Byte pair encoding

Byte pair encoding[1][2] (also known as **digram coding**)[3] is an algorithm, first described in 1994 by Philip Gage for encoding strings of text into tabular form for use in downstream modeling.[4] Its modification is notable as the large language model tokenizer with an ability to combine both tokens that encode single characters (including single digits or single punctuation marks) and those that encode whole words (even the longest compound words).[5][6][7] This modification, in the first step, assumes all unique characters to be an initial set of 1-character long n-grams (i.e. initial "tokens"). Then, successively the most frequent pair of adjacent characters is merged into a new, 2-character long n-gram and all instances of the pair are replaced by this new token. This is repeated until a vocabulary of prescribed size is obtained. Note that new words can always be constructed from final vocabulary tokens and initial-set characters.[8]

All the unique tokens found in a corpus are listed in a token vocabulary, the size of which, in the case of GPT-3.5 and GPT-4, is 100256.

The difference between the modified and the original algorithm is that the original algorithm does not merge the most frequent pair of bytes of data, but replaces them by a new byte that was not contained in the initial dataset. A lookup table of the replacements is required to rebuild the initial dataset. The algorithm is effective for tokenization because it has low computational overhead and remains consistent and reliable.

Original algorithm

The original algorithm operates by iteratively replacing the most common contiguous sequences of characters in a target text with unused 'placeholder' bytes. The iteration ends when no sequences can be found, leaving the target text effectively compressed. Decompression can be performed by reversing this process, querying known placeholder terms against their corresponding denoted sequence, using a lookup table. In the original paper, this lookup table is encoded and stored alongside the compressed text.

Example

Suppose the data to be encoded is

```
aaabdaaabac
```

The byte pair "aa" occurs most often, so it will be replaced by a byte that is not used in the data, such as "Z". Now there is the following data and replacement table:

```
ZabdZabac
Z=aa
```

Then the process is repeated with byte pair "ab", replacing it with "Y":

```
ZYdZYac
Y=ab
Z=aa
```

The only literal byte pair left occurs only once, and the encoding might stop here. Alternatively, the process could continue with recursive byte pair encoding, replacing "ZY" with "X":

```
XdXac
X=ZY
Y=ab
Z=aa
```

This data cannot be compressed further by byte pair encoding because there are no pairs of bytes that occur more than once.

To decompress the data, simply perform the replacements in the reverse order.

See also

- Re-Pair
- Sequitur algorithm

References

1. Gage, Philip (1994). "A New Algorithm for Data Compression" (http://www.pennelynn.com/Documents/CUJ/HTML/94HTML/19940045.HTM). *The C User Journal*.
2. "A New Algorithm for Data Compression" (http://www.drdobbs.com/article/print?articleId=184402829). *Dr. Dobb's Journal*. 1 February 1994. Retrieved 10 August 2020.
3. Witten, Ian H.; Moffat, Alistair; Bell, Timothy C. (1994). *Managing Gigabytes*. New York: Van Nostrand Reinhold. ISBN 978-0-442-01863-4.
4. "Byte Pair Encoding" (https://web.archive.org/web/20160326130908/http://www.csse.monash.edu.au/cluster/RJK/Compress/problem.html). Archived from the original (http://www.csse.monash.edu.au/cluster/RJK/Compress/problem.html) on 2016-03-26.
5. Sennrich, Rico; Birch, Alexandra; Haddow, Barry (2015-08-31). "Neural Machine Translation of Rare Words with Subword Units". arXiv:1508.07909 (https://arxiv.org/abs/1508.07909) [cs.CL (https://arxiv.org/archive/cs.CL)].
6. Brown, Tom B.; Mann, Benjamin; Ryde r, Nick; Subbiah, Melanie; Kaplan, Jared; Dhariwal, Prafulla; Neelakantan, Arvind; Shyam, Pranav; Sastry, Girish; Askell, Amanda; Agarwal, Sandhini (2020-06-04). "Language Models are Few-Shot Learners". arXiv:2005.14165 (https://arxiv.org/abs/2005.14165) [cs.CL (https://arxiv.org/archive/cs.CL)].
7. "google/sentencepiece" (https://github.com/google/sentencepiece). Google. 2021-03-02. Retrieved 2021-03-02.
8. Paaß, Gerhard; Giesselbach, Sven (2022). "Pre-trained Language Models" (https://link.springer.com/chapter/10.1007/978-3-031-23190-2_2). *Foundation Models for Natural Language Processing*. Artificial Intelligence: Foundations, Theory, and Algorithms. pp. 19–78. doi:10.1007/978-3-031-23190-2_2 (https://doi.org/10.1007%2F978-3-031-23190-2_2). ISBN 9783031231902. Retrieved 3 August 2023.