to_split_on = [
    ("h1", "Header 1"),
    ("h2", "Header 2"),
]


# 지정된 헤더를 기준으로 HTML 텍스트를 분할하는 HTMLHeaderTextSplitter 객체 생성하기
html_splitter = HTMLHeaderTextSplitter(headers_to_split_on=headers_to_split_on)


# 지정된 URL의 HTML 페이지를 분할하여 결과를 html_header_splits 변수에 저장하기
html_header_splits = html_splitter.split_text_from_url(url)


# 분할된 결과 출력하기
for header in html_header_splits:
    print(f"{header.page_content[:100]}")
    print(f"{header.metadata}", end="\n=====================\n")
```

- 셀 출력

```
CNN values your feedback
1. How relevant is this ad to you?
2. Did you encounter any technical i
{}
=====================
An El Niño winter is coming. Here’s what that could mean for the US
{'Header 1': 'An El Niño winter is coming. Here’s what that could mean for the U
=====================
By , CNN Meteorologist
Mary Gilbert
3 min read
Published
        4:44 AM EDT, Mon September
{'Header 1': 'An El Niño winter is coming. Here’s what that could mean for the U
=====================
What could this winter look like?
{'Header 1': 'An El Niño winter is coming. Here’s what that could mean for the U
=====================
No two El Niño winters are the same, but many have temperature and precipitatio
```

```
{}
=====================
```

- *next:* ***`재귀적 JSON 분할 (RecursiveJsonSplitter)`***
