WIKIPEDIA
The Free Encyclopedia

Byte pair encoding

Byte pair encoding[1][2] (also known as **BPE**, or **digram coding**)[3] is an algorithm, first described in 1994 by Philip Gage, for encoding strings of text into smaller strings by creating and using a translation table.[4] A slightly-modified version of the algorithm is used in large language model tokenizers.

The original version of the algorithm focused on compression. It replaces the highest-frequency pair of bytes with 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 modified version builds "tokens" (units of recognition) that match varying amounts of source text, from single characters (including single digits or single punctuation marks) to whole words (even long compound words).[5][6][7]

Original algorithm

The original BPE 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[8]

```
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.

Modified algorithm

The original BPE algorithm is modified for use in language modeling, especially for large language models based on neural networks. Compared to the original BPE, the modified BPE does not aim to maximally compress text, but rather, to encode plaintext into "tokens", which are natural numbers.[9] 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.[10]

The modified tokenization algorithm initially treats the set of unique characters as 1-character-long n-grams (the initial tokens). Then, successively, the most frequent pair of adjacent tokens is merged into a new, longer 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.[11]

This modified BPE approach has been extended from spoken language to sign language in recent years.[12]

Example

Suppose we are encoding the previous example of "aaabdaaabac", with a specified vocabulary size of 6, then we would first be encoded as "0, 0, 0, 1, 2, 0, 0, 0, 1, 0, 3" with a vocabulary of "a=0, b=1, d=2, c=3". Then it would proceed as before, and obtain "4, 5, 2, 4, 5, 0, 3" with a vocabulary of "a=0, b=1, d=2, c=3, aa=4, ab=5".

So far this is essentially the same as before. However, if we only had specified a vocabulary size of 5, then the process would stop at vocabulary "a=0, b=1, d=2, c=3, aa=4", so that the example would be encoded as "4, 0, 1, 2, 4, 0, 1, 0, 3". Conversely, if we had specified a vocabulary size of 8, then it would be encoded as "7, 6, 0, 3", with a vocabulary of "a=0, b=1, d=2, c=3, aa=4, ab=5, aaab=6, aaabd=7". This is not maximally efficient, but the modified BPE does not aim to maximally compress a dataset, but aim to encode it efficiently for language model training.[13]

Byte-level BPE

In the above example, the output of the BPE is a vocabulary, which can be used to encode any text that is written with the letters "abcd". It will not be able to encode text containing other symbols, such as "no". Even giving each of the 26 letters an entry in the vocabulary, since there are many languages in the world using many different scripts, inevitably some symbols would be unencodable by such a vocabulary.

One solution is to replace any unencodable symbol with a special symbol named UNK ("unknown").

The byte-level BPE is another approach. It simply converts the text into UTF-8 first, and treat it as a stream of bytes. This guarantees that any text encoded in UTF-8 can be encoded by the BPE. This has been used in BERT-like models like RoBERTa, BART, and DeBERTa, and GPT-like models like GPT-2.[14][15][16]

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. Campesato, Oswald (2024-12-26). *Large Language Models for Developers: A Prompt-based Exploration of LLMs*. Walter de Gruyter GmbH. ISBN 978-1-5015-2095-2.
9. "Counting Ability of Large Language Models and Impact of Tokenization" (https://arxiv.org/html/2410.19730v1). *arXiv*. Retrieved 2025-04-07.
10. "What is BBPE - Tokenizer behind LLMs" (https://www.kaggle.com/code/binfeng2021/what-is-bbpe-tokenizer-behind-llms). *Kaggle*. Retrieved 2025-04-07.
11. Paaß, Gerhard; Giesselbach, Sven (2022). "Pre-trained Language Models". *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.
12. Taro Miyazaki, Sihan Tan, Tsubasa Uchida, Hiroyuki Kaneko (May 25, 2024). "Sign Language Translation with Gloss Pair Encoding" (https://aclanthology.org/2024.signlang-1.29.pdf) (PDF). *Proceedings of the 11th Workshop on the Representation and Processing of Sign Languages*.
13. Pai, Suhas (2025-03-06). *Designing Large Language Model Applications: A Holistic Approach to LLMs*. O'Reilly Media. ISBN 978-1-0981-5046-4.
14. "Byte-Pair Encoding tokenization" (https://huggingface.co/learn/nlp-course/en/chapter6/5). *Hugging Face NLP Course*. Retrieved 2025-01-27.

15. Yıldırım, Savaş; Chenaghlu, Meysam Asgari (2021-09-15). *Mastering Transformers: Build state-of-the-art models from scratch with advanced natural language processing techniques*. Packt Publishing Ltd. ISBN 978-1-80107-889-4.
16. Wang, Changhan; Cho, Kyunghyun (2020-04-03). "Neural Machine Translation with Byte-Level Subwords" (https://doi.org/10.1609%2Faaai.v34i05.6451). *AAAI Conference on Artificial Intelligence*. **34** (05): 9154–9160 (https://pdfguru.com/compress-pdf). arXiv:1909.03341 (https://arxiv.org/abs/1909.03341). doi:10.1609/aaai.v34i05.6451 (https://doi.org/10.1609%2Faaai.v34i05.6451). ISSN 2374-3468 (https://search.worldcat.org/issn/2374-3468).