Split by tokens

Language models have a token limit. You should not exceed the token limit. When you split your text into chunks it is therefore a good idea to count the number of tokens. There are many tokenizers. When you count tokens in your text you should use the same tokenizer as used in the language model.

tiktoken

> tiktoken is a fast `BPE` tokenizer created by `OpenAI`.

We can use it to estimate tokens used. It will probably be more accurate for the OpenAI models.

1. How the text is split: by character passed in.
2. How the chunk size is measured: by `tiktoken` tokenizer.

```
%pip install --upgrade --quiet  tiktoken
```

```
# This is a long document we can split up.
with open("../../state_of_the_union.txt") as
f:
    state_of_the_union = f.read()
from langchain.text_splitter import
CharacterTextSplitter
```

```
text_splitter =
CharacterTextSplitter.from_tiktoken_encoder(
    chunk_size=100, chunk_overlap=0
)
texts =
text_splitter.split_text(state_of_the_union)
```

```
print(texts[0])
```

```
Madam Speaker, Madam Vice President, our
First Lady and Second Gentleman. Members of
Congress and the Cabinet. Justices of the
Supreme Court. My fellow Americans.

Last year COVID-19 kept us apart. This year
we are finally together again.

Tonight, we meet as Democrats Republicans and
Independents. But most importantly as
```

```
Americans.

With a duty to one another to the American
people to the Constitution.
```

Note that if we use `CharacterTextSplitter.from_tiktoken_encoder`, text is only split by `CharacterTextSplitter` and `tiktoken` tokenizer is used to merge splits. It means that split can be larger than chunk size measured by `tiktoken` tokenizer. We can use `RecursiveCharacterTextSplitter.from_tiktoken_encoder` to make sure splits are not larger than chunk size of tokens allowed by the language model, where each split will be recursively split if it has a larger size.

We can also load a tiktoken splitter directly, which ensure each split is smaller than chunk size.

```
from langchain.text_splitter import
TokenTextSplitter

text_splitter =
TokenTextSplitter(chunk_size=10,
chunk_overlap=0)

texts =
text_splitter.split_text(state_of_the_union)
print(texts[0])
```

spaCy

> spaCy is an open-source software library for advanced natural language processing, written in the programming languages Python and Cython.

Another alternative to `NLTK` is to use spaCy tokenizer.

1. How the text is split: by `spaCy` tokenizer.
2. How the chunk size is measured: by number of characters.

```
%pip install --upgrade --quiet  spacy
```

```
# This is a long document we can split up.
with open("../../state_of_the_union.txt") as f:
    state_of_the_union = f.read()
```

```
from langchain.text_splitter import SpacyTextSplitter
```

```
text_splitter =
SpacyTextSplitter(chunk_size=1000)
```

```
texts =
text_splitter.split_text(state_of_the_union)
print(texts[0])
```

```
Madam Speaker, Madam Vice President, our
First Lady and Second Gentleman.

Members of Congress and the Cabinet.

Justices of the Supreme Court.

My fellow Americans.



Last year COVID-19 kept us apart.

This year we are finally together again.



Tonight, we meet as Democrats Republicans and
Independents.

But most importantly as Americans.
```

```


With a duty to one another to the American
people to the Constitution.



And with an unwavering resolve that freedom
will always triumph over tyranny.



Six days ago, Russia’s Vladimir Putin sought
to shake the foundations of the free world
thinking he could make it bend to his
menacing ways.

But he badly miscalculated.



He thought he could roll into Ukraine and the
world would roll over.

Instead he met a wall of strength he never
imagined.



He met the Ukrainian people.
```

```
From President Zelenskyy to every Ukrainian,
their fearlessness, their courage, their
determination, inspires the world.
```

SentenceTransformers

The `SentenceTransformersTokenTextSplitter` is a specialized text splitter for use with the sentence-transformer models. The default behaviour is to split the text into chunks that fit the token window of the sentence transformer model that you would like to use.

```
from langchain.text_splitter import
SentenceTransformersTokenTextSplitter
```

```
splitter =
SentenceTransformersTokenTextSplitter(chunk_ove
text = "Lorem "
```

```
count_start_and_stop_tokens = 2
text_token_count =
splitter.count_tokens(text=text) -
count_start_and_stop_tokens
print(text_token_count)
```

```
2
```

```
token_multiplier =
splitter.maximum_tokens_per_chunk //
text_token_count + 1

# `text_to_split` does not fit in a single
chunk
text_to_split = text * token_multiplier

print(f"tokens in text to split:
{splitter.count_tokens(text=text_to_split)}")
```

```
tokens in text to split: 514
```

```
text_chunks =
splitter.split_text(text=text_to_split)

print(text_chunks[1])
```

```
lorem
```

NLTK

> The Natural Language Toolkit, or more commonly NLTK, is a suite of libraries and programs for symbolic and statistical natural language processing (NLP) for English written in the Python programming language.

Rather than just splitting on “”, we can use `NLTK` to split based on NLTK tokenizers.

1. How the text is split: by `NLTK` tokenizer.
2. How the chunk size is measured: by number of characters.

```
# pip install nltk
```

```
# This is a long document we can split up.
with open("../../state_of_the_union.txt") as f:
    state_of_the_union = f.read()
```

```
from langchain.text_splitter import NLTKTextSplitter
```

```
text_splitter =
NLTKTextSplitter(chunk_size=1000)
```

```
texts =
text_splitter.split_text(state_of_the_union)
print(texts[0])
```

```
Madam Speaker, Madam Vice President, our
First Lady and Second Gentleman.

Members of Congress and the Cabinet.

Justices of the Supreme Court.

My fellow Americans.

Last year COVID-19 kept us apart.

This year we are finally together again.

Tonight, we meet as Democrats Republicans and
Independents.

But most importantly as Americans.

With a duty to one another to the American
people to the Constitution.
```

```
And with an unwavering resolve that freedom
will always triumph over tyranny.

Six days ago, Russia’s Vladimir Putin sought
to shake the foundations of the free world
thinking he could make it bend to his
menacing ways.

But he badly miscalculated.

He thought he could roll into Ukraine and the
world would roll over.

Instead he met a wall of strength he never
imagined.

He met the Ukrainian people.

From President Zelenskyy to every Ukrainian,
their fearlessness, their courage, their
determination, inspires the world.

Groups of citizens blocking tanks with their
bodies.
```

KoNLPY

> KoNLPy: Korean NLP in Python is is a Python package for natural language processing (NLP) of the Korean language.

Token splitting involves the segmentation of text into smaller, more manageable units called tokens. These tokens are often words, phrases, symbols, or other meaningful elements crucial for further processing and analysis. In languages like English, token splitting typically involves separating words by spaces and punctuation marks. The effectiveness of token splitting largely depends on the tokenizer's understanding of the language structure, ensuring the generation of meaningful tokens. Since tokenizers designed for the English language are not equipped to understand the unique semantic structures of other languages, such as Korean, they cannot be effectively used for Korean language processing.

Token splitting for Korean with KoNLPy's Kkma Analyzer

In case of Korean text, KoNLPY includes at morphological analyzer called `Kkma` (Korean Knowledge Morpheme Analyzer). `Kkma` provides detailed morphological analysis of Korean text. It breaks down sentences into words and words into their respective morphemes, identifying parts of speech for each token. It can segment a block of text into individual sentences, which is particularly useful for processing long texts.

Usage Considerations

While `Kkma` is renowned for its detailed analysis, it is important to note that this precision may impact processing speed. Thus, `Kkma` is best suited for applications where analytical depth is prioritized over rapid text processing.

```
# pip install konlpy
```

```
# This is a long Korean document that we want
to split up into its component sentences.
with open("./your_korean_doc.txt") as f:
    korean_document = f.read()
```

```
from langchain.text_splitter import
KonlpyTextSplitter

text_splitter = KonlpyTextSplitter()
```

```
texts =
text_splitter.split_text(korean_document)
# The sentences are split with "\n\n"
```

```
characters.
print(texts[0])
```

```
□□□ □□□ □□□ □ □□□□□ □□□□ □□□ □□□.

□□ □□□ □□□ □□□ □□□□, □□ □□□ □□□ □□□ □□□□.

□□, □ □□□□ □□□□□ □□ □□□ □□ □□□.

□ □□ □□□□□ □□ □□ □□ □□□ □□□ □□ □□□ □□□.

□□ □□, □□□ □□□□ □□ □□□□ □ □□ □ □ □ □□ □□□ □□
□.

□ □□□ □□ □□□□ □□□, □□ □□□□□ □□□ □□□ □□□□.

□□□ □□ □□□ □□□□ □□□.

□□□ □□□□ □□ □□□ □□□ □□ □□ □□□ □□ □□ □□.

□□□ □□ □□□□, □ □□□ □□□ □□□□ □□□ □□ □□□□□ □□.

□□□ □□ □□□ □□□ □□□ □ □□ □□□□□ □□□ □ □ □□□□ □□
□ □□□□.

□ □ □ □□□ □□ □□□ □□□ □□□ □□, □□□ □□□ □□□ □□□
□.

□□ □□□ □□□ □ □□ □□□ □□□ □□□ □□□ □□□.
```

```
□□□□□ □ □□□ □□ □□□ □□ □, □ □□ □□ □□ □□□ □□□.

□ □□□ □□ □□ □□ □□ □□□ □□, □□□ □□□ □ □□□ □□ □□
□□□□□□ □□□□.

- □□□ (The Tale of Chunhyang)
```

Hugging Face tokenizer

> Hugging Face has many tokenizers.

We use Hugging Face tokenizer, the GPT2TokenizerFast to count the text length in tokens.

1. How the text is split: by character passed in.
2. How the chunk size is measured: by number of tokens calculated by the `Hugging Face` tokenizer.

```python
from transformers import GPT2TokenizerFast

tokenizer =
GPT2TokenizerFast.from_pretrained("gpt2")
```

```python
# This is a long document we can split up.
with open("../../../state_of_the_union.txt")
as f:
    state_of_the_union = f.read()
from langchain.text_splitter import
CharacterTextSplitter
```

```
text_splitter =
CharacterTextSplitter.from_huggingface_tokenize
    tokenizer, chunk_size=100, chunk_overlap=0
)
texts =
text_splitter.split_text(state_of_the_union)
```

```
print(texts[0])
```

```
Madam Speaker, Madam Vice President, our
First Lady and Second Gentleman. Members of
Congress and the Cabinet. Justices of the
Supreme Court. My fellow Americans.

Last year COVID-19 kept us apart. This year
we are finally together again.

Tonight, we meet as Democrats Republicans and
Independents. But most importantly as
Americans.
```

```
With a duty to one another to the American
people to the Constitution.
```