HFT: High Frequency Tokens for Low-Resource NMT

Edoardo Signoroni and **Pavel Rychlý**
Faculty of Informatics
Masaryk University
Brno, CZ, 602 00
e.signoroni@mail.muni.cz, pary@fi.muni.cz

Abstract

Tokenization has been shown to impact the quality of downstream tasks, such as Neural Machine Translation (NMT), which is susceptible to out-of-vocabulary words and low frequency training data. Current state-of-the-art algorithms have been helpful in addressing the issues of out-of-vocabulary words, bigger vocabulary sizes and token frequency by implementing subword segmentation. We argue, however, that there is still room for improvement, in particular regarding low-frequency tokens in the training data. In this paper, we present "High Frequency Tokenizer", or HFT, a new language-independent subword segmentation algorithm that addresses this issue. We also propose a new metric to measure the frequency coverage of a tokenizer's vocabulary, based on a frequency rank weighted average of the frequency values of its items. We experiment with a diverse set of language corpora, vocabulary sizes, and writing systems and report improvements on both frequency statistics and on the average length of the output. We also observe a positive impact on downstream NMT.

Introduction

Tokenization is a fundamental preprocessing step for NMT and it was shown to impact the quality of the final output (Domingo et al., 2018; Gowda and May, 2020; Sennrich et al., 2016) . It involves splitting a longer text in smaller parts called tokens, separating punctuation from words, with current algorithms also implementing subword segmentation. These methods enable NMT models capable of open-vocabulary translation by encoding rare and unknown words as sequences of subword units. This is even more relevant for languages that produce words by agglutination or compounding. Subword segmentation, while usually not adhering to morphological constraints, mimics these processes by learning the most optimal segmentation from training data, thus generating vocabularies of subword tokens capable of generating new words not seen at training time.

For training of an NMT system, the frequency of tokens in the training data is vital. The more frequent the token, the better its representation. While still performing better than Statistical MT, NMT still shows weakness in translating low-frequency words (Koehn and Knowles, 2017). Therefore it is desired that a vocabulary contains a series of well represented, and so, high-frequency tokens. In this regard, some metrics have been proposed in to determine the best settings for training a NMT system (Gowda and May, 2020): i. Frequency at the 95th percentile $F_{95\%}$; ii. Mean Average Sequence length μ. These metrics can be used as an index of the performance of a tokenization algorithm. In our evaluation, we compare different tokenization algorithms against these metrics. Nonetheless, we argue that $F_{95\%}$ is not optimal to measure the frequency coverage of the vocabulary: due to its punctual nature it does not represent the whole vocabulary. Thus, we propose to use a weighted average of the frequencies. All these metrics will be discussed in Section 2.

Usually the tokenization process involves some normalization, often in the form of lower casing or true casing, to handle the difference in spelling in real data and reduce the low-frequency problem. This is even more relevant for small datasets, in which the tokens are inherently less well represented. We argue that this solution is not optimal, as in some cases retaining explicit information regarding uppercase, lowercase, white spaces, and caps lock text can be useful for downstream tasks.

To address these issues, we present "High Frequency Tokenizer", or HFT, a new language-independent subword tokenization algorithm aimed at improving the frequency of the tokens in the vocabulary.

Thus our contributions are the following:

- **High Frequency Tokenizer**, or **HFT**, a new

Proceedings of the 29th International Conference on Computational Linguistic, pages 56–63
October 12–17, 2022.

language-independent subword segmentation algorithm to improve the frequency coverage of tokens;

- a **new metric** to evaluate the performance of tokenizers in this regard, which improves on the Frequency at the 95th percentile proposed by (Gowda and May, 2020)

This paper is structured as follows: Section 1 details the HFT segmentation algorithm; Section 2 relates our evaluation, its experimental setup and results, Section 3 relates to some limitations to be addressed in the future; Section 4 gives an overview of some related work; Section 5 outlines our conclusions.

1 High Frequency Tokenizer

HFT uses the advantage of pretokenization, where sentences are split into tokens on the borders of alphanumeric and non-alphanumeric characters. The current prototype uses the regular expression `\b` of the Unix `sed`[1] command . Both the beginning and the end of each token is explicitly annotated.

HFT subwords are learnt from these tokens, they never cross the token boundaries, each token from the pretokenization is handled independently from other tokens. It speeds up both vocabulary learning and actual subword tokenization.

We also use case normalization for characters with both uppercase and lowercase. A single uppercase letter is changed to a special <uppercase-next> character and lowercase version of the given letter. A sequence of uppercase letters is changed to lowercase with a special <all-uppercase> and <end-of-uppercase> characters attached to the beginning and the end of the sequence. Figure 2 gives the special characters `hft` uses in pretokenization and tokenization.

The learning algorithm starts from a vocabulary containing all characters from the training text as possible subwords. The vocabulary contains the number of occurrences of the given subword (character). Then it gradually increase the vocabulary in the following steps:

1. it processes all the words (tokens) from the pretokenized text to find the best subword segmentation using only subwords from the current vocabulary, counts the frequencies of each subword and of all possible subword candidates (pairs of succeeding subwords);
2. selects the top K candidates with the highest frequency and adds them as new subwords to the vocabulary (K is 5% of the target vocabulary size as default);
3. removes from the vocabulary all non-single-character subwords with frequency lower than the last added candidate;
4. repeat from 1. until the requested vocabulary size is reached

The best subword tokenization (in step 1) searches in all possible subword segmentation sequences the one with the lowest number of tokens and (for same number of tokens) the highest minimum frequency.

2 Evaluation

We train and compare `hft` with the `sentencepiece` (Kudo and Richardson, 2018) implementation of `bpe` (Sennrich et al., 2016) and `unigram` (Kudo, 2018) on two of the metrics presented by Gowda and May (2020). We use portions of different bilingual and monolingual datasets: The English section of the English-Marathi and the Irish part of the English-Irish from the LoResMT 2021 shared task[2]; a sample of the Hindi half of the English-Hindi IITB parallel corpus (Kunchukuttan et al., 2018) and of the Lithuanian portion of the Lithuanian-English of Europarl (Koehn, 2005) used in the WMT19 News Translation Shared Task[3]. As for monolingual corpora, we used different translations of the Bible[4] (Christodoulopoulos and Steedman, 2014) in a diverse range of languages and writing systems retrieved from OPUS (Tiedemann and Nygaard, 2004). Figure 3 gives an overview of the size of the datasets.

2.1 Experimental Setup

Following Gowda and May (2020), we evaluate our tokenizer on two statistics: **Frequency at 95% Class Rank**($F_{95\%}$), defined as the least frequency in the 95^{th} percentile of most frequent tokens, and **Mean Sequence Length**(μ), which is computed as the arithmetic mean of the lengths of the tokenized sequences. We also propose and test a new metric, **Frequency Rank Weighted Average** ν, to improve on the intuition of $F_{95\%}$.

[1] https://www.gnu.org/software/sed/manual/sed.html

[2] https://github.com/loresmt/loresmt-2021

[3] https://www.statmt.org/wmt19/translation-task.html

[4] We are aware of the shortcomings of the Bible as a NLP dataset

↑¦state¦ ↑¦health¦ ↑¦minister¦ ↑¦rajesh¦ ↑¦tope¦ ¦also¦ ¦acknowledged¦ ¦the¦ ¦situation¦ ,_ ¦saying¦ ,_" ↑¦efforts¦
↑¦currently¦ ,_ ¦there¦ ¦are¦ ¦2,56,278¦ ¦isolation¦ ¦beds¦ _(¦excluding¦ ∆¦icu¦∇)_ ¦available¦ ¦in¦ ¦the¦ ¦state¦

Figure 1: A sample of pretokenized text from the English dataset.

¦	<token-delimiter>
↑	<single-uppercase>
_	<explicit-whitespace>
∇	<all-uppercase>
∆	<end-of-uppercase>

Figure 2: Special characters in the pretokenization and tokenization.

$F_{P\%}$ is a way to quantify the minimum number of training examples for at least the *P*th percentile of tokens, while the bottom (1-*P*) is discarded to account for noise inherent in real-world data. An higher value of $F_{95\%}$ reflects the presence of many training examples per token, and thus is the desired setting for ML methods.

We argue that this metric is not optimal to capture the frequency coverage of a tokenizer's vocabulary, since it considers just one value, and not the whole structure of the vocabulary. Instead, we propose a **Frequency Rank Weighted Average** ν. Assuming a vocabulary ranked according to descending frequencies, we compute ν as:

$$\nu = \frac{\sum_{i=1}^{n}(i \cdot f_{x_i})}{\sum_{i=1}^{n} i} \quad (1)$$

where f_{x_i} is the frequency of the token x at the vocabulary index i, and n is the length of the vocabulary. We improved on the intuition of $F_{95\%}$, which purpose is to assure good token coverage even at lower frequency ranks. Following this objective, our metric gives more weight to lower frequency tokens in the vocabulary, all the while considering all of its length.

Gowda and May (2020) cast NMT as a classification task in an autoregressive setting, where the total error accumulated grows proportionally with the length of the sequence, altering the prediction of subsequent tokens in the sentence. Thus, a smaller sequence length is preferred.

We compare `hft` with `bpe` and `unigram`, the latter two being trained with the `sentence piece` module. We train models separately for each language and for different vocabulary sizes. Following previous work (Gowda and May, 2020; Sennrich et al., 2016; Sennrich and Zhang, 2019), we limit our investigation between vocabulary sizes of 500 and 8k tokens, since it was shown that bigger vocabulary sizes for small datasets harm the quality of the translation. Other parameters for the `sentence piece` trainer are left in the default setting.

We compute $F_{95\%}$, μ, and ν on the same train portion of the data, since these metrics do not involve any downstream task or validation on external data.

Figure 4 gives a sample of the results of our experiments regarding the metrics mentioned above.

We also report on some preliminary evaluation of the impact of HFT against BPE on downstream NMT. We train BPE and HFT tokenizers for both source and target language separately and then we tokenize the data with BPE (Sennrich et al., 2016), as implemented in `subword-nmt`[5], and HFT, with our implementation. We used a vocabulary size of 2000 for *en-ga* and 3000 for *en-mr*. The size of the vocabulary is set at the same value for both tokenization methods for the same dataset. For this evaluation, we do not optimize any other hyperparameter nor we employ techniques such as backtranslation.

2.2 Results

The following sections detail our results: Section 2.3 relates to the metrics explained in Section 2.1, while Sections 2.4 and 2.5 report some preliminary results on downstream NMT.

2.3 Metrics

`hft`'s performance on both $F_{95\%}$ and the Average Length μ seems promising, improving on both `bpe` and `unigram` in most of the cases. Recall that according to Gowda and May (2020) a higher value of $F_{95\%}$ and a lower value of μ is the desired outcome. For each vocabulary size, a higher value of ν means a better frequency coverage.

In the case of $F_{95\%}$, it starts at lower values, and then picks up the pace after some vocabulary size threshold. From our qualitative evaluation of the models, we deduce that this is due to the choice of storing every character occurring in the data at least

[5]https://github.com/rsennrich/subword-nmt

Language		Dataset		Sent.	Script	Sample
Amharic	Afro-Asiatic	am	Bible	30.580	Ge'ez	በመጀመሪያ እግዚአብሔር ሰማይንና
Arabic	Afro-Asiatic	ar	Bible	31.102	Arabic	في البدء خلق الله السموات والارض
Cherokee	Iroquian	chr	Bible-NT	7.957	Cherokee	ᎯᎠ ᎪᏪᎵ ᎧᏃᎮᏛ ᏧᏁᏢᏔᏅᏒ ᏥᏌ ᎦᎶᏁᏛ,
Czech	Indo-Eur.	cs	Bible	38.116	Latin(Czech)	Na počátku stvořil Bůh
English	Indo-Eur.	en	LoResMT	20.933	Latin	It is also doubtful whether
Finnish	Ugro-Finnic	fi	Bible	38.613	Latin	Alussa loi Jumala taivaan
Irish	Indo-Eur.	ga	LoResMT	8.112	Latin	Cén chaoi a n-oibríonn
Hindi	Indo-Eur.	hi	IITB	20.000	Devanagari	यह कार्य दोनों प्रकार के हैं - ऑनलाइन और बाहरी।
Italian	Indo-Eur.	it	Bible	38.536	Latin	In principio Dio creò il
Japanese	Japonic	ja	Bible	31.087	Kana/Kanji	はじめに神は天と地とを創造された。
Jakaltek	Mayan	jak	Bible-NT	12.509	Latin(Jak.)	Ha' icham Abraham yeb naj
Lithuanian	Indo-Eur.	lt	Europarl	20.000	Latin(Lith.)	Tačiau balsavau prieš prane
Marathi	Indo-Eur.	mr	LoResMT	20.933	Devanagari	राज्यात हळूहळू अनलॉक होण्यास सुरूवात झाली
Burmese	Sino-Tibetan	my	Bible	30.928	Burmese	အစအဦး၌ ဘုရားသခင်သည် ကောင်းကင်နှင့်
Ojibwe	Algic	ojb	Bible-NT	7.945	Ojibwe	ᒦᑕᐦ ᐅᐅᐌ ᐌᔑᐲᐃᑳᑌᓂᒃ ᐋᓃᑎ ᒋᓭᐠ x ᑳᐱᐅᒋ ᓂᑖᐎᑭᒻ
Swedish	Indo-Eur.	sv	Bible	38.879	Latin	I begynnelsen skapade Gud
Syriac	Afro-Asiatic	syr	Bible-NT	7.954	Syriac	ܕܝܫܘܥ ܡܫܝܚܐ ܒܪܗ ܕܕܘܝܕ ܒܪܗ ܕܐܒܪܗܡ
isiZulu	Niger-Congo	zu	Bible-NT	9.095	Latin(Zulu)	Incwadi yokuzalwa kukaJesu

Figure 3: Overview of the datasets. From left to right, the table gives: the name of the language and its family, the name of the dataset, its size, the name of the writing system used, and a sample of the text.

once. This is done to prevent out-of-vocabulary tokens, similarly to both `bpe` and `unigram`, but leads to a bigger portion of smaller vocabularies being made up of characters. This is particularly evident in the Japanese dataset, which contains a larger amount of ideograms. This issue will be addressed in future research.

Regarding the Average Length μ of the segmented output, we find bigger improvements on some of the datasets, such as Jakaltek, isiZulu, and Lithuanian. In other cases, the performance increase is smaller, depending on the dataset. Conversely, we observe a significant increase of μ for other languages, such as Burmese and Japanese. The reason of this behavior is worthy of further investigation.

When looking at the Frequency Rank Weighted Average ν, `hft` outperforms `bpe` slightly and `unigram` by a bigger margin on each vocabulary size. However, a more in-depth analysis is need for specific datasets, such as Japanese, which are more problematic than others.

Taking a look at the frequency distribution of a vocabulary's elements, plotted in Figure 5, it is noticeable that our algorithm trades off frequency values between the most frequent elements, which are better represented in `bpe` and `unigram`, and the tokens with lower frequency, which frequency counts in `hft` are higher. In fact, we can see that `hft` consistently has higher values for the bottom part of the vocabulary. We argue that this is in fact a very good trade: while the higher ranking tokens are still very well represented, we also achieve better frequencies and representations on the lower occurring tokens.

`hft` often achieves better performance than other methods with regards to $F_{95\%}$ and μ. Nevertheless, our segmentation algorithm is not free from issues. We will discuss these in section 3.

2.4 Downstream NMT

We also report on a preliminary evaluation of the impact of `hft` against `bpe` on downstream NMT. We train `bpe` and `hft` tokenizers for both source and target language separately and then we tokenize the data with `bpe` (Sennrich et al., 2016), as implemented in `subword-nmt`[6], and `hft`, with our implementation. We used a vocabulary size of 2000 for *en-ga* and 3000 for *en-mr*. The size of the vocabulary is set at the same value for both tokenization methods for the same dataset. For this evaluation, we do not optimize any other hyperparameter nor we employ techniques such as backtranslation.

[6]https://github.com/rsennrich/subword-nmt

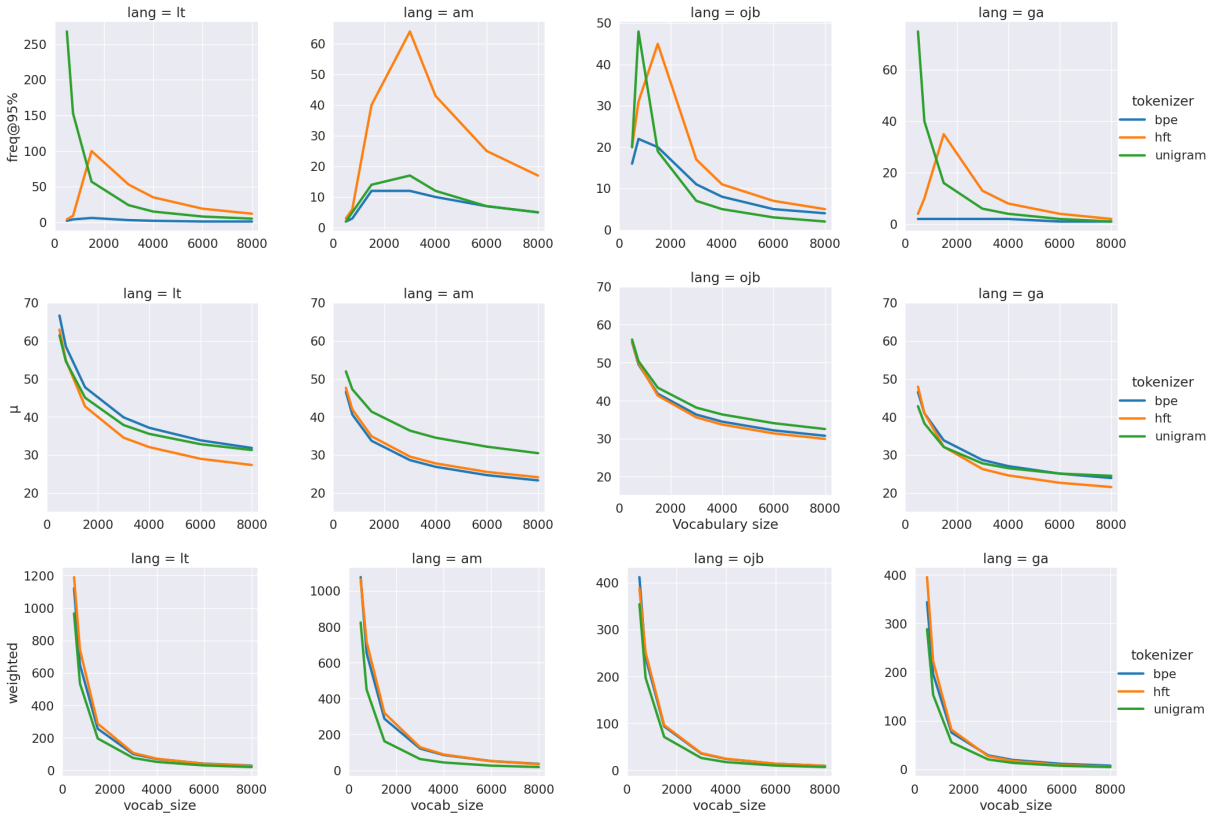

Figure 4: $F_{95\%}$ (higher is better), μ (lower is better), and ν weighted average (higher is better) plotted against vocabulary size on the Lithuanian, Amharic, Ojibwe, and Irish datasets.

We use Fairseq (Ott et al., 2019) to train 5 default Transformers (Vaswani et al., 2017) for both directions and tokenizer type for 30 epochs with dropout of 0.1, label smoothing of 0.1, and 4096 maximum tokens for each training batch. We use *adam* as optimizer, a learning rate of 0.0005 and the inverted square root scheduler.

We optimize for BLEU during training on the validation set at each epoch, with detokenized text for `bpe` (obtained with the Fairseq `-remove-bpe` argument) and tokenized text for `hft`, since we currently do not have a custom Fairseq plugin to allow detokenized training on `hft`. These preliminary results are summarized in Table 1.

Training the Transformer on data tokenized with `hft` leads to a better average NMT performance on both datasets we experimented on, with an increment in BLEU from +0.82 to +2.15. The overall low BLEU scores can be explained by the fact that we did not optimized neither the architecture nor the parameters of the Transformer to the specific low-resource dataset.

2.5 Qualitative Evaluation

We conduct some preliminary analysis of the translation systems' output. To obtain our candidates for manual evaluation, we compute sentence-level sacreBLEU score on the output of both `bpe`- and `hft`-based systems, against a reference translation. We then compute the difference in BLEU score between the two different outputs, and list them by decreasing size of the gap, that is the most changed first. This is done to observe where `hft` has the biggest impact. We consider the first 50 candidates for both *en-ga* and *en-mr* parallel corpora. Due to linguistic constraints, however, we are able to manually analyze only the *ga-en* and the *mr-en*.

While more in-depth examination is warranted in this regard, we can already see that `hft` provides some benefits, such as the one shown in Figure 6. In this case, the named entity *Naxals*[7] was correctly generated by the `hft`-based model, while the bpe-based one gives an almost nonsensical translation.

[7] A group of Maoist communists currently leading an insurgency against the Indian Government in the so-called "Red corridor" area of east and central India.

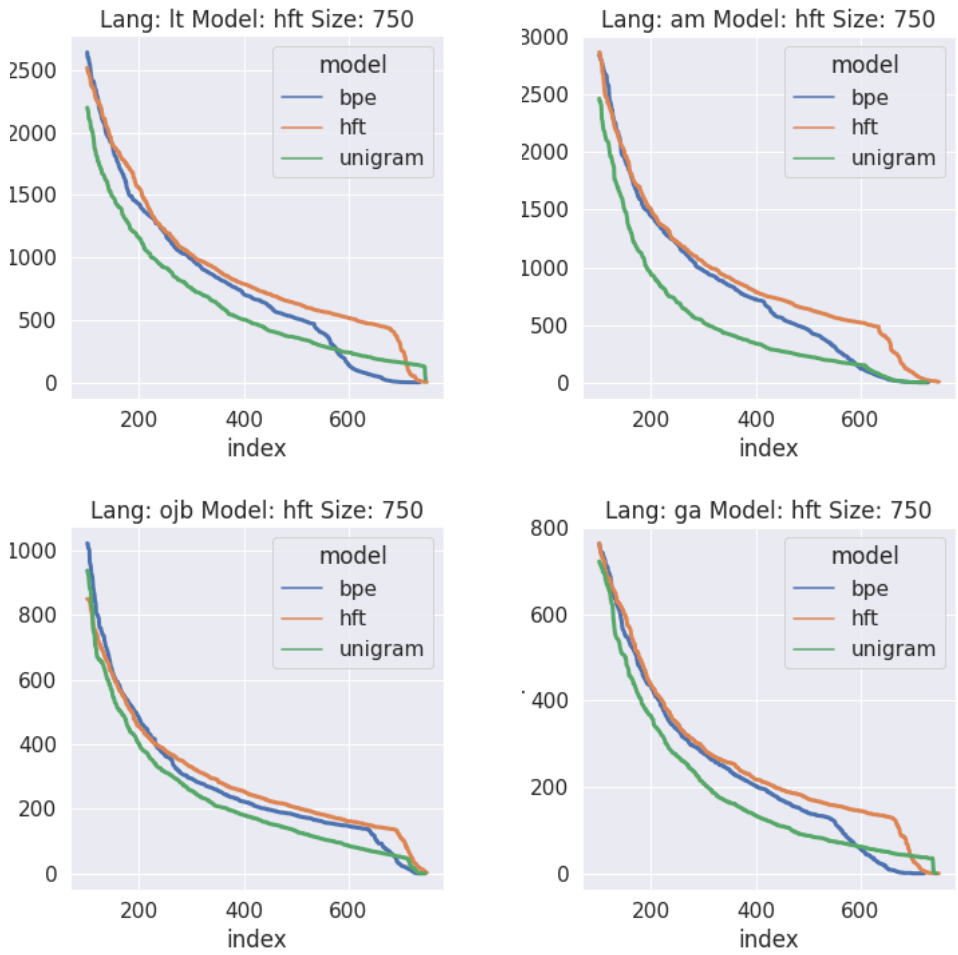

Figure 5: Frequency distribution of tokens in a sample of the 0.75k vocabularies. We do not plot the first 100 most frequent tokens to obtain cleaner plots and to focus on the bottom of the vocabulary.

3 Limitations and Future Work

As we mentioned in section 2.2, we opted to include in the final vocabulary each character seen in the training data at least once to avoid out-of-vocabulary tokens. This, however, has the side-effect of saturating and inflating the vocabulary with low-frequency tokens. At lower vocabulary sizes, these entries make up a bigger percentage of the overall vocabulary, becoming less and less relevant as the size increases. This explains why our method becomes effective over a threshold in the size of the vocabulary, which varies depending on the size and, more importantly, on the amount of unique characters in the dataset.

The presence of these character may be due to the inherent complexity of the writing system. This is the case of Japanese, which uses two sets of syllabic characters, *hiragana* and *katakana* (collectively referred to as *kana*), and a huge amount of Chinese-derived ideograms, called *kanji*.

The other source for unique characters in the data is noise, in the form of non-standard orthography, special characters, non-linguistic text, and so on. If present in the dataset, these sections can quickly saturate our vocabulary. This issue makes the method somewhat susceptible to noise, and must be addressed in future work.

Another aspect to investigate further is the relationship between μ and different writing systems. From our evaluation, we have seen that `hft` improves the performance on this metric for most of the data and writing systems we included in our evaluation. However, for Japanese and Burmese, μ is higher than other methods. It is worthwhile to investigate this matter in the future.

While the preliminary evaluation of `hft`'s impact on downstream NMT seems to show promising results, we acknowledge that the testing sample is not vast. Moreover, using an unoptimized Transformer does not completely reflect real-world appli-

DATASET	MODEL	BLEU						INCREMENT
		1	2	3	4	5	avg	
en-ga	t-bpe	4.46	4.54	4.06	4.69	4.73	4.50	
	t-hft	**5.34**	**5.49**	**5.95**	**5.69**	**5.59**	**5.61**	**+1.11**
ga-en	t-bpe	5.57	5.48	5.12	5.80	5.51	5.50	
	t-hft	**6.09**	**6.49**	**6.57**	**6.10**	**6.33**	**6.32**	**+0.82**
en-mr	t-bpe	7.49	7.21	6.88	6.57	6.12	6.85	
	t-hft	**7.33**	**7.99**	**8.80**	**8.31**	**8.31**	**8.14**	**+1.29**
mr-en	t-bpe	9.58	8.56	10.15	8.58	9.56	9.29	
	t-hft	**11.05**	**12.09**	**12.19**	**11.06**	**10.82**	**11.44**	**+2.15**

Table 1: Results of the evaluation on NMT given in sacreBLEU scores, for each dataset and trained model. The last column reports the increment of hft models over the `bpe` baseline.

cations, where the NMT system would be carefully tuned to the specific dataset. We plan to undertake a more comprehensive evaluation on downstream translation in the future, by enlarging the testing sample and employing `hft` in settings closer to real applications.

Lastly, we report that `hft` has longer training time than other algorithms in the current implementation, which are however still in the range of minutes for the bigger datasets we used. We plan to work on this shortcoming in the next implementation of the tokenizer.

4 Related Work

Before Sennrich et al. (2016), MT coped with the problem of out-of-vocabulary words by backing off to a dictionary with sub-optimal assumptions regarding morphological identities and transliterations. `bpe` addressed this issue by adapting a compression algorithm to the task of word segmentation. The methods initializes the symbol vocabulary with the character vocabulary, plus a special end-of-word symbol. Then it iteratively counts all symbols pairs and replaces every occurrence of the most frequent pair ('A', 'B') with the new symbol 'AB'. These character *n*-grams are then merged together in a similar fashion. They do not consider pairs that cross word boundaries. Following these steps, `bpe` allows for open-vocabulary NMT, which better handles out-of-vocabulary and rare words, by representing them as a sequence of subword units.

The `unigram` method by Kudo (2018) is based on a unigram language model. This makes the assumption that each subword occurs independently, thus formulating the probability of a subword sequence $X = (x_1, ..., x_M)$ as the product of the subword occurrence probabilities $p(x_i)$. To find the vocabulary set and their probabilities, they employ an iterative algorithm which starts by creating a seed vocabulary of unique characters and most frequent substrings, without considering those that cross word boundaries. Then the following steps are repeated until the vocabulary reaches the desired size: i. fixing the set of vocabulary, optimize $p(x)$ with the Expectation Maximization (EM) algorithm; ii. compute the $loss_i$, for each subword x_i, as the amount the likelihood L is reduced when removing x_i from the vocabulary; iii. sort the symbols by loss, and keeping the top $n\%$ of subwords, while always keeping single characters to avoid out-of-vocabulary. Thus `unigram` can output multiple segmentations and their probabilities, making it more flexible than `bpe`.

In `sentence piece` (Kudo and Richardson, 2018) both these segmentation algorithms are implemented in a way that removes the need for preprocessing steps, such as pretokenization, and trains subword models directly from the raw sentences. This allows for the creation for a purely end-to-end and language-independent system.

5 Conclusions

In this paper we present **High Frequency Tokenizer**, or HFT, a new language-independent subword tokenization algorithm to improve on the frequency coverage of tokens in the vocabulary of NMT systems. We demonstrate its performance on a diverse dataset of languages and writing systems, and show that our approach can be beneficial to downstream NMT.

However, some issues still remain to be investigated, such as the frequency coverage for smaller vocabularies and the mean output length for some

```
ref: Hundreds of Naxals have been infected with corona throughout the penance.

bpe: Help teachers to have died the corona infection.

hft: Hundreds of Naxals have been infected with corona.
```

Figure 6: Example from the *mr-en* translation systems. The first line gives the reference translation, the second gives the translation from a `bpe`-based system, while the last gives the translation from an `hft`-based system. The named entity *Naxals* is preserved by `hft`.

languages. This will be the matter for future research. We also plan to further evaluate `hft`'s impact on downstream NMT.

The `hft` scripts are available on GitHub,[8] together with the evaluation's data and results.[9]

Acknowledgements

We thank the reviewer for their useful inputs. This work was partly supported by the Internal Grant Agency of Masaryk University, Lexical Computing, and the Ministry of Education of the Czech Republic within the LINDAT-CLARIAH-CZ project LM2018101.

References

Christos Christodoulopoulos and Mark Steedman. 2014. A massively parallel corpus: the bible in 100 languages. *Language Resources and Evaluation*, 49:1–21.

Miguel Domingo, Mercedes Garcıa-Martınez, Alexandre Helle, Francisco Casacuberta, and Manuel Herranz. 2018. How much does tokenization affect neural machine translation?

Thamme Gowda and Jonathan May. 2020. Finding the optimal vocabulary size for neural machine translation. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3955–3964, Online. Association for Computational Linguistics.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of Machine Translation Summit X: Papers*, pages 79–86, Phuket, Thailand.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39, Vancouver. Association for Computational Linguistics.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Melbourne, Australia. Association for Computational Linguistics.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Anoop Kunchukuttan, Pratik Mehta, and Pushpak Bhattacharyya. 2018. The IIT Bombay English-Hindi parallel corpus. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Rico Sennrich and Biao Zhang. 2019. Revisiting low-resource neural machine translation: A case study. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 211–221, Florence, Italy. Association for Computational Linguistics.

Jörg Tiedemann and Lars Nygaard. 2004. The OPUS corpus - parallel and free: `http://logos.uio.no/opus`. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal. European Language Resources Association (ELRA).

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

[8] https://github.com/pary42/hftoks

[9] https://github.com/edoardosignoroni/hftoks-eval