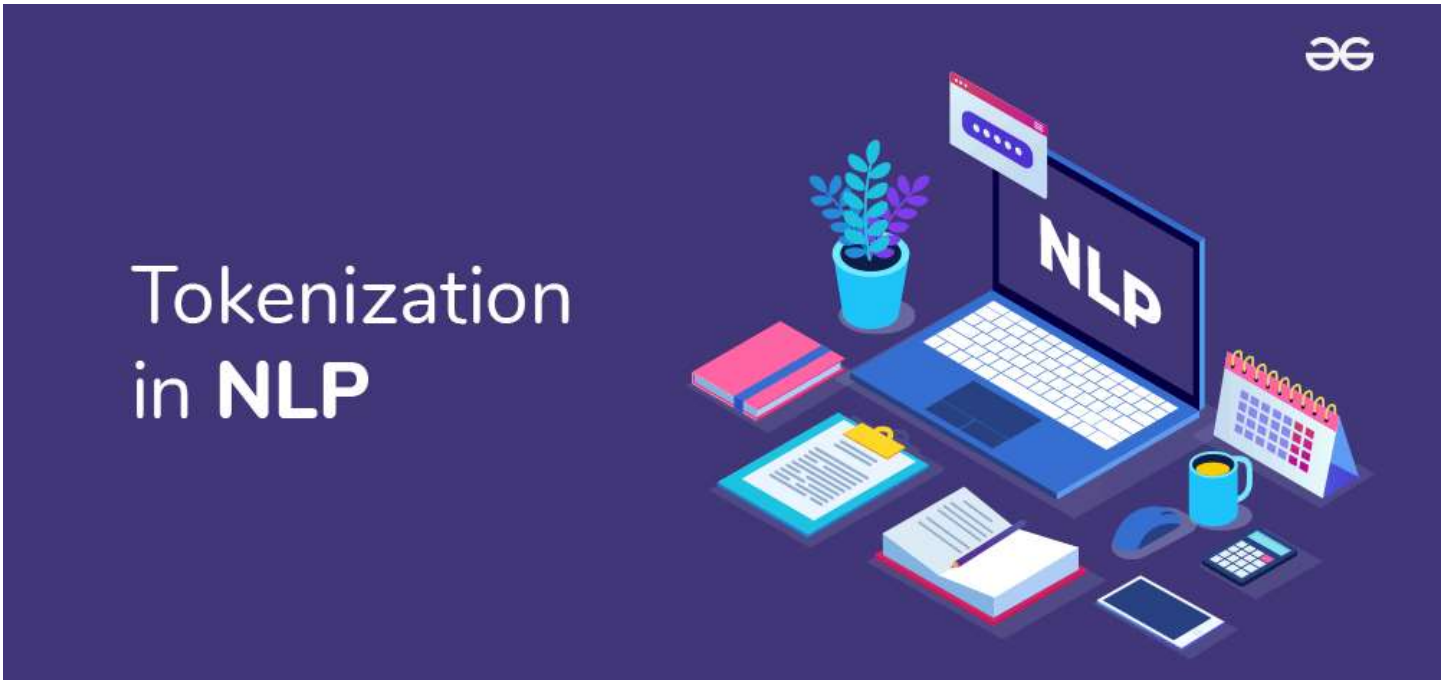
NLP | How tokenizing text, sentence, words works

Tokenization in **natural language processing (NLP)** is a technique that involves dividing a sentence or phrase into smaller units known as tokens. These tokens can encompass words, dates, punctuation marks, or even fragments of words. The article aims to cover the fundamentals of tokenization, it's types and use case.



What is Tokenization in NLP?

Natural Language Processing (NLP) is a subfield of computer science, artificial intelligence, information engineering, and human-computer interaction. This field focuses on how to program computers to process and analyze large amounts of natural language data. It is difficult to perform as the process of reading and understanding languages is far more complex than it seems at first glance. Tokenization is a foundation step in NLP pipeline that shapes the entire workflow.

Tokenization is the process of dividing a text into smaller units known as tokens. **Tokens** are typically words or sub-words in the context of natural language processing. Tokenization is a critical step in many NLP tasks, including text processing, language modelling, and machine translation. The process involves splitting a string, or text into a list of tokens. One can think of tokens as parts like a word is a token in a sentence, and a sentence is a token in a paragraph.

Tokenization involves using a tokenizer to segment unstructured data and natural language text into distinct chunks of information, treating them as different elements. The tokens within a document can

be used as vector, transforming an unstructured text document into a numerical data structure suitable for machine learning. This rapid conversion enables the immediate utilization of these tokenized elements by a computer to initiate practical actions and responses. Alternatively, they may serve as features within a machine learning pipeline, prompting more sophisticated decision-making processes or behaviors.

Types of Tokenization

Tokenization can be classified into several types based on how the text is segmented. Here are some types of tokenization:

Word Tokenization:

Word tokenization divides the text into individual words. Many NLP tasks use this approach, in which words are treated as the basic units of meaning.

Example:

```
Input: "Tokenization is an important NLP task."
Output: ["Tokenization", "is", "an", "important", "NLP", "task", "."]
```

Sentence Tokenization:

The text is segmented into sentences during sentence tokenization. This is useful for tasks requiring individual sentence analysis or processing.

Example:

```
Input: "Tokenization is an important NLP task. It helps break down text into smaller units."
Output: ["Tokenization is an important NLP task.", "It helps break down text into smaller unit
```

Subword Tokenization:

Subword tokenization entails breaking down words into smaller units, which can be especially useful when dealing with morphologically rich languages or rare words.

Example:

```
Input: "tokenization"
```

```
Output: ["token", "ization"]
```

Character Tokenization:

This process divides the text into individual characters. This can be useful for modelling character-level language.

Example:

```
Input: "Tokenization"
Output: ["T", "o", "k", "e", "n", "i", "z", "a", "t", "i", "o", "n"]
```

Need of Tokenization

Tokenization is a crucial step in text processing and natural language processing (NLP) for several reasons.

- **Effective Text Processing:** Tokenization reduces the size of raw text so that it can be handled more easily for processing and analysis.
- **Feature extraction:** Text data can be represented numerically for algorithmic comprehension by using tokens as features in machine learning models.
- **Language Modelling:** Tokenization in NLP facilitates the creation of organized representations of language, which is useful for tasks like text generation and language modelling.
- **Information Retrieval:** Tokenization is essential for indexing and searching in systems that store and retrieve information efficiently based on words or phrases.
- **Text Analysis:** Tokenization is used in many NLP tasks, including sentiment analysis and named entity recognition, to determine the function and context of individual words in a sentence.
- **Vocabulary Management:** By generating a list of distinct tokens that stand in for words in the dataset, tokenization helps manage a corpus's vocabulary.
- **Task-Specific Adaptation:** Tokenization can be customized to meet the needs of particular NLP tasks, meaning that it will work best in applications such as summarization and machine translation.
- **Preprocessing Step:** This essential preprocessing step transforms unprocessed text into a format appropriate for additional statistical and computational analysis.

Implementation for Tokenization

Sentence Tokenization using sent_tokenize

The code snippet uses **sent_tokenize** function from NLTK library. The `sent_tokenize` function is used to segment a given text into a list of sentences.

Python3

```
from  nltk.tokenize  import  sent_tokenize

text  =  "Hello everyone. Welcome to GeeksforGeeks. You are studying NLP article."

sent_tokenize(text)
```

Output:

```
['Hello everyone.',
 'Welcome to GeeksforGeeks.',
 'You are studying NLP article']
```

> **How sent_tokenize works ?**
>
> The sent_tokenize function uses an instance of PunktSentenceTokenizer from the nltk.tokenize.punkt module, which is already been trained and thus very well knows to mark the end and beginning of sentence at what characters and punctuation.

Sentence Tokenization using PunktSentenceTokenizer

When we have huge chunks of data then it is efficient to use ' `PunktSentenceTokenizer'` from the NLTK library. The Punkt tokenizer is a data-driven sentence tokenizer that comes with NLTK. It is trained on large corpus of text to identify sentence boundaries.

Python3

```
import  nltk.data

tokenizer  =  nltk.data.load(``'tokenizers/punkt/PY3/english.pickle'``)

tokenizer.tokenize(text)
```

Output:

```
['Hello everyone.',
 'Welcome to GeeksforGeeks.',
 'You are studying NLP article']
```

Tokenize sentence of different language

One can also tokenize sentence from different languages using different pickle file other than English. In the following code snippet, we have used NLTK library to tokenize a Spanish text into sentences using pre-trained Punkt tokenizer for Spanish. The Punkt tokenizer is a data-driven tokenizer that uses machine learning techniques to identify sentence boundaries.

Python3

```
import  nltk.data

spanish_tokenizer  =  nltk.data.load(``'tokenizers/punkt/PY3/spanish.pickle'``)

text  =  'Hola amigo. Estoy bien.'

spanish_tokenizer.tokenize(text)
```

Output:

```
['Hola amigo.',
 'Estoy bien.']
```

Word Tokenization using work_tokenize

The code snipped uses the word_tokenize function from NLTK library to tokenize a given text into individual words. The word_tokenize function is helpful for breaking down a sentence or text into its constituent words, facilitating further analysis or processing at the word level in natural language processing tasks.

Python3

```
from  nltk.tokenize  import  word_tokenize

text  =  "Hello everyone. Welcome to GeeksforGeeks."

word_tokenize(text)
```

Output:

```
['Hello', 'everyone', '.', 'Welcome', 'to', 'GeeksforGeeks', '.']
```

> ### How *word_tokenize* works?

> word_tokenize() function is a wrapper function that calls tokenize() on an instance of the TreebankWordTokenizer class.

Word Tokenization Using TreebankWordTokenizer

The code snippet uses the `TreebankWordTokenizer` from the Natural Language Toolkit (NLTK) to tokenize a given text into individual words.

Python3

```
from nltk.tokenize import TreebankWordTokenizer

tokenizer = TreebankWordTokenizer()

tokenizer.tokenize(text)
```

Output:

```
['Hello', 'everyone.', 'Welcome', 'to', 'GeeksforGeeks', '.']
```

These tokenizers work by separating the words using punctuation and spaces. And as mentioned in the code outputs above, it doesn't discard the punctuation, allowing a user to decide what to do with the punctuations at the time of pre-processing.

Word Tokenization using WordPunctTokenizer

The `WordPunctTokenizer` is one of the NLTK tokenizers that splits words based on punctuation boundaries. Each punctuation mark is treated as a separate token.

Python3

```
from nltk.tokenize import WordPunctTokenizer

tokenizer = WordPunctTokenizer()

tokenizer.tokenize(``"Let's see how it's working."``)
```

Output:

```
['Let', "'", 's', 'see', 'how', 'it', "'", 's', 'working', '.']
```

Word Tokenization using Regular Expression

The code snippet uses the `RegexpTokenizer` from the Natural Language Toolkit (NLTK) to tokenize a given text based on a regular expression pattern.

Python3

```
from nltk.tokenize import RegexpTokenizer

tokenizer = RegexpTokenizer(r``'\w+'``)

text = "Let's see how it's working."

tokenizer.tokenize(text)
```

Output:

```
['Let', 's', 'see', 'how', 'it', 's', 'working']
```

Using regular expressions allows for more fine-grained control over tokenization, and you can customize the pattern based on your specific requirements.

More Techniques for Tokenization

We have discussed the ways to implement how can we perform tokenization using NLTK library. We can also implement tokenization using following methods and libraries:

- **Spacy:** Spacy is NLP library that provide robust tokenization capabilities.
- **BERT tokenizer:** BERT uses WordPiece tokenizer is a type of subword tokenizer for tokenizing input text. Using regular expressions allows for more fine-grained control over tokenization, and you can customize the pattern based on your specific requirements.
- **Byte-Pair Encoding:** Byte Pair Encoding (BPE) is a data compression algorithm that has also found applications in the field of natural language processing, specifically for tokenization. It is a subword tokenization technique that works by iteratively merging the most frequent pairs of consecutive bytes (or characters) in a given corpus.
- **Sentence Piece:** SentencePiece is another subword tokenization algorithm commonly used for natural language processing tasks. It is designed to be language-agnostic and works by iteratively merging frequent sequences of characters or subwords in a given corpus.

Limitations of Tokenization

- Tokenization is unable to capture the meaning of the sentence hence, results in **ambiguity**.
- In certain languages like Chinese, Japanese, Arabic, lack distinct spaces between words. Hence, there is an **absence of clear boundaries** that complicates the process of tokenization.
- Text may also include more than one word, for example email address, URLs and **special symbols**, hence it is difficult to decide how to tokenize such elements.

Tokenization – Frequently Asked Questions (FAQs)

Q. What is Tokenization in NLP?

> Tokenization is the process of converting a sequence of text into smaller parts known as tokens in the context of Natural Language Processing (NLP) and machine learning. These tokens can be as short as a character or as long as a sentence.

Q. What is Lemmatization in NLP?

> Lemmatization is a text pre-processing method that helps natural language processing (NLP) models find similarities by reducing a word to its most basic meaning. A lemmatization algorithm, for instance, would reduce the word better to its lemme, or good.

Q. Which are most common types of tokenization?

> Word tokenization, which divides text into words, sentence tokenization, which divides text into sentences, subword tokenization, which divides words into smaller units, and character tokenization, which divides text into individual characters, are common forms of tokenization.