TextSplitter

class langchain_text_splitters.base.TextSplitter(chunk_size: int = 4000, chunk_overlap: int = 200, length_function: ~typing.Callable[[str], int] = <built-in function len>, keep_separator: bool | ~typing.Literal['start', 'end'] = False, add_start_index: bool = False, strip_whitespace: bool = True) # [source]

Interface for splitting text into chunks.

Create a new TextSplitter.

Parameters:

- **chunk_size** (int) – Maximum size of chunks to return
- **chunk_overlap** (int) – Overlap in characters between chunks
- **length_function** (Callable[[str], int]) – Function that measures the length of given chunks
- **keep_separator** (Union[bool, Literal['start', 'end']]) – Whether to keep the separator and where to place it in each corresponding chunk (True='start')
- **add_start_index** (bool) – If True, includes chunk's start index in metadata
- **strip_whitespace** (bool) – If True, strips whitespace from the start and end of every document

Methods

`__init__`([chunk_size, chunk_overlap, ...])	Create a new TextSplitter.
`atransform_documents`(documents, **kwargs)	Asynchronously transform a list of documents.
`create_documents`(texts[, metadatas])	Create documents from a list of texts.
`from_huggingface_tokenizer`(tokenizer, **kwargs)	Text splitter that uses HuggingFace tokenizer to count length.
`from_tiktoken_encoder`([encoding_name, ...])	Text splitter that uses tiktoken encoder to count length.
`split_documents`(documents)	Split documents.
`split_text`(text)	Split text into multiple components.

`transform_documents`(documents, **kwargs)	Transform sequence of documents by splitting them.

`__init__(chunk_size: int = 4000, chunk_overlap: int = 200, length_function: ~typing.Callable[[str], int] = <built-in function len>, keep_separator: bool | ~typing.Literal['start', 'end'] = False, add_start_index: bool = False, strip_whitespace: bool = True) → None` # [source]

Create a new TextSplitter.

Parameters:

- **chunk_size** (int) – Maximum size of chunks to return
- **chunk_overlap** (int) – Overlap in characters between chunks
- **length_function** (Callable[[str], int]) – Function that measures the length of given chunks
- **keep_separator** (bool | Literal['start', 'end']) – Whether to keep the separator and where to place it in each corresponding chunk (True='start')
- **add_start_index** (bool) – If True, includes chunk's start index in metadata
- **strip_whitespace** (bool) – If True, strips whitespace from the start and end of every document

Return type:

None

`async atransform_documents(`

`documents: Sequence[Document],`

`**kwargs: Any,`

`) → Sequence[Document]` #

Asynchronously transform a list of documents.

Parameters:

- **documents** (Sequence[Document]) – A sequence of Documents to be transformed.
- **kwargs** (Any)

Returns:

A sequence of transformed Documents.

Return type:

Sequence[Document]

```
create_documents(
  texts: list[str],
  metadatas: list[dict[Any, Any]] | None = None,
) → list[Document] #                                    [source]
```

Create documents from a list of texts.

Parameters:

- **texts** (list[str])
- **metadatas** (list[dict[Any, Any]] | None)

Return type:

list[Document]

```
classmethod from_huggingface_tokenizer(
  tokenizer: Any,
  **kwargs: Any,
) → TextSplitter #                                      [source]
```

Text splitter that uses HuggingFace tokenizer to count length.

Parameters:

- **tokenizer** (Any)
- **kwargs** (Any)

Return type:

TextSplitter

```
classmethod from_tiktoken_encoder(
  encoding_name: str = 'gpt2',
  model_name: str | None = None,
```

```
    allowed_special: Literal['all'] | Set[str] = {},
    disallowed_special: Literal['all'] | Collection[str] = 'all',
    **kwargs: Any,
) → TS #
```

[source]

Text splitter that uses tiktoken encoder to count length.

Parameters:

- **encoding_name** (str)
- **model_name** (str | None)
- **allowed_special** (Literal['all'] | ~collections.abc.Set[str])
- **disallowed_special** (Literal['all'] | ~collections.abc.Collection[str])
- **kwargs** (Any)

Return type:

TS

```
split_documents(
    documents: Iterable[Document],
) → list[Document] #
```

[source]

Split documents.

Parameters:

documents (Iterable[Document])

Return type:

list[Document]

```
abstractmethod split_text(text: str) → list[str] #
```

[source]

Split text into multiple components.

Parameters:

text (str)

Return type:

list[str]

```
transform_documents(
  documents: Sequence[Document],
  **kwargs: Any,
) → Sequence[Document] #                                    [source]
```

Transform sequence of documents by splitting them.

Parameters:

- **documents** (Sequence[Document])
- **kwargs** (Any)

Return type:

Sequence[Document]
