Yuqing Qiao
1/27/2024

Byte Pair Encoding (BPE) Implementation and Evaluation on NLTK Dataset

1. Implementation of Byte Pair Encoding (BPE) Algorithm

The implementation of the BPE algorithm can be summarized through the following key components of the BPE class:

1. Initialization of Corpus and Vocabulary (init_corpus)

   Tokenization: It tokenizes each word in the dataset into single characters. This tokenization includes adding a special end-of-word symbol (_) to each word to distinguish between naturally occurring character sequences and those at the end of words.

   Frequency Count: It counts the frequency of each unique tokenized word.

   Vocabulary Initialization: It initializes the vocabulary with unique characters found in the corpus and the special end-of-word symbol.

2. Identifying Frequent Pairs

   The get_pairs method identifies the most frequent pairs of tokens (characters) in the corpus.

3. Merging the Corpus and Updating the Vocabulary

   The merge_corpus method takes the most frequent pair identified by get_pairs and merges occurrences of this pair throughout the corpus. This process involves replacing each occurrence of the pair with a new token that combines the two characters.

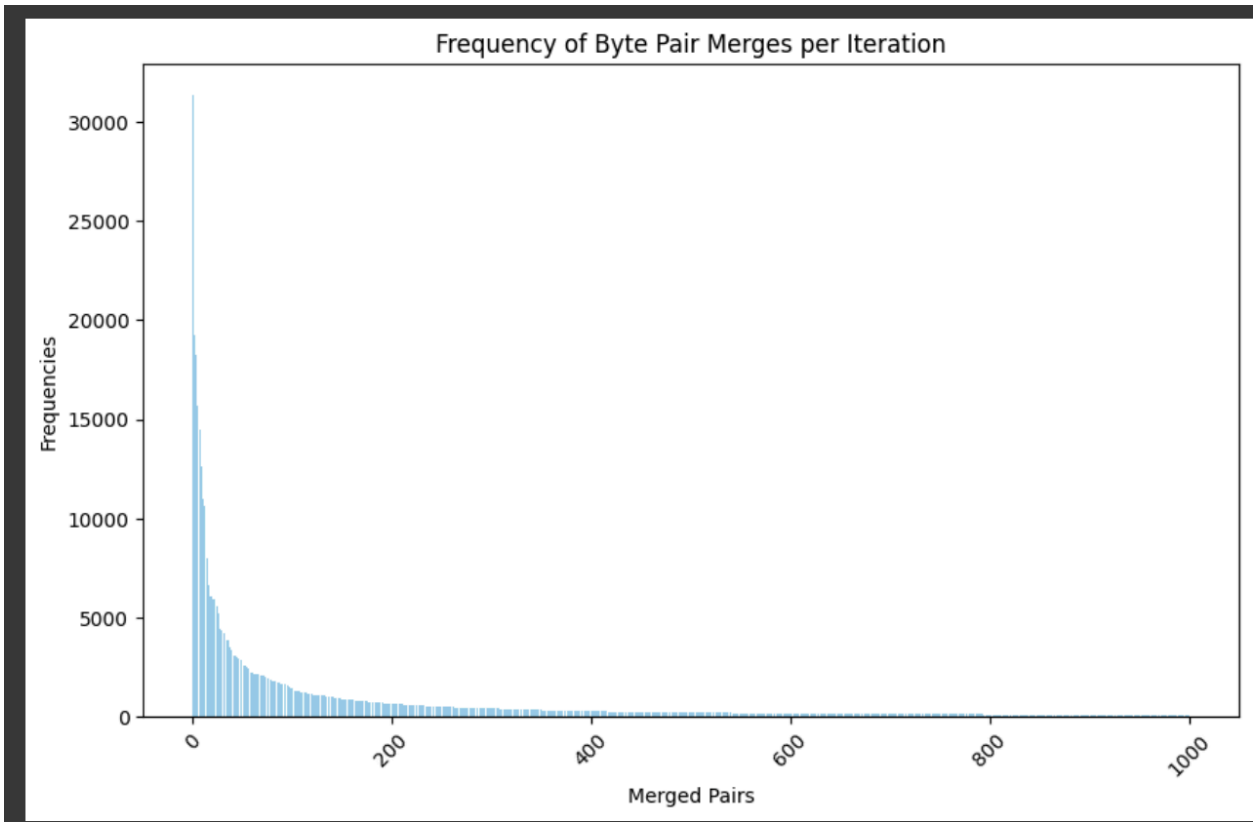
2. Experimental Setup

Custom sample sentences were used to test and debug the BPE class and the effect of varying the hyperparameter 'k', which controls the number of merges, is also tested. The vocabulary was debugged in each iteration to monitor the merges in the corpus.

Additionally, a visualization method was developed to plot a graph showing the relationship between vocabulary size and the frequency of merges.

3. Result

The visualization graph shows the relationship between vocabulary size and the frequency of merges indicates that the frequency of merges converges and lowers as the vocabulary size grows. Additionally, the accuracy and coverage of the BPE algorithm were evaluated, revealing an accuracy of approximately 32% and a coverage of 4% with hyperparameter k set to 1000 merges. The low accuracy and coverage can be attributed to the presence of numerous subwords in the BPE vocabulary that do not align with those in the referenced standard tokenization in NLTK word_tokenize. It takes a larger number of merges to improve the accuracy and coverage.



Successfully shows BPE merge most frequent pair in corpus and add the best to the vocab:

```
--After 6 iteration
Most freq pairs:  ('h', 'i')
Corpus after update:  {'lower_': 1, 'low_': 1, 'hi g h er . _': 1}

--After 7 iteration
Most freq pairs:  ('hi', 'g')
Corpus after update:  {'lower_': 1, 'low_': 1, 'hig h er . _': 1}

--After 8 iteration
Most freq pairs:  ('hig', 'h')
Corpus after update:  {'lower_': 1, 'low_': 1, 'high er . _': 1}

--After 9 iteration
Most freq pairs:  ('high', 'er')
Corpus after update:  {'lower_': 1, 'low_': 1, 'higher . _': 1}

Vocab:  {'tal', 'Oh', 'cla', 'ac', 'ut', ';', 'ation._', 'les_', 'ent,_', '
Corpus:  {'lower_': 1, 'low_': 1, 'higher . _': 1}
Decoding:  ['lower', 'low', 'higher']
```

4. Strength

BPE is effective at dealing with rare words. It can represent and process words using subwords that are not seen during training. BPE allows an efficient representation of the corpus, by merging frequent subwords.

5. Weaknesses

BPE does not consider the context of words. BPE may create subwords or words that are not meaningful. The hyperparameter K significantly affects the performance.

6. Challenges

Debugging and identifying the corpus structure to reveal merged subwords is a challenge to me. Considerable research was also dedicated to understanding the creation of a vocabulary composed of BPE-generated subwords, as opposed to traditional word tokenization.