Optilearn.text.word_tokenizer

08 November 2024 19:24

word_tokenizer

Tokenizes input text based on specified parameters, providing flexible customization options to suit different text-processing requirements. This function supports handling punctuation, URLs, email addresses, case sensitivity, special character tokenization, and more, making it versatile for tasks in natural language processing (NLP).

Parameters

- **text** *(str)*:
  The input text to be tokenized. This string is split into tokens according to the chosen options for punctuation, case, and special character handling.
- **output_case** *(str, optional)*:
  Specifies the case format for the output tokens. Available options:
  - 'same' *(default)*: Tokens are returned in the original case found in text.
  - 'lower': Converts all tokens to lowercase.
  - 'upper': Converts all tokens to uppercase.
- **del_words** *((list, tuple, set), optional)*:
  A collection of words to exclude from the output. This can be useful for filtering out stop words or irrelevant terms during tokenization. If None, no words are excluded. Default is None.
- **punc_tokenize** *(bool, optional)*:
  Determines how punctuation marks are handled in the tokenization process.
  - True *(default)*: Each punctuation mark is treated as a separate token.
  - False: Punctuation is ignored in the output, so it won't appear as a token.
- **include_slash** *(bool, optional)*:
  Controls whether slashes (both / and \) are included as separate tokens.
  - True *(default)*: Slashes are included as tokens.
  - False: Slashes are excluded from the output.
- **smart_recognition** *(bool, optional)*:
  Enables more sophisticated tokenization that recognizes certain patterns, such as:
  - Contractions (e.g., "don't" → "do", "n't")
  - Possessives (e.g., "Sarah's" → "Sarah", "'s")
  - Complex entities like URLs and email addresses When True *(default)*, this setting improves tokenization accuracy, particularly in text with contractions or special formats.
- **sp_character_tokens** *(bool, optional)*:
  Specifies whether to treat special characters (like emojis and symbols) as individual tokens.
  - True: Each special character is treated as a token.
  - False *(default)*: Special characters are ignored.

Returns

- **list**:
  A list of tokens derived from the input text, structured according to the parameters specified. The tokens can include or exclude punctuation, adhere to case preferences, and filter out specific words as needed.

Examples

Below are some common use cases and examples to illustrate how the word_tokenizer function can be used.

Basic Usage

To tokenize a sentence, with specific words removed:

```
text = "This is a sample sentence."
del_words = ['is', 'a']
tokens = word_tokenizer(text, output_case='lower', del_words=del_words)
print(tokens)  # Output: ['this', 'sample', 'sentence']
```

Handling Punctuation

Tokenizing text while ignoring punctuation:

```
text = "Complex-tokenization example, with punctuation!"
tokens = word_tokenizer(text, punc_tokenize=False)
print(tokens)  # Output: ['Complex-tokenization', 'example', 'with', 'punctuation']
```

Recognizing URLs

Ensuring URLs are preserved correctly as tokens:

```
text = "Visit our website at https://www.example.com."
tokens = word_tokenizer(text)
print(tokens)  # Output: ['Visit', 'our', 'website', 'at', 'https', '://www.example.com',
```

Recognizing Email Addresses

Ensuring email addresses are tokenized appropriately:

```
text = "Please contact us at example@gmail.com."
tokens = word_tokenizer(text)
print(tokens)  # Output: ['Please', 'contact', 'us', 'at', 'example', '@', 'gmail.com',
```

Including Special Characters
Treating special characters and emojis as separate tokens:

```
text = "Hello world! 😊"
tokens = word_tokenizer(text, sp_character_tokens=True)
print(tokens)  # Output: ['Hello', 'world', '!', '😊']
```

Excluding Slashes
Removing slashes from the tokenized output:

```
text = "Use backslash \\ or forward slash /."
tokens = word_tokenizer(text, include_slash=False)
print(tokens)  # Output: ['Use', 'backslash', 'or', 'forward', 'slash', '.']
```

Notes
- **Tokenization Flexibility**: This function provides multiple parameters to control how text is split, making it adaptable to various NLP preprocessing needs.
- **Pattern Recognition**: With smart_recognition, this function can preserve complex patterns like URLs and emails, offering more accurate tokenization in these contexts.
- **Special Character Support**: By setting sp_character_tokens=True, this function can handle emojis, symbols, and other unique characters that may appear in user-generated content.
- **Word Filtering**: del_words allows for custom stop-word removal or exclusion of specific words, useful for text analysis or preparing data for machine learning models.

This function is well-suited for projects in text processing, social media analysis, or any application needing refined tokenization control.