Lecture → Byte Pair Encoding *

Tokenization algorithms

1. Word based
2. Sub-word based
3. Character based

My hobby is playing cricket

['My', 'hobby', 'is', 'playing', 'cricket']

Problem - What do we do with Out of Vocabulary (OOV) words, different meaning of similar words [boy, boys]

football

My hobby is playing cricket

['m', 'y', 'h', ...]

Very small vocabulary. Every language has fixed number of characters (English ≈ 256) } Solves the OOV problem

Problem: The meaning associated with words is completely lost. Also, the tokenized sequence is much longer than the initial raw text.

dinosaur

8 tokens

Sub-word based Tokenization *

Sub-word based Tokenization *

Rule 1: Do not split frequently used words into smaller subwords

Rule 2: Split the rare words into smaller meaningful subwords

eg: "boy" should not be split

"boys" should be split into "boy" and "s"

① The subword splitting helps the model learn that different words with same root word as "token" like "tokens" and "tokenizing" are similar in meaning

② It also help the model learn that "tokenization"

(2) It also helps the model learn that "tokenization" and "modernization" are made up of different root words but have the same suffix "ization" and are used in same syntactic situations.

* BYTE PAIR ENCODING (BPE) is a subword tokenization algorithm *

* BPE algorithm: (1994) Most common pair of consecutive bytes of data is replaced with a byte that does not occur in data


A new algorithm for data compression

(3) Let us take a simple example to understand this (source: Wiki)

Original data: aaabdaaabac

(a) The byte pair 'aa' occurs the most. We will replace it with Z as Z does not occur in the data

(b) Compressed data: ZabdZabac

(c) The next common byte pair is 'ab'. We will replace this by Y

(c) The next common byte pair is 'ab'. We will replace this by Y

(d) Compressed data: ZYdZYac

ZYdZYac

only byte pair left Appears only once, so we do not encode it

(4) How is the BPE* algorithm used for LLMs?

(a) BPE ensures that most common words in the vocabulary are represented as a single token, while rare words are broken down into two or more subword tokens.

(b) Let us take a practical example

* Let us consider the below dataset of words

{"old": 7, "older": 3, "finest": 9, "lowest": 4}

* Preprocessing: We need to add end token "</w>" at the end of each word.

{"old</w>": 7, "older</w>": 3, "finest</w>": 9, "lowest</w>": 4}

{"old</w>": 7, "older</w>": 3, "finest</w>": 9, "lowest</w>": 4}

* Let us now split words into characters and count their frequency.

Number	Token	Frequency
1	</w>	23
2	o	14
3	l	14
4	d	10
5	e	16
6	r	3
7	f	9
8	i	9
9	n	9
10	s	13
11	t	13
12	w	4

(c) Next step in the BPE algorithm is to look for the most frequent pairing.

→ Merge them and perform the same iteration again and again until we reach the token limit or iteration limit.

(d) Iteration 1: Start with second most common token "e"

Most common byte pair starting with e: "es"

(d) Iteration 1: Start with second most common token "e"

Most common byte pair starting with e: "es"

Number	Token	Frequency
1	</w>	23
2	o	14
3	l	14
4	d	10
5	e	16 - 13 = 3
6	r	3
7	f	9
8	i	9
9	n	9
10	s	13 - 13 = 0
11	t	13
12	w	4
13	es	9 + 4 = 13

(e) Iteration 2: Merge the tokens "es" and "t" as they have appeared 13 times in our dataset

(e) Iteration 2: Merge the tokens "es" and "t" as they have appeared 13 times in our dataset.

Number	Token	Frequency
1	</w>	23
2	o	14
3	l	14
4	d	10
5	e	16 - 13 = 3
6	r	3
7	f	9
8	i	9
9	n	9
10	s	13 - 13 = 0
11	t	13 - 13 = 0
12	w	4
13	es	9 + 4 = 13 - 13 = 0
14	est	13

(f) Now let us look at the "</w>" token.

(f) Now let us look at the "</w>" token.

We see that "est </w>" has appeared 13 times.

Number	Token	Frequency
1	</w>	23 - 13 = 10
2	o	14
3	l	14
4	d	10
5	e	16 - 13 = 3
6	r	3
7	f	9
8	i	9
9	n	9
10	s	13 - 13 = 0
11	t	13 - 13 = 0
12	w	4
13	es	9 + 4 = 13 - 13 = 0
14	est	13 - 13 = 0
15	est</w>	13

→ helps algorithm understand difference between estimate and highest.

(g) Iteration 4: "o" and "l" has appeared 10 times.

Number	Token	Frequency
1	</w>	23

(g) Iteration 4: "o" and "l" has appeared 10 times

Number	Token	Frequency
1	</w>	23
2	o	14 - 10 = 4
3	l	14 - 10 = 4
4	d	10
5	e	16 - 13 = 3
6	r	3
7	f	9
8	i	9
9	n	9
10	s	13 - 13 = 0
11	t	13 - 13 = 0
12	w	4
13	es	9 + 4 = 13 - 13 = 0
14	est	13
15	ol	7 + 3 = 10

(h) Iteration 5: "ol" and "d" has appeared 10 times

Number	Token	Frequency
1	</w>	23 - 13 = 10
2	o	14 - 10 = 4

(h) Iteration 5: "ol" and "d" has appeared 10 times.

Number	Token	Frequency
1	</w>	23 - 13 = 10
2	o	14 - 10 = 4
3	l	14 - 10 = 4
4	d	10 - 10 = 0
5	e	16 - 13 = 3
6	r	3
7	f	9
8	i	9
9	n	9
10	s	13 - 13 = 0
11	t	13 - 13 = 0
12	w	4
13	es	9 + 4 = 13 - 13 = 0
14	est	13 - 13 = 0
15	est</w>	13
16	ol	7 + 3 = 10 - 10 = 0
17	old	7 + 3 = 10

(i) "f", "i", "n" appear 9 times. But we just have one word with these characters. So, we are not merging them.

(j) Let us remove tokens with zero count.

(j) Let us remove tokens with zero count.

Number	Token	Frequency
1	</w>	10
2	o	4
3	l	4
4	e	3
5	r	3
6	f	9
7	i	9
8	n	9
9	w	4
10	est</w>	13
11	old	10

This list of 11 tokens will serve as our vocabulary.

Number	Token	Frequency
1	</w>	10
2	o	4
3	l	4
4	e	3
5	r	3
6	f	9
7	i	9
8	n	9
9	w	4
10	est</w>	13
11	old	10

This list of 11 tokens will serve as our vocabulary.

The stopping criteria can either be the token count or the number of iterations.