

## Understand subword-based tokenization algorithm used by state-of-the-art NLP models — WordPiece



 362



Photo by [Glen](#) on [Unsplash](#)

Over the past few years, there has been a lot of buzz in the field of AI and especially NLP. 😊 Understanding and analyzing human language is not only a challenging problem but fascinating as well. The human language looks simple but is very complicated as even a short text can contain references to both personal life and the outside world. 😬 This complexity brings a lot of challenges. Researchers around the world are working to overcome these challenges and are building smarter real-world applications. 🧑💻

When we start working with text, we perform a set of pre-processing steps to convert text to numbers. These steps are crucial in any model development process or even in analyzing texts. In this multi-stage pre-processing process, one important step is tokenization which again can be of different types. There is a word, subword, and character-based tokenization. Each has its own purpose, advantage, and disadvantage. Let us first know a bit about the subword-based tokenization algorithm.

## **Subword-based tokenization**

Subword-based tokenization is a solution between word and character-based tokenization. The main idea is to solve the issues faced by word-based tokenization (very large vocabulary size, large number of OOV tokens, and different meaning of very similar words) and character-based tokenization (very long sequences and less meaningful individual tokens).

The subword-based tokenization algorithms do not split the frequently used words into smaller subwords. It rather splits the rare words into smaller meaningful subwords. For example, “boy” is not split but “boys” is split into “boy” and “s”. This helps the model learn that the word “boys” is formed using the word “boy” with slightly different meanings but the same root word.

Some of the popular subword-based tokenization algorithms are WordPiece, Byte-Pair Encoding (BPE), Unigram, and SentencePiece. We will go through WordPiece algorithm in this article. WordPiece is used in language models

like BERT, DistilBERT, Electra. There are two implementations of WordPiece algorithm — bottom-up and top-bottom. The original bottom-up approach is based on BPE. BERT uses the top-down implementation of WordPiece. In this article, I will talk about the original bottom-up implementation based on BPE.

In case you want to know the difference between the three tokenization techniques then you can read [this](#) article, a hands-on tutorial on TDS. 🥰

### **Word, Subword, and Character-Based Tokenization: Know the Difference**

The differences that anyone working on an NLP project should know

[towardsdatascience.com](https://towardsdatascience.com)

Let us get started with the WordPiece algorithm. 🏃

## **WordPiece**

WordPiece is a subword-based tokenization algorithm. It was first outlined in the paper “[Japanese and Korean Voice Search \(Schuster et al., 2012\)](#)”. The algorithm gained popularity through the famous state-of-the-art model BERT. This algorithm is not very different from BPE, so I recommend you first understand BPE before going through this article.

In case you are looking for a good source, here is an article on the Byte-Pair Encoding (BPE) algorithm. 🙄 You can read this article which will explain to you the step-by-step process followed by the BPE algorithm.

### **Byte-Pair Encoding: Subword-based tokenization algorithm**

Understand subword-based tokenization algorithm used by state-of-the-art NLP models — Byte-Pair Encoding (BPE)

[towardsdatascience.com](https://towardsdatascience.com)

BPE takes a pair of tokens (bytes), looks at the frequency of each pair, and merges the pair which has the highest combined frequency. The process is greedy as it looks for the highest combined frequency at each step.

So, what's the problem with BPE? 🤔 It can have instances where there is more than one way to encode a particular word. It then gets difficult for the algorithm to choose subword tokens as there is no way to prioritize which one to use first. Hence, the same input can be represented by different encodings impacting the accuracy of the learned representations. 🧐

Look at the following subword tokens table.

Number	Token	Frequency
1	li	3
2	l	5
3	ea	2
4	eb	4
5	near	6
6	bra	2
7	al	4
8	n	5
9	r	1
10	ge	11
11	g	7
12	e	8

Suppose this is the vocabulary for a small corpus and we want to tokenize our input phrase “linear algebra”. We can tokenize it as follows:

linear = **li** + **near** *or* **li** + **n** + **ea** + **r**

algebra = **al** + **ge** + **bra** *or* **al** + **g** + **e** + **bra**

We can see that there are two different ways to tokenize each word in the given phrase, giving a total of four ways to tokenize this phrase. So, the same input text can be encoded in four ways and this is indeed a problem. 🤖

Whenever we think about advancement/improvement in any domain, we always look for a better and more realistic approach. One approach which might work better than BPE's approach of frequency is to take into account the impact which merging of a particular byte-pair (symbol pair) has at each step. 👍

In data science, when we move a step ahead of frequency or count, we look for probabilistic approaches. In WordPiece, we actually do the same. The only difference between WordPiece and BPE is the way in which symbol pairs are added to the vocabulary. At each iterative step, WordPiece chooses a symbol pair which will result in the largest increase in likelihood upon merging. Maximizing the likelihood of the training data is equivalent to finding the symbol pair whose probability divided by the probability of the first followed by the probability of the second symbol in the pair is greater than any other symbol pair.

For example, the algorithm will check if the probability of occurrence of “ac”

[Open in app](#) ↗

Medium

🔍 Search

✍ Write



any other symbol pair.

Thus, we can say that WordPiece evaluates what it will lose by merging the two symbols to ensure that the step it is taking is actually worth it or not. 😊

The WordPiece algorithm is iterative and the summary of the algorithm according to the [paper](#) is as follows:

1. Initialize the word unit inventory with the base characters.



2. Build a language model on the training data using the word inventory from 1.
3. Generate a new word unit by combining two units out of the current word inventory. The word unit inventory will be incremented by 1 after adding this new word unit. The new word unit is chosen from all the possible ones so that it increases the likelihood of the training data the most when added to the model.
4. Goto 2 until a pre-defined limit of word units is reached or the likelihood increase falls below a certain threshold.

You must be thinking that training must be a computationally expensive procedure if done using brute force. If yes, then you are right. 🤖 The time complexity is  $O(K^2)$  where  $K$  is the number of current word units. As for each iteration, we need to test for all possible pair combinations and train a new language model every time. However, the training algorithm can speed up significantly by following a few simple tricks as discussed in the paper. 🏃 We can test only the pairs that actually exist in the training data, test only the pairs with a significant chance of being the best (the ones with high priors), combine several clustering steps into a single iteration (possible for a group of pairs that don't affect each other). As per the paper, these greedy speed-ups helped build a 200k vocabulary for the Japanese and Korean datasets in just a few hours on a single machine. Isn't that amazing? 🥳 This inventory can then be used for language modeling, dictionary building, and decoding.

### Is it still greedy?

Yes, even after following the probabilistic approach, WordPiece algorithm is still greedy as it picks the best pair at each iteration to merge and build a tokenizer from the bottom-up (characters to pairs and so on). A fully probabilistic model would have used probability to both choose the pairs to merge and whether to merge them or not. Despite that, the algorithm is quite popular and gives great results.

So, let's summarize. 👍

WordPiece algorithm trains a language model on the base vocabulary, picks the pair which has the highest likelihood, add this pair to the vocabulary, train the language model on the new vocabulary and repeat the steps repeated until the desired vocabulary size or likelihood threshold is reached.

I hope you got an idea about the WordPiece algorithm. In case you want to know about the top-down approach as well, I encourage you to read [this](#) blog by Tensorflow.

### References:

1. <https://static.googleusercontent.com/media/research.google.com/ja//pubs/archive/37842.pdf>
2. [https://huggingface.co/transformers/tokenizer\\_summary.html](https://huggingface.co/transformers/tokenizer_summary.html)
3. [https://www.tensorflow.org/text/guide/subwords\\_tokenizer#applying\\_wordpiece](https://www.tensorflow.org/text/guide/subwords_tokenizer#applying_wordpiece)

Thank you, everyone, for reading this article. Do share your valuable feedback or suggestion. Happy reading! 📖 ✍️

Data Science

NLP

Deep Learning

Machine Learning

Artificial Intelligence



## Written by Chetna Khanna

690 Followers · Writer for Towards Data Science

Follow

Engineer — Data & ML | Love to read | Love to write | <https://www.linkedin.com/in/chetna-khanna/>

### More from Chetna Khanna and Towards Data Science



 Chetna Khanna in Towards Data Science

## What and why behind `fit_transform()` vs `transform()` in...

Scikit-learn is the most useful library for machine learning in Python programming...

Aug 25, 2020



2.2K



55



 Ahmed Besbes in Towards Data Science

## What Nobody Tells You About RAGs

A deep dive into why RAG doesn't always work as expected: an overview of the...

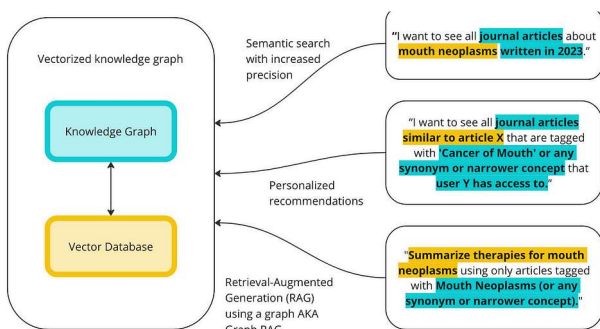
★ Aug 23



1.6K



24



 Steve Hedden in Towards Data Science



 Chetna Khanna in Towards Data Science



# How to Implement Graph RAG Using Knowledge Graphs and...

A Step-by-Step Tutorial on Implementing Retrieval-Augmented Generation (RAG),...

Sep 6 1.1K 13



# Observational vs Experimental Study

Is your statistical study observational or experimental? Let us find out.

Dec 5, 2020 94 5



See all from Chetna Khanna

See all from Towards Data Science

## Recommended from Medium



Anurag

### TF-IDF Vectorizer Explained

The Term Frequency-Inverse Document Frequency (TF-IDF) vectorizer is a widely...

Apr 30



	reducing it to its basic units or tokens.	rare or out-of-vocabulary words by breaking them down into smaller subwords.
Method	Splits text based on spaces and punctuation, treating each word or symbol as a separate token.	Uses algorithms to split words into frequently occurring subwords or character sequences, ensuring that even unseen words can be represented.
Examples	- "I love NLP" -> ["I", "love", "NLP"]  - "Happy coding!" -> ["Happy", "coding", "!"]	- "unhappiness" -> ["un", "happi", "ness"] - "transformer" -> ["trans", "form", "er"]
Applications	Basic text processing, word-based indexing, or text analysis.	Machine translation, text generation, and language modeling, especially in languages with rich morphology or when dealing with technical jargon.

Ayushman Pranav

### Exploring Subword Tokenization in Natural Language Processing wit...

In the Last article

May 7





## Predictive Modeling w/ Python

20 stories · 1548 saves



## Practical Guides to Machine Learning

10 stories · 1877 saves



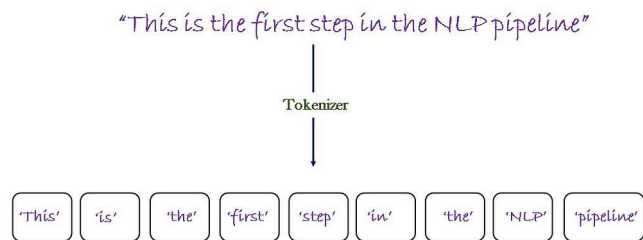
## Natural Language Processing


1715 stories · 1287 saves



## data science and AI

40 stories · 247 saves



 Abhishek Kumar Pandey

## Word Piece Tokenization Algorithm

Easy:

Mar 30  3



 Malik kashif

## New York Street

 Sep 15  808  24

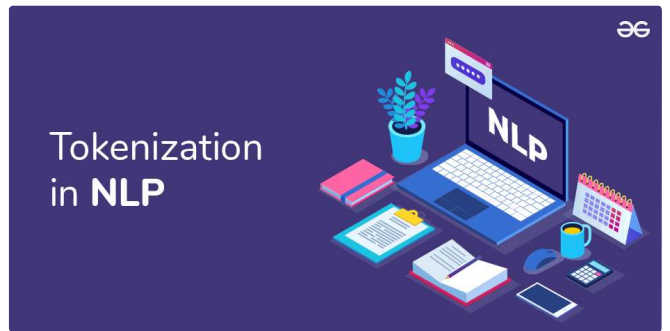


 Tayyib UI Hassan Gondal in The Deep Hub

## All you need to know about Tokenization in LLMs

In this blog, I'll explain everything about tokenization, which is an important step...

Jul 4  103



 Jainvidip

## Understanding Tokenization in Natural Language Processing...

Natural Language Processing (NLP) is a branch of artificial intelligence that focuses ...

Jul 6



[See more recommendations](#)