**НЛП | Как работает маркировка текста, предложения, слов**

Последнее обновление: 31 января 2024 г.

**Токенизация** в**обработке естественного языка (NLP)** — это метод, который предполагает разделение предложения или фразы на более мелкие единицы, называемые токенами. Эти токены могут включать в себя слова, даты, знаки препинания или даже фрагменты слов. Цель статьи — рассмотреть основы токенизации, её типы и области применения.

**Что такое токенизация в НЛП?**

[Обработка естественного языка (NLP)](https://translated.turbopages.org/proxy_u/en-ru.ru.412cad94-680dcc27-421e0f8a-74722d776562/https/www.geeksforgeeks.org/natural-language-processing-overview/) — это раздел [информатики](https://translated.turbopages.org/proxy_u/en-ru.ru.412cad94-680dcc27-421e0f8a-74722d776562/https/www.geeksforgeeks.org/computer-science-projects/), [искусственного интеллекта](https://translated.turbopages.org/proxy_u/en-ru.ru.412cad94-680dcc27-421e0f8a-74722d776562/https/www.geeksforgeeks.org/artificial-intelligence-an-introduction/), информационной инженерии и взаимодействия человека с компьютером. Эта область фокусируется на том, как программировать компьютеры для обработки и анализа больших объёмов данных на естественном языке. Это сложно, поскольку процесс чтения и понимания языков гораздо сложнее, чем кажется на первый взгляд. [Токенизация](https://translated.turbopages.org/proxy_u/en-ru.ru.412cad94-680dcc27-421e0f8a-74722d776562/https/www.geeksforgeeks.org/tokenize-text-using-nltk-python/) — это базовый этап в конвейере NLP, который формирует весь рабочий процесс.

**Токенизация** — это процесс разделения текста на более мелкие единицы, известные как токены.**Токены** обычно представляют собой слова или части слов в контексте обработки естественного языка. Токенизация является важным этапом во многих задачах обработки естественного языка, включая [обработку текста](https://translated.turbopages.org/proxy_u/en-ru.ru.412cad94-680dcc27-421e0f8a-74722d776562/https/www.geeksforgeeks.org/text-preprocessing-in-python-set-1/), [языковое моделирование](https://translated.turbopages.org/proxy_u/en-ru.ru.412cad94-680dcc27-421e0f8a-74722d776562/https/www.geeksforgeeks.org/videos/what-is-language-modelling-in-nlp/) и [машинный перевод](https://translated.turbopages.org/proxy_u/en-ru.ru.412cad94-680dcc27-421e0f8a-74722d776562/https/www.geeksforgeeks.org/machine-translation-of-languages-in-artificial-intelligence/). Этот процесс включает в себя разделение строки или текста на список токенов. Токены можно рассматривать как части, например, слово является токеном в предложении, а предложение — токеном в абзаце.

Токенизация предполагает использование токенизатора для разделения неструктурированных данных и текста на естественном языке на отдельные фрагменты информации, которые рассматриваются как разные элементы. Токены в документе можно использовать в качестве вектора, преобразующего неструктурированный текстовый документ в числовую структуру данных, подходящую для машинного обучения. Такое быстрое преобразование позволяет немедленно использовать эти токенизированные элементы в компьютере для выполнения практических действий и реакций. Кроме того, они могут служить в качестве признаков в конвейере машинного обучения, стимулируя более сложные процессы принятия решений или поведения.

**Типы токенизации**

Токенизацию можно разделить на несколько типов в зависимости от того, как сегментируется текст. Вот несколько типов токенизации:

**Токенизация слов:**

Токенизация слов делит текст на отдельные слова. Этот подход используется во многих задачах НЛП, где слова рассматриваются как основные смысловые единицы.

**Пример:**

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

**Токенизация предложений:**

Во время токенизации предложений текст разбивается на предложения. Это полезно для задач, требующих анализа или обработки отдельных предложений.

**Пример:**

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 units."]

**Токенизация подслова:**

Токенизация по частям слова подразумевает разбиение слов на более мелкие единицы, что может быть особенно полезно при работе с морфологически богатыми языками или редкими словами.

Пример:

Input: "tokenization"  
Output: ["token", "ization"]

**Маркировка символов:**

Этот процесс разбивает текст на отдельные символы. Это может быть полезно для моделирования языка на уровне символов.

Пример:

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

**Необходимость токенизации**

Токенизация является важным этапом в[обработке текста](https://translated.turbopages.org/proxy_u/en-ru.ru.412cad94-680dcc27-421e0f8a-74722d776562/https/www.geeksforgeeks.org/text-preprocessing-in-python-set-1/)и обработке естественного языка (NLP) по нескольким причинам.

* **Эффективная обработка текста:** токенизация уменьшает размер исходного текста, чтобы его было легче обрабатывать и анализировать.
* **Извлечение признаков:**текстовые данные могут быть представлены в числовом виде для алгоритмического понимания с помощью токенов в качестве признаков в [моделях машинного обучения](https://translated.turbopages.org/proxy_u/en-ru.ru.412cad94-680dcc27-421e0f8a-74722d776562/https/www.geeksforgeeks.org/machine-learning/).
* **Моделирование языка:** токенизация в НЛП способствует созданию организованных представлений о языке, что полезно для таких задач, как генерация текста и моделирование языка.
* **Поиск информации:** токенизация необходима для индексирования и поиска в системах, которые эффективно хранят и извлекают информацию на основе слов или фраз.
* **Анализ текста:** Токенизация используется во многих задачах НЛП, включая [анализ тональности](https://translated.turbopages.org/proxy_u/en-ru.ru.412cad94-680dcc27-421e0f8a-74722d776562/https/www.geeksforgeeks.org/what-is-sentiment-analysis/) и [распознавание именованных сущностей](https://translated.turbopages.org/proxy_u/en-ru.ru.412cad94-680dcc27-421e0f8a-74722d776562/https/www.geeksforgeeks.org/named-entity-recognition/), чтобы определить функцию и контекст отдельных слов в предложении.
* **Управление словарным запасом:** Токенизация помогает управлять словарным запасом корпуса текстов, создавая список отдельных токенов, которые заменяют слова в наборе данных.
* **Адаптация к конкретным задачам:** токенизацию можно настроить в соответствии с потребностями конкретных задач НЛП, что означает, что она будет лучше всего работать в таких приложениях, как обобщение и машинный перевод.
* **Этап предварительной обработки:** этот важный этап предварительной обработки преобразует необработанный текст в формат, подходящий для дополнительного статистического и вычислительного анализа.

**Реализация для токенизации**

**Токенизация предложения использование sent\_tokenize**

Фрагмент кода использует**функцию sent\_tokenize** из библиотеки NLTK. Функция sent\_tokenize используется для разделения заданного текста на список предложений.

**Python3**

|  |
| --- |
| from nltk.tokenize import sent\_tokenize    text = "Hello everyone. Welcome to GeeksforGeeks. You are studying NLP article."  sent\_tokenize(text) |

**Выходной сигнал:**

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

***Как работает sent\_tokenize ?***

*Функция sent\_tokenize использует экземпляр PunktSentenceTokenizer из модуля nltk.tokenize.punkt, который уже обучен и поэтому точно знает, какими символами и знаками препинания обозначать конец и начало предложения.*

**Токенизация предложения с помощью PunktSentenceTokenizer**

Когда у нас есть огромные массивы данных, эффективнее использовать ‘PunktSentenceTokenizer' из библиотеки NLTK. Токенизатор Punkt — это основанный на данных токенизатор предложений, который поставляется вместе с NLTK. Он обучается на большом массиве текстов для определения границ предложений.

**Python3**

|  |
| --- |
| import nltk.data   # Loading PunktSentenceTokenizer using English pickle file  tokenizer = nltk.data.load('tokenizers/punkt/PY3/english.pickle')  tokenizer.tokenize(text) |

**Выходной сигнал:**

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

**Обозначить предложение на другом языке**

Можно также токенизировать предложения на разных языках, используя файлы pickle, отличные от английского. В следующем фрагменте кода мы использовали библиотеку NLTK для токенизации испанского текста на предложения с помощью предварительно обученного токенизатора Punkt для испанского языка. Токенизатор Punkt — это основанный на данных токенизатор, который использует методы машинного обучения для определения границ предложений.

**Python3**

|  |
| --- |
| import nltk.data  spanish\_tokenizer =  nltk.data.load('tokenizers/punkt/PY3/spanish.pickle')   text = 'Hola amigo. Estoy bien.'  spanish\_tokenizer.tokenize(text) |

**Выходной сигнал:**

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

**Токенизация слова с помощью work\_tokenize**

В этом фрагменте кода используется функция word\_tokenize из библиотеки NLTK для разбиения заданного текста на отдельные слова. Функция word\_tokenize полезна для разбиения предложения или текста на составляющие его слова, что упрощает дальнейший анализ или обработку на уровне слов в задачах обработки естественного языка.

**Python3**

|  |
| --- |
| from nltk.tokenize import word\_tokenize  text = "Hello everyone. Welcome to GeeksforGeeks."  word\_tokenize(text) |

**Выходной сигнал:**

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

***Как word\_tokenize работает?***

*Функция word\_tokenize() — это функция-оболочка, которая вызывает функцию tokenize() для экземпляра класса TreebankWordTokenizer.*

**Токенизация слов С помощью TreebankWordTokenizer**

Фрагмент кода использует TreebankWordTokenizer Natural Language Toolkit (NLTK) для разбиения заданного текста на отдельные слова.

**Python3**

|  |
| --- |
| from nltk.tokenize import TreebankWordTokenizer  tokenizer = TreebankWordTokenizer()  tokenizer.tokenize(text) |

**Выходной сигнал:**

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

Эти токенизаторы работают, разделяя слова с помощью знаков препинания и пробелов. И, как указано в выходных данных кода выше, они не удаляют знаки препинания, позволяя пользователю решать, что делать с ними во время предварительной обработки.

**Токенизация слов с помощью WordPunctTokenizer**

The WordPunctTokenizer — это один из токенизаторов NLTK, который разделяет слова по границам знаков препинания. Каждый знак препинания рассматривается как отдельный токен.

**Python3**

|  |
| --- |
| from nltk.tokenize import WordPunctTokenizer  tokenizer = WordPunctTokenizer()  tokenizer.tokenize("Let's see how it's working.") |

**Выходной сигнал:**

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

**Токенизация слова с использованием регулярного выражения**

Фрагмент кода использует RegexpTokenizerиз[Natural Language Toolkit (NLTK)](https://translated.turbopages.org/proxy_u/en-ru.ru.412cad94-680dcc27-421e0f8a-74722d776562/https/www.geeksforgeeks.org/tokenize-text-using-nltk-python/)для токенизации заданного текста на основе шаблона регулярного выражения.

**Python3**

|  |
| --- |
| from nltk.tokenize import RegexpTokenizer  tokenizer = RegexpTokenizer(r'\w+')  text = "Let's see how it's working."  tokenizer.tokenize(text) |

**Выходной сигнал:**

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

Использование регулярных выражений позволяет более точно контролировать токенизацию, и вы можете настроить шаблон в соответствии со своими требованиями.

**Дополнительные методы токенизации**

Мы обсудили способы реализации токенизации с помощью библиотеки NLTK. Мы также можем реализовать токенизацию с помощью следующих методов и библиотек:

* **Spacy:** [Spacy](https://translated.turbopages.org/proxy_u/en-ru.ru.412cad94-680dcc27-421e0f8a-74722d776562/https/www.geeksforgeeks.org/tokenization-using-spacy-library/) — это библиотека НЛП, которая предоставляет широкие возможности токенизации.
* **Токенизатор BERT:**[BERT](https://translated.turbopages.org/proxy_u/en-ru.ru.412cad94-680dcc27-421e0f8a-74722d776562/https/www.geeksforgeeks.org/explanation-of-bert-model-nlp/) использует токенизатор WordPiece, который является разновидностью токенизатора подслов для токенизации входного текста. Использование регулярных выражений позволяет более точно контролировать токенизацию, и вы можете настроить шаблон в соответствии с вашими конкретными требованиями.
* **Кодирование байтовых пар:**[кодирование байтовых пар (BPE)](https://translated.turbopages.org/proxy_u/en-ru.ru.412cad94-680dcc27-421e0f8a-74722d776562/https/www.geeksforgeeks.org/byte-pair-encoding-bpe-in-nlp/) — это алгоритм сжатия данных, который также нашёл применение в области обработки естественного языка, в частности для токенизации. Это [метод токенизации по частям слова,](https://translated.turbopages.org/proxy_u/en-ru.ru.412cad94-680dcc27-421e0f8a-74722d776562/https/www.geeksforgeeks.org/subword-tokenization-in-nlp/)который работает путём итеративного объединения наиболее часто встречающихся пар последовательных байтов (или символов) в заданном корпусе.
* **Sentence Piece:**SentencePiece — это ещё один алгоритм токенизации по частям слов, обычно используемый для задач обработки естественного языка. Он не зависит от языка и работает путём итеративного объединения часто встречающихся последовательностей символов или частей слов в заданном корпусе.

**Ограничения токенизации**

* Токенизация не может уловить смысл предложения, что приводит к **неоднозначности**.
* В некоторых языках, таких как китайский, японский, арабский, нет чётких пробелов между словами. Следовательно, **отсутствуют чёткие границы,** что усложняет процесс токенизации.
* Текст также может содержать более одного слова, например адрес электронной почты, URL-адреса и **специальные символы**, поэтому сложно решить, как токенизировать такие элементы.