Tokenização e Segmentação de Sentenças

A tokenização e a segmentação da palavras podem ser realizadas com a lib `Stanza`, as quais são executadas em conjunto pelo `TokenizeProcessor`. Este processador divide o texto de entrada em tokens e frases. Este processador pode ser chamado pela função `tokenize`.

Nome	Nome da Classe	pré-requisito	Resumo	Descrição
tokenize	TokenizeProcessor	-	Segmenta um documento em sentenças gerando uma lista de `Tokens`. Este processo também detecta palavras compostas que representam um token com a utilização de MWTProcessor.	Tokeniza o texto e realiza a segmentação de sentenças.

Opções

A seguir, algumas configurações para configuração do `TokenizeProcessor`:

Nome	Type	Default	Descrição
tokenize_batch_size	int	32	Este argumento especifica o número de parágrafos a serem anlalisados em uma batelada de processamento.
tokenize_pretokenized	bool	False	Se o processamento for realizado com um texto já tokenizado, colocar True.
tokenize_no_ssplit	bool	False	Não processa a segmentação, somente tokenização.

Exemplo de Usabilidade

A utilização do `TokenizeProcessor` geralmente é a primeira etapa do processo. Após essa aplicação, o documento se torna uma lista de tokens e setenças. Em cada lista de sentença é possivel acessar os tokens individualmente.Tokenization and Sentence Segmentation

A seguir um exemplo deste processamento:

```
import stanza
nlp = stanza.Pipeline(lang='en', processors='tokenize')
doc = nlp('This is a test sentence for stanza. This is another sentence.')
for i, sentence in enumerate(doc.sentences):
    print(f'====== Sentence {i+1} tokens =======')
    print(*[f'id: {token.id}\ttext: {token.text}' for token in sentence.tokens], sep='\n')
```

Este código irá gerar a seguinte saída:

```
====== Sentence 1 tokens =======
id: 1   text: This
id: 2   text: is
id: 3   text: a
id: 4   text: test
id: 5   text: sentence
id: 6   text: for
id: 7   text: stanza
id: 8   text: .
====== Sentence 2 tokens =======
id: 1   text: This
id: 2   text: is
id: 3   text: another
id: 4   text: sentence
id: 5   text: .
```

Para acessar somente a segmentação de sentenças é possível fazendo:

```
print([sentence.text for sentence in doc.sentences])
```

Tokenização sem Segmentação de Sentença

```
import stanza
nlp = stanza.Pipeline(lang='en', processors='tokenize', tokenize_no_ssplit=True)
doc = nlp('This is a sentence.\n\nThis is a second. This is a third.')
for i, sentence in enumerate(doc.sentences):
    print(f'====== Sentence {i+1} tokens =======')
    print(*[f'id: {token.id}\ttext: {token.text}' for token in sentence.tokens], sep='\n')
```

A saída do código quando exclui-se a segmentação de sentença proporciona somente 2 sentenças definidas, tal como pode ser observado abaixo.

```
====== Sentence 1 tokens =======
id: 1   text: This
id: 2   text: is
id: 3   text: a
id: 4   text: sentence
id: 5   text: .
====== Sentence 2 tokens =======
id: 1   text: This
id: 2   text: is
id: 3   text: a
id: 4   text: second
id: 5   text: .
id: 6   text: This
id: 7   text: is
id: 8   text: a
id: 9   text: third
id: 10  text: .
```

Quando a segmentação de sentença é setada (default), a saída é de acordo com:

```
====== Sentence 1 tokens =======
id: 1   text: This
id: 2   text: is
id: 3   text: a
id: 4   text: sentence
id: 5   text: .
====== Sentence 2 tokens =======
id: 1   text: This
id: 2   text: is
id: 3   text: a
id: 4   text: second
id: 5   text: .
====== Sentence 3 tokens =======
id: 1   text: This
id: 2   text: is
id: 3   text: a
id: 4   text: third
id: 5   text: .
```

Processar texto pré-tokenizado

Em alguns casos o texto pode está já tokenizado. Neste caso, é possível utilziar somente a segmentação de sentenças, bata setar `tokenize_pretokenized` como `True`:

```
import stanza
nlp = stanza.Pipeline(lang='en', processors='tokenize', tokenize_pretokenized=True)
doc = nlp('This is token.ization done my way!\nSentence split, too!')
for i, sentence in enumerate(doc.sentences):
    print(f'====== Sentence {i+1} tokens =======')
    print(*[f'id: {token.id}\ttext: {token.text}' for token in sentence.tokens], sep='\n')
```

Um alternativa é utilziar uma lista de stings:

```
import stanza
nlp = stanza.Pipeline(lang='en', processors='tokenize', tokenize_pretokenized=True)
doc = nlp([['This', 'is', 'token.ization', 'done', 'my', 'way!'], ['Sentence', 'split,', 'too!']])
for i, sentence in enumerate(doc.sentences):
    print(f'====== Sentence {i+1} tokens =======')
    print(*[f'id: {token.id}\ttext: {token.text}' for token in sentence.tokens], sep='\n')
```

Segue a saída com `tokenize_pretokenized=True`

```
====== Sentence 1 tokens =======
id: 1   text: This
id: 2   text: is
id: 3   text: token.ization
id: 4   text: done
id: 5   text: my
id: 6   text: way!
====== Sentence 2 tokens =======
id: 1   text: Sentence
id: 2   text: split,
id: 3   text: too!
```

Segue a saída com `tokenize_pretokenized=False`

```
====== Sentence 1 tokens =======
id: 1   text: This
id: 2   text: is
id: 3   text: token
id: 4   text: .
id: 5   text: ization
id: 6   text: done
id: 7   text: my
id: 8   text: way
id: 9   text: !
====== Sentence 2 tokens =======
id: 1   text: Sentence
id: 2   text: split
id: 3   text: ,
id: 4   text: too
id: 5   text: !
```