Investigating the Effectiveness of BPE: The Power of Shorter Sequences

Matthias Gallé
NAVER LABS Europe
matthias.galle@naverlabs.com

Abstract

Byte-Pair Encoding (BPE) is an unsupervised sub-word tokenization technique, commonly used in neural machine translation and other NLP tasks. Its effectiveness makes it a *de facto* standard, but the reasons for this are not well understood. We link BPE to the broader family of dictionary-based compression algorithms and compare it with other members of this family. Our experiments across datasets, language pairs, translation models, and vocabulary size show that – given a fixed vocabulary size budget – the fewer tokens an algorithm needs to cover the test set, the better the translation (as measured by BLEU).

1 Introduction

It is common practice in modern NLP to represent text in a continuous space. However, the interface between the discrete world of language and the continuous representation persists and is arguably nowhere so important as in deciding what should be the atomic symbols that will be used as input and/or output. While using words as tokens seems to be the most natural encoding, it has two major drawbacks: first it suffers from the out-of-vocabulary problems at inference time, and secondly the large number of words (which can easily reach the hundred of thousands) creates additional constraints on the memory and time that is required to train and infer with these models. On the other extreme, working at the character level has the advantage of a small and closed set. While current performance of character-level Neural Machine Translation (NMT) is competitive, its computational cost are much higher (up to 8 times (Cherry et al., 2018)). The commonly used intermediate approach is to pre-process the dataset by inferring sub-word tokens, which are then used as the atomic symbols to be translated. The most popular such method is known as **byte-pair encoding** (BPE), which merges iteratively the most frequent pair of symbols. Starting with characters as individual symbols, BPE creates new symbols representing bigrams and eventually higher-level order grams.

Such an approach has the advantage of incorporating some dataset-wide information in the local decision of what should be a token. It is however not linguistically motivated, and intents to create sub-word tokens based on morphological analyzer have failed to consistently improve the performance of the resulting translation (Zhou, 2018; Domingo et al., 2018), or do so only in special cases (morphological rich cases, target side only) (Huck et al., 2017; Pinnis et al., 2017). While this is unsatisfactory, to our knowledge the reasons for the effectivenesses of BPE has not been studied in the literature.

In this paper we hypothesize that the reason lies in the compression capacity of BPE: this is, that for two token vocabularies of the same size, the one that allows to cover better (with fewer number of symbols) a given sentence will achieve a better translation. To test this hypothesis, we link BPE to a large family of compression algorithms that aim to discover a dictionary of tokens and use that dictionary to parse the original dataset with the least amount of tokens. This allows us to take different algorithms from this family and to compare different unsupervised token inference techniques, all guided by the same compression principle. For a fixed number of token types, some of these algorithms need more tokens to cover the full sentence than BPE while others need less. Using different NMT models and experimenting across language pairs, datasets and vocabulary size, we show that the average number of tokens per sentence can be directly linked to the quality of translation.

Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pages 1375–1381, Hong Kong, China, November 3–7, 2019.

2 Related Work and Context

2.1 Sub-word tokenization for NMT

Instead of relying on characters or words, unsupervised sub-word tokenization techniques induce a set of tokens that are then used to parse any new sequence. The standard approach is BPE, introduced for NMT in Sennrich et al. (2016) which merges incrementally the most frequent bigrams. Kudo (2018) proposes a different technique which allows the model to sample different tokenizations from the same word, resulting in a more robust translation model. The new tokenization method alone achieves comparable results, and no discussion of the size of the final parse is provided. In our experiments, we used the code of that paper.[1]

Alternative segmentations, using a morphological analyser for instance, have not shown consistent improvement (Zhou, 2018). Domingo et al. (2018) benchmark different segmentation techniques on different languages. While their results indicate that there is no clear winner, the vocabulary sizes of the different approaches are different and therefore harder to compare. Note that – differently from previous work (Cherry et al., 2018; Domingo et al., 2018) – our set-up allows us to compare models with the same vocabulary size. Comparing the compression capacity of models with different vocabulary size is misleading as a larger vocabulary size can obviously lead to smaller sequences. This however ignores the cost of expressing each symbol, which increases with the vocabulary size. Such an analysis could be achieved with an information-theoretical approach which is beyond the scope of this paper.

Indeed, BPE has the attractive characteristic that it allows to control the size of the vocabulary and can adapt to memory constraints. However, it introduces an additional hyper-parameter in the model and can be considered unsatisfactory from a modelling perspective. Recent proposals to avoid this include using several tokenizations at the same time, either hierarchically (Morishita et al., 2018) or through a lattice encoding (Su et al., 2017; Sperber et al., 2017); or dynamically, such as increasing the vocabulary size incrementally while training (Salesky et al., 2018) or reading the input sequence character by character and adding a halting gate which decides when to generate an output word (Kreutzer and Sokolov, 2018).

[1] https://github.com/google/sentencepiece

2.2 Byte-Pair Encoding

BPE is an instance of a the so-called *macro-based* compression algorithms, which seeks redundancies in the text by detecting repeated patterns, and compresses the sequences by replacing an occurrence of such a pattern with pointers to a previous occurrence. They achieve good compression by replacing subwords with (shorter) references, as opposed to statistical-based compression algorithms which are based on information theory and assign codewords to single symbols (but both can of course be combined). Its use for neural machine translation was introduced in (Sennrich et al., 2016), although a similar technique had been used previously for the creation of dictionaries and language models for voice search (Schuster and Nakajima, 2012). For the general framework, Storer and Szymanski (1982) provide a good overview of different possible frameworks of macro schemes. The authors differentiate between *external* and *internal* macro schemes. External macro schemes contain pointers to an external dictionary, while the pointers of internal macro schemes point to positions of the sequence itself.

From the external macro schemes, BPE belongs to the class of compression algorithms called **fixed size dictionary**. In this framework, the dictionary consists in a set of tokens $V = \{\omega_1, \dots \omega_n\}$ which are the atomic symbols. A sequence of ω_i has to be uniquely decodable, for instance by being of fixed length or prefix-free. In general in NMT, prefix-freeness is attained by separating these tokens with a special symbol (whitespace). Given a new sequence s (the words from the test sequences), the goal is then to find $d_1 \dots d_m$, $d_i \in V$, such that their concatenation $d_1 \cdot d_2 \cdot \dots d_m = s$ and m is minimal. This problem was proposed in 1973 by Wagner (1973) together with a dynamic algorithm that solves this problem in an optimal way. This optimal way is called **optimal parsing** by Bell et al. (1990) and "Minimal Space" by Schuegraf and Heaps (1974) where it is solved by a shortest-path algorithm. There also exist faster approximate algorithms: see Katajainen and Raita (1992) and Bell et al. (1990, Chapter 8.1).

This formulation assume that the set of words is given. Finding the set of words whose optimal parsing is minimal is an NP-complete problem: Storer and Szymanski (1982) proves this for variations where the pointers may be recursive (this is, enabling phrases themselves to be

parsed with pointer to other phrases) or the phrases may overlap. BPE is an instance of the recursive, non-overlapping version. This version is equivalent to the so-called Smallest Grammar Problem (Charikar et al., 2005), because the resulting parsing can be formalized as a non-branching, non–auto-recursive context free grammar, also called a straight-line program. This problem can be seen as a specific formulation of the Minimum Description Length principle (Rissanen, 1978) which states that the best model for some data is the one which minimizes cost of describing the data given the model *plus* the cost of describing the model. In a straight-line grammar the description of the data can be seen as the right-hand side of the starting non-terminal, while the description of the model is the combined parsing cost of all other non-terminals. This principle has been applied in a very similar form for unsupervised inference of linguistic structures in (De Marcken, 1996).

Several algorithms that approximate the Smallest Grammar Problem have been given in the literature, including the one that iteratively replaces the most frequent pair. The name BPE comes from Gage (1994) who applied it to data compression, although the same idea had been used before for pattern discovery in natural language (Wolff, 1975) as well as measuring the complexity of genetic sequences (Ángel Jiménez-Montaño, 1984). Larsson and Moffat (2000) propose the REPAIR implementation, whose time complexity is linear with respect to the size of the input sequence. A theoretical analysis of its compression capacity can be found in Navarro and Russo (2008), and recent research focuses on reducing its memory footprint (Gagie et al., 2019).

3 Other compression algorithms

We investigate the impact that different such algorithms have on the final translation, using the same budget on the vocabulary size. Our hypothesis is that smaller parses would result in improved translation, and that therefore algorithms that infer a dictionary of tokens that cover the sequence better should be preferred.

Despite its simplicity, BPE performs very well in standard compression benchmarks (Gage, 1994; Carrascosa et al., 2012). The best performing ones have an unreasonable time-complexity (including one of complexity $\mathcal{O}(n^7)$ (Carrascosa et al., 2011) and an even slower genetic algorithm (Benz and Kötzing, 2013)). Based on these benchmarks, we decided to use the so-called IRRMGP algorithm, which outperforms BPE, as well as other worse performing algorithms. **IRRMGP** works as follows: as BPE it is an iterative repeat replacement (IRR) algorithm, but instead of choosing in each iteration the most frequent bigram, it chooses the substring ω that maximizes $|\omega| \times occ(\omega)$ (where $occ(\omega)$ is the number of occurrences of ω in the training corpus). Once it has reached the desired number of tokens, it re-parses the sequences through an optimal parsing of the original sequence *and* inferred tokens so far (minimal grammar parsing, MGP), removes any un-used token and continues looking for new words. The modules of IRR and MGP are iterated until the dictionary budget is reached. While a worst-case analysis bounds its running time as $\mathcal{O}(n^4)$, in practice its running time increases quadratically with the sequence length (Carrascosa et al., 2012). Those high running times however precludes our analysis to run on very large data-sets.

We also compare against two other compression algorithm. **MAXLEN** (Bentley and McIlroy, 1999) is another iterative repeat replacement algorithm, but selects the longest repeat in each iteration. It can also be implement in linear-time (Nakamura et al., 2009). The second algorithm is the popular **SEQUITUR** (Nevill-Manning and Witten, 1997). It runs on-line, maintaining two invariants while it reads the sequence: (i) bigram uniqueness (no bigram appears more than once) and (ii) rule utility (each non-terminal is used more than once). Assuming a constant time hashing function, SEQUITUR runs in (amortized) linear time with respect to the sequence length, and only requires a linear amount of memory. In our experiments, the original implementation of SEQUITUR[2] runs about 3 times faster than SENTENCEPIECE.

3.1 Core tokens

It is not straightforward to run SEQUITUR to obtain a fixed number of tokens, as it constructs the dictionary online. For this, we run it until the end, and select a posteriori a set of good tokens. For that selection process, we use the parse tree of the final grammar and compute the "core nodes" (Siyari et al., 2016). This corresponds to the set of

[2] https://github.com/craignm/sequitur

tokens which are used the most in the hierarchical parse of the dataset. Siyari et al. (2016) show that the number of times token ω is used is equal to $|\omega| \times freq(\omega)$, where $freq(\omega)$ is the number of times ω appears in the derivation of the unique target sequence. The core is then computed by taking the highest scoring such token, removing it, updating the parse and iterating until reaching the vocabulary size.

4 Experiments

We worked on the lowercased Kyoto Free Translation Task (Neubig, 2011, KFTT) and IWSLT'14 (en↔de), without removing any sentences (see Table 1). For the translation model, we used ConvS2S (Gehring et al., 2017) and Transformer (Vaswani et al., 2017), allowing us to compare models with different field of view: local for ConvS2S, and global for Transformer. For ConvS2S, we used 4 layers and an embedding size of 256, dropout of 0.2, an initial step-size of 0.25. For Transformer, we used 4 attention heads, dropout was set to 0.5 and initial step-size to 0.001 (except for MAXLEN, for which we used 10e-4). These parameters were fixed in initial experiments maximising the performance on BPE. We ran training for 150 epochs, and use the average of the three best models (as measured on the validation set).

To apply sub-word tokenization over unseen sentences, several approaches are possible. First we tried to run an optimal parsing on each word with the given token set but the resulting translations were worse than the baseline BPE model. The reported results here are obtained with a greedy parsing, where the words in the dictionary are first sorted and then applied from the top downwards. Words are sorted first by their length (longer first) and then by their frequency (more frequent first). Using a consistent tokenization – applied equally in training and inference – seems to help the model, while in an optimal parsing setting a slight change in the context could substantially change the parsing. We believe that this brittleness causes the worse performance of the optimal parsing. Note that the standard BPE procedure also applies greedily the dictionary of words, in the order they were generated. Sub-word regularization (Kudo, 2018) could be used to make the training more robust to this tokenization mismatch. Moreover, that approach could be adapted to work with any of the inferred tokenization: while a vanilla optimal parsing does not seem to support a probabilistic approach which could allow a sampling procedure, there might be several optimal parsings of one word (Carrascosa et al. (2011) show both theoretical and empirical evidence that there can be an exponential number of parses with the same size) and generating several of those could make the translation system more robust.

We compare BPE, IRRMGP, MAXLEN, the core on the words inferred by SEQUITUR and an additional baseline RANDOM: given a vocabulary budget of N, we pick up N random *maximal repeats* (Gusfield, 1997) of the multi-sequence set of all tokens. Those are then applied to the training and testing data as before (applying them greedily, first by length – longer first – and then by frequency – more frequent first).

The training data of both languages were concatenated, and we used two different vocabulary sizes: 16k and 32k. Tokenization in all cases was done at the sub-word level, so that no token crossed word boundaries. Inference is done with a beam search of size 5, and for BLEU evaluation we used SacreBleu.[3]

The results in Fig. 1 show that there is indeed a strong correlation between the number of tokens used to cover the test sentences, and the resulting BLEU score. BPE is competitive with the best used compression algorithm, something which can be attributed to its compression capacity which is only slightly worse than that of IRRMGP (sometimes hard to distinguish for IWSLT in Fig. 1 because the numbers are relatively close).

5 Conclusion

In this paper we provide evidence that the effectiveness of BPE can be linked to its compression capacity, that is, the capacity of finding a set of words such that they are able to cover the sequence with as few words as possible. For this, we use other algorithms of the same family which allows us to control the number of tokens, as it is obviously easier to cover the sequence better if we can access more tokens. Our benchmark shows that the size of the final coverage can indeed be strongly linked to the translation quality.

Those conclusions however do not preclude the

[3] `BLEU+case.mixed+numrefs.1+smooth.exp+tok.intl+version.1.2.17`

name	lang	type	train	dev	test
KFTT	ja ↔ en	wikipedia	329.9k	1.2k	1.2k
IWSLT	de ↔ en	TED(x) talks	160.2k	7.3K	6.8K

Table 1: Datasets characteristics.

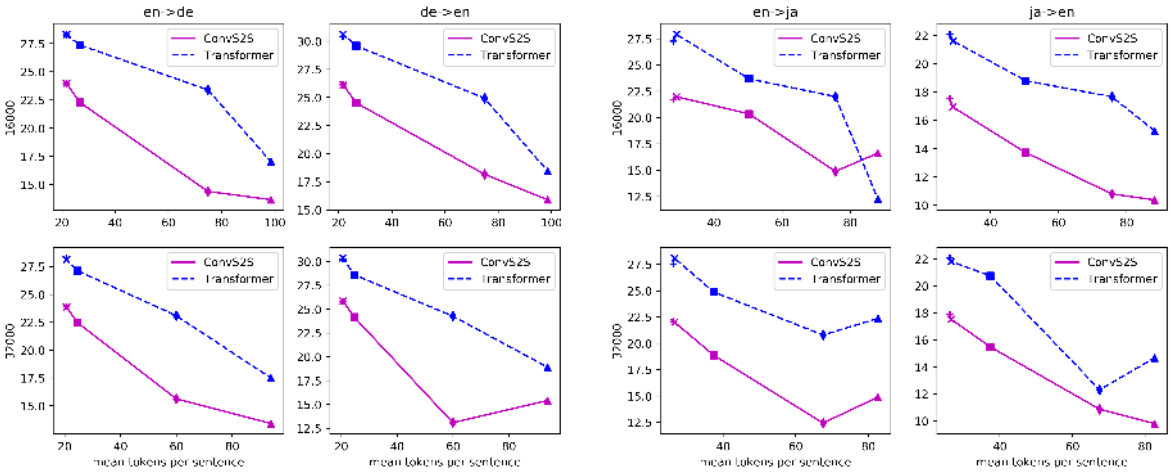

Figure 1: x-axis is the average number of tokens of the training sentences (source and target) and y-axis the BLEU score for 16k and 32k tokens (first and second row respectively). The markers correspond to IRRMGP (+), BPE (×), SEQUITUR (■), RANDOM (♦) and MAXLEN (▲). Each point corresponds to the average over 5 training runs.

use of morphological knowledge. For example, Huck et al. (2017) show that a combined approach can sometimes outperform pure BPE tokenization. Following standard practices in NMT, in our experiments we only performed intra-word tokenization, using the white-space as a starting segmentation beyond which no token could be formed. When this restriction was removed, the resulting sequences were even smaller but BLEU scores decreased. While this might be because the hyperparameters were set to maximise the BLEU score of BPE intra-word, it could be an indication that some linguistic prior (like word tokenization) trumps this specific formalization of Occam's Razor.

Acknowledgements

We thank the anonymous reviewers and meta-reviewer for their insightful remarks and propositions, as well as Laurent Besacier and Alexandre Bérard for fruitful discussions.

References

Timothy Bell, John Cleary, and Ian H Witten. 1990. *Text Compression*. Prentice Hall.

Jon Bentley and Douglas McIlroy. 1999. Data compression using long common strings. In *DCC*, pages 287–295. IEEE.

Florian Benz and Timo Kötzing. 2013. An effective heuristic for the smallest grammar problem. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, pages 487–494. ACM.

Rafael Carrascosa, François Coste, Matthias Gallé, and Gabriel Infante-Lopez. 2011. The smallest grammar problem as constituents choice and minimal grammar parsing. *Algorithms*, 4(4):262–284.

Rafael Carrascosa, François Coste, Matthias Gallé, and Gabriel Infante-Lopez. 2012. Searching for smallest grammars on large sequences and application to dna. *Journal of Discrete Algorithms*, 11:62–72.

Moses Charikar, Eric Lehman, Ding Liu, Rina Panigrahy, Manoj Prabhakaran, April Rasala, Amit Sahai, and abhi shelat. 2005. The smallest grammar problem. *IEEE Transactions on Information Theory*, 51(7):2554–2576.

Colin Cherry, George Foster, Ankur Bapna, Orhan Firat, and Wolfgang Macherey. 2018. Revisiting character-based neural machine translation with capacity and compression. In *EMNLP*.

Carl De Marcken. 1996. Linguistic structure as composition and perturbation. In *Proceedings of the 34th*

annual meeting on Association for Computational Linguistics, pages 335–341. Association for Computational Linguistics.

M. Domingo, M. Garcıa-Martınez, A. Helle, F. Casacuberta, and M. Herranz. 2018. How Much Does Tokenization Affect Neural Machine Translation? *arXiv e-prints*.

Philip Gage. 1994. A new algorithm for data compression. *The C Users Journal*, 12(2).

Travis Gagie, Giovanni Manzini, Gonzalo Navarro, Hiroshi Sakamoto, Yoshimasa Takabatake, et al. 2019. Rpair: Rescaling repair with rsync. *arXiv preprint arXiv:1906.00809*.

Jonas Gehring, Michael Auli, David Grangier, and Yann Dauphin. 2017. A convolutional encoder model for neural machine translation. In *ACL*, pages 123–135. Association for Computational Linguistics.

Dan Gusfield. 1997. *Algorithms on strings, trees, and sequences: computer science and computational biology*. Cambridge university press.

Matthias Huck, Simon Riess, and Alexander Fraser. 2017. Target-side word segmentation strategies for neural machine translation. In *WMT*, pages 56–67, Copenhagen, Denmark. Association for Computational Linguistics.

Miguel Ángel Jiménez-Montaño. 1984. On the syntactic structure of protein sequences and the concept of grammar complexity. *Bulletin of Mathematical Biology*, 46(4):641–659.

Jyrki Katajainen and Timo Raita. 1992. An analysis of the longest match and the greedy heuristics in text encoding. *Journal of the ACM*, 39(2).

J. Kreutzer and A. Sokolov. 2018. Learning to Segment Inputs for NMT Favors Character-Level Processing. In *IWSLT*.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *ACL*.

N Jesper Larsson and Alistair Moffat. 2000. Off-line dictionary-based compression. *Proceedings of the IEEE*, 88(11):1722–1732.

Makoto Morishita, Jun Suzuki, and Masaaki Nagata. 2018. Improving neural machine translation by incorporating hierarchical subword features. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 618–629.

Ryosuke Nakamura, Shunsuke Inenaga, Hideo Bannai, Takashi Funamoto, Masayuki Takeda, and Ayumi Shinohara. 2009. Linear-time text compression by longest-first substitution. *Algorithms*, 2(4):1429–1448.

Gonzalo Navarro and Luís Russo. 2008. Re-pair achieves high-order entropy. In *Data Compression Conference*, page 537.

Graham Neubig. 2011. The Kyoto free translation task. http://www.phontron.com/kftt.

Craig G Nevill-Manning and Ian H Witten. 1997. Compression and explanation using hierarchical grammars. *The Computer Journal*, 40(2,3):103–116.

Mārcis Pinnis, Rihards Krišlauks, Daiga Deksne, and Toms Miks. 2017. Neural machine translation for morphologically rich languages with improved subword units and synthetic data. In *Text, Speech, and Dialogue*, pages 237–245, Cham. Springer International Publishing.

Jorma Rissanen. 1978. Modeling by shortest data description. *Automatica*, 14(5):465–471.

E. Salesky, A. Runge, A. Coda, J. Niehues, and G. Neubig. 2018. Optimizing Segmentation Granularity for Neural Machine Translation. *arXiv e-prints*.

Ernst J Schuegraf and H S Heaps. 1974. A comparison of algorithms for data base compression by use of fragments as language elements. *Information Storage and Retrieval*, 10:309–319.

Mike Schuster and Kaisuke Nakajima. 2012. Japanese and korean voice search. In *ICASSP*, pages 5149–5152.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *ACL*, volume 1, pages 1715–1725.

Payam Siyari, Bistra Dilkina, and Constantine Dovrolis. 2016. Lexis: An optimization framework for discovering the hierarchical structure of sequential data. In *KDD*, pages 1185–1194. ACM.

Matthias Sperber, Graham Neubig, Jan Niehues, and Alex Waibel. 2017. Neural lattice-to-sequence models for uncertain inputs. *arXiv preprint arXiv:1704.00559*.

James A Storer and Thomas G Szymanski. 1982. Data compression via textual substitution. *Journal of the ACM*, 29(4):928–951.

Jinsong Su, Zhixing Tan, Deyi Xiong, Rongrong Ji, Xiaodong Shi, and Yang Liu. 2017. Lattice-based recurrent neural network encoders for neural machine translation. In *AAAI*, pages 3302–3308.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Robert Wagner. 1973. Common phrases and minimum-space text storage. *Communications of the ACM*, 16(3).

J Gerard Wolff. 1975. An algorithm for the segmentation of an artificial language analogue. *British Journal of PsychologyJ*, 66:79–90.

Giulio Zhou. 2018. Morphological zero-shot neural machine translation. Master's thesis, University of Edinburgh.