TokenTextSplitter

```
class langchain_text_splitters.base.TokenTextSplitter(
  encoding_name: str = 'gpt2',
  model_name: str | None = None,
  allowed_special: Literal['all'] | Set[str] = {},
  disallowed_special: Literal['all'] | Collection[str] = 'all',
  **kwargs: Any,
) #
```

[source]

Splitting text to tokens using model tokenizer.

Create a new TextSplitter.

Methods

`__init__`([encoding_name, model_name, ...])	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)	Splits the input text into smaller chunks based on tokenization.
`transform_documents`(documents, **kwargs)	Transform sequence of documents by splitting them.

Parameters:

- **encoding_name** (str)
- **model_name** (Optional[str])

- **allowed_special** (Union[Literal['all'], Set[str]])
- **disallowed_special** (Union[Literal['all'], Collection[str]])
- **kwargs** (Any)

```
__init__(
  encoding_name: str = 'gpt2',
  model_name: str | None = None,
  allowed_special: Literal['all'] | Set[str] = {},
  disallowed_special: Literal['all'] | Collection[str] = 'all',
  **kwargs: Any,
) → None #
```

[source]

Create a new TextSplitter.

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:

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] #
```

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 #
```

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 #
```

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] #
```

Split documents.

Parameters:

documents (Iterable[Document])

Return type:

list[Document]

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

[source]

Splits the input text into smaller chunks based on tokenization.

This method uses a custom tokenizer configuration to encode the input text into tokens, processes the tokens in chunks of a specified size with overlap, and decodes them back into text chunks. The splitting is performed using the split_text_on_tokens function.

Parameters:

text (str) – The input text to be split into smaller chunks.

Returns:

A list of text chunks, where each chunk is derived from a portion of the input text based on the tokenization and chunking rules.

Return type:

List[str]

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

Transform sequence of documents by splitting them.

Parameters:

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

Return type:

Sequence[Document]

Examples using TokenTextSplitter

- Apache Doris
- AzureAISearchRetriever
- How to handle long text when doing extraction