What changes when you randomly choose BPE merge operations? Not much.

Jonne Sälevä and **Constantine Lignos**
Michtom School of Computer Science
Brandeis University
`jonnesaleva,lignos@brandeis.edu`

Abstract

We introduce two simple randomized variants of byte pair encoding (BPE) and explore whether randomizing the selection of merge operations substantially affects a downstream machine translation task. We focus on translation into morphologically rich languages, hypothesizing that this task may show sensitivity to the method of choosing subwords. Analysis using a Bayesian linear model indicates that one variant performs nearly indistinguishably compared to standard BPE while the other degrades performance less than we anticipated. We conclude that although standard BPE is widely used, there exists an interesting universe of potential variations on it worth investigating. Our code is available at: `https://github.com/bltlab/random-bpe`.

1 Introduction and related work

Most neural machine translation (NMT) models assume their inputs to be sequences of units drawn from a fixed vocabulary. While these units were tokens in the early years of NMT (Cho et al., 2014; Sutskever et al., 2014), there has since been a transition to *subword*-level models that learn a vocabulary of "word pieces" which serve as an intermediate representation between words and characters (Mielke et al., 2021). Such representations are attractive because they solve the closed-vocabulary problem of early, word-level NMT (Luong et al., 2015) while also yielding more semantically meaningful units than individual characters.

Well-known subword segmentation algorithms include byte pair encoding (BPE) (Sennrich et al., 2016), SentencePiece Unigram LM (Kudo and Richardson, 2018; Kudo, 2018) and the WordPiece algorithm (Wu et al., 2016; Song et al., 2021). All of them include a hyperparameter that controls the size of the subword vocabulary: SentencePiece and WordPiece do this explicitly with a vocabulary size parameter, whereas BPE specifies the number of *merge operations* which implicitly define the subword vocabulary.

Prior work has addressed the problem of optimally selecting the vocabulary size. Haddow et al. (2018) and Sennrich and Zhang (2019) find that using too large a subword vocabulary can result in low-frequency tokens being represented as atomic units, which makes it difficult to learn proper representations for them. Gowda and May (2020) suggest a heuristic: use as many subwords as possible provided that at least 95% of the subwords have 100 or more examples in the training set. Gutierrez-Vasques et al. (2021) find that around 350 merge operations are enough to generate similar subword distributions across languages.

Subword segmentation algorithms usually build their subword vocabularies by optimizing an objective function that is independent of the downstream task. For instance, SentencePiece employs the probabilities under its unigram language model, while BPE aims to maximize the degree of sequence compression by greedily selecting and merging the symbol pairs that occur most frequently. Others have re-framed this process as finding an "optimal" set of units that maximize more sophisticated probabilistic criteria. Vilar and Federico (2021) introduce an extension of BPE that learns a subword vocabulary by maximizing a likelihood objective over potential subwords. He et al. (2020) introduce a method that treats the segmentation as a latent variable to be marginalized out and seek to find segmentations that maximize the downstream task probability directly.

In this paper, we build upon the concept of stochastic segmentation and conduct neural machine translation experiments on four languages (German, Finnish, Estonian and Uzbek) of varying morphological complexity, using variants of BPE that randomly sample merge operations instead of deterministically choosing the most frequent one.

Our negative result challenges our initial beliefs

that standard BPE would produce the most effective subword representations for translation and that the success of BPE was due to the greedy selection process for learning merge operations. We find that even when merge operations are randomly sampled uniformly, the performance degradation is less than we anticipated. We conclude by discussing how this finding relates to the overall role of subwords in NMT.

2 Byte pair encoding and randomization

We briefly review the BPE training algorithm and introduce our randomized variants. The pseudocode for the algorithm we use can be seen in Algorithm 1. Our presentation is adapted from the BPE algorithm in Vilar and Federico (2021).

Algorithm 1: BPE training algorithm.

Input: D: Training corpus. M: Number of merge operations to learn.
Output: R: list of learned merges.

```
1  def trainBPE(D, M, method):
2      R ← []
3      while |R| ≤ M do
4          C ← countSymbolPairs(D)
5          (x, y) ← choosePair(C, method)
6          rule ← ⟨(x, y) → xy⟩
7          R ← append(R, rule)
8          D ← applyRule(D, rule)
9      return R
10 def choosePair(counts, method):
11     if method = standard then
12         pair ← arg max_{pair∈counts} counts[pair]
13     else if method = uniform then
14         probs ∝ 1
15         pair ← sample(counts, probs)
16     else
17         probs ← softmax(counts)
18         pair ← sample(counts, probs)
19     return pair
```

2.1 Standard BPE algorithm

The standard byte pair encoding algorithm (Sennrich et al., 2016) is a greedy algorithm that takes as input a corpus D—typically the training set or another large collection of text—as well an integer M that specifies the number of merge operations to learn. After first segmenting D into space-separated characters, the algorithm counts how many times each pair of symbols occurs in D (countSymbolPairs). Based on the counts, the algorithm finds the most frequent symbol pair (choosePair) and learns a new merge operation that merges the constituent symbols into a new

symbol. After learning the merge operation, the algorithm replaces all occurrences of the symbol pair in D with the new merged symbol (applyRule).

While the initial merge operations merge individual characters, during the later iterations larger chunks of words are merged together as well. For example, if the most frequent symbol pair was (ab,c), the algorithm would learn the rule ab c → abc which replaces all occurrences of ab c with abc, taking care to not cross word boundaries.

This is repeated for M iterations until a desired number of merge operations is learned, after which the algorithm returns the list of merge operations as output. At test time, the algorithm splits incoming lines of text into individual characters and then applies each of the learned merge operations in order, resulting in text where each space-separated token is an individual subword.

2.2 Randomized BPE variants

To extend BPE to randomized variants, we replace the step of picking the most frequent symbol pair at each iteration with random sampling.

Softmax sampling In our first variant, we assign each symbol pair a probability of being sampled based on how often it occurs in the observed data. We apply a softmax to the observed symbol pair occurrence counts and draw a random symbol pair to merge according to a categorical distribution with the softmax probabilities as its parameters.

Uniform sampling As our second variant, we select each merge operation with uniform probability from the set of observed symbol pairs. Since every symbol pair has equal probability of being sampled, the frequency of each symbol pair is not used in sampling.

3 Experimental setup

Task and data We experiment with translation from English to several morphologically rich languages: Finnish, Estonian, German, and Uzbek. Statistics for each dataset can be found in the Appendix. For all languages except Uzbek, we use the WMT shared task data from He et al. (2020). For Uzbek, we use the Turkic Interlingua corpus (Mirzakhalov et al., 2021).

Tokenization and subword segmentation All of our datasets had previously been tokenized. We performed BPE segmentation on those tokens at the character level using subword-nmt, which we

		BLEU			chrF		
Language	Merges	Standard	Random	Uniform	Standard	Random	Uniform
Estonian	2,000	18.08 ± 0.07	18.17 ± 0.06	17.48 ± 0.04	51.01 ± 0.09	51.10 ± 0.09	50.38 ± 0.07
	5,000	17.98 ± 0.11	17.89 ± 0.09	17.43 ± 0.06	50.80 ± 0.14	50.65 ± 0.11	50.48 ± 0.06
	32,000	16.13 ± 0.06	16.13 ± 0.10	16.86 ± 0.06	48.70 ± 0.09	48.65 ± 0.05	50.21 ± 0.09
Finnish	2,000	16.40 ± 0.08	16.20 ± 0.04	15.26 ± 0.14	50.99 ± 0.07	50.90 ± 0.06	49.76 ± 0.12
	5,000	15.77 ± 0.08	16.01 ± 0.04	14.63 ± 0.09	50.64 ± 0.07	50.67 ± 0.06	49.32 ± 0.09
	32,000	13.83 ± 0.09	13.88 ± 0.09	13.28 ± 0.11	48.20 ± 0.11	48.20 ± 0.07	47.92 ± 0.08
German	2,000	24.56 ± 0.05	24.46 ± 0.06	22.54 ± 0.08	55.77 ± 0.03	55.74 ± 0.03	53.41 ± 0.08
	5,000	24.84 ± 0.07	24.79 ± 0.10	22.73 ± 0.04	56.12 ± 0.04	55.98 ± 0.04	53.65 ± 0.05
	32,000	25.49 ± 0.06	25.33 ± 0.05	22.91 ± 0.07	56.60 ± 0.03	56.54 ± 0.04	54.26 ± 0.05
Uzbek	2,000	47.31 ± 0.21	45.82 ± 1.14	37.85 ± 0.24	64.51 ± 0.18	63.24 ± 1.00	57.66 ± 0.23
	5,000	46.77 ± 1.10	45.39 ± 1.46	38.79 ± 0.20	63.78 ± 0.92	62.52 ± 1.31	58.39 ± 0.15
	32,000	48.63 ± 0.75	47.98 ± 0.70	41.73 ± 0.51	64.76 ± 0.56	64.24 ± 0.59	60.43 ± 0.42

Table 1: Mean and standard error of BLEU and chrF scores across target languages, merge operations and BPE segmentation types. All numbers computed over 10 replications with different random seeds.

modified to support randomized subword sampling. All subword vocabularies are learned separately for each language. As the number of merge operations M is a hyperparameter, we experiment with the values 2,000, 5,000, and 32,000. The largest value, 32,000, is taken directly from He et al. (2020); the smaller values of 2,000 and 5,000 are motivated the observation that higher numbers of merges tend to lead to a near-word-level segmentations for which learning good representations may not be feasible (Sennrich and Zhang, 2019).

Model and training   Our model is a standard Transformer-based encoder-decoder model, as implemented in the `fairseq` library. Our architecture is similar to `transformer-base`, with 512-dimensional embeddings on both the encoder and decoder side, 2048-dimensional feedforward layers, and 6 stacked Transformer layers with 8 attention heads each in both the encoder and decoder. We train all our models for 10,000 updates using a learning rate of 0.005 and the largest feasible batch size (36K tokens per batch for Finnish and Estonian, 30K tokens per batch for German, and 12K tokens per batch for Uzbek). Each translation experiment is run on a single NVIDIA V100 GPU (24GB). We simulate training on multiple GPUs by accumulating gradients for 16 backward passes before each parameter update. To estimate the variability of our results across random seeds, we perform 10 replications of each experiment.

Evaluation   We evaluate all of our models with the `sacrebleu` library (Post, 2018) using BLEU[1]

(Papineni et al., 2002) as well as chrF[2] (Popović, 2015) as it is a tokenization-free metric. Both metrics are computed using the default parameters. We use the `sacremoses`[3] detokenizer to create the detokenized versions of our corpora.

4   Results

Our main experimental results are displayed in Table 1 and Figure 1. For most languages translation performance appears to be rather stable across seeds, but in Uzbek standard errors are larger than other languages and they seem to increase with increasing numbers of merge operations. We believe this noisiness is due to the smaller size of the Uzbek dataset rather than any language-specific phenomena.

Initially, we would have expected that standard BPE would perform the best out of all methods and that different BPE variants would produce noticeable performance differences for all languages. Somewhat contrary to our hypothesis, we find that using randomized BPE variants seems to have quite a small average effect with significant variation in the effect size from language to language. Uniform segmentation tends to consistently perform worse than standard BPE and softmax-based sampling, which can be explained by the roughly 3x longer sequences the model produces. Looking across merge operations, the BLEU/chrF differences between the best and worst BPE variants seem to be less than 1.0 and 1.5 points for Estonian

[1]Version string: `nrefs:1|case:mixed|eff:no| tok:13a|smooth:exp|version:2.3.1`

[2]Version string: `nrefs:1|case:mixed|eff:yes| nc:6|nw:0|space:no|version:2.3.1`

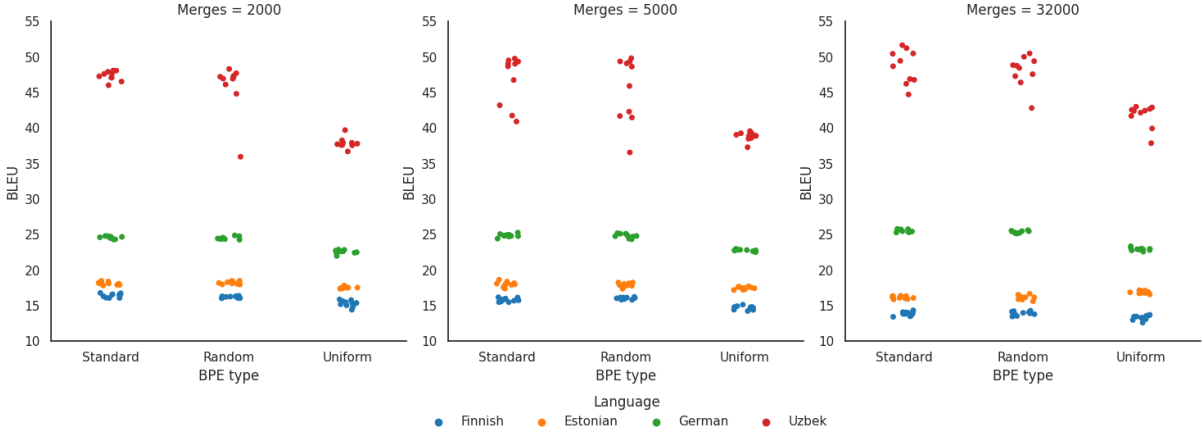[3]https://github.com/alvations/ sacremoses

Figure 1: Translation performance (BLEU) across languages and merge operations. A figure showing chrF is provided in the Appendix.

and Finnish, respectively. However, German and Uzbek show a different picture, with BLEU/chrF differences of around 2-2.5 points for German and 4-9 points for Uzbek.

The impact of varying the number of merge operations varies and again bifurcates the set of languages. Finnish and Estonian seem to suffer slightly as the merge operations are increased (approx. -2 to -2.5 points in BLEU/chrF), whereas German and Uzbek seem to benefit from more merge operations (approx. +1 to +2 points in BLEU/chrF). We find this perplexing, as the German and Uzbek datasets are the largest and smallest used in our experiments.

To analyze these results, we fit a hierarchical, Bayesian linear model with language-specific effects for the BPE variant and number of merge operations:

$$\mu = \alpha^{(l)} + \beta_b^{(l)} + \gamma_m^{(l)} + \epsilon$$

where $\alpha^{(l)}$ is an intercept, $\beta_b^{(l)}$ is the effect of using BPE variant b, $\gamma_m^{(l)}$ is the effect of using m merge operations, and ϵ represents residual sampling error. All effects are specific to language l and are drawn from common prior distributions: $\alpha^{(l)} \sim \mathcal{N}(0, \sigma_\alpha^2)$, $\beta_b^{(l)} \sim \mathcal{N}(\bar{\beta}_b, \sigma_{\beta,b}^2)$ and $\gamma_m^{(l)} \sim \mathcal{N}(\bar{\gamma}_m, \sigma_{\gamma,m}^2)$. Since our model is hierarchical, we also infer posteriors for the language-independent effects of using each BPE variant/number of merge operations, $\bar{\beta}_b$ and $\bar{\gamma}_m$, as well as the standard deviations $\sigma_{\beta,b}^2$ and $\sigma_{\gamma,m}^2$ that quantify between-language variation in the BPE and merge effects. We set the priors of the average effects to $\mathcal{N}(0, 1)$ and those of the standard devi-

ations to $\mathcal{N}^+(1)$, except for σ_α^2 for which we use the default $\mathcal{N}^+(s_\alpha), s_\alpha \approx 68$ prior specified by the Bambi modeling library (Capretto et al., 2022) which we use to fit our model. We fit all our models using the No-U-Turn Sampler (Hoffman and Gelman, 2014). We run 4 Markov chains in parallel and draw 1,000 posterior samples from each chain. Prior to sampling, we also run each chain for 1,000 warm-up steps.

Table 2 shows a posterior mean point estimate for each effect of interest and quantifies their uncertainty using a 94% highest density interval (HDI). The effect sizes of randomized BPE variants seem confirm our experimental results. While the language-independent average effect sizes are all modest in magnitude, ranging from -0.97 for uniform BPE to +0.61 for standard BPE, there is substantial variation in the effect sizes when using uniform random sampling: effect sizes ranging from -6.65 for Uzbek to +0.26 for Estonian. Most importantly, the uncertainty intervals include zero for all languages and BPE types except for $\beta_{\text{Uniform, German}}$ and $\beta_{\text{Uniform, Uzbek}}$.

The effects for the number of merges are largely similar, with small average effects and between-language variation in effects on both sides of zero. While effect sizes tend to be very small for 2,000 and 5,000 merge operations, the effect varies with 32K merges. German and Uzbek seem to benefit from using 32K merge operations (posterior means 0.59 and 2.30, respectively). In contrast, Finnish and Estonian have significantly negative effect sizes (-1.83 and -1.31, respectively) with the entire 94% HDI for Finnish below zero as well.

62

Parameter	Mean	HDI (lower)	HDI (upper)	Incl. zero?
BPE effects (average)				
$\bar{\beta}_{\text{Standard}}$	0.61	-0.71	1.85	✓
$\bar{\beta}_{\text{Uniform}}$	-0.97	-2.57	0.58	✓
$\bar{\beta}_{\text{Random}}$	0.35	-0.92	1.68	✓
BPE effects (language-specific)				
$\beta_{\text{Standard, Estonian}}$	0.45	-1.00	1.86	✓
$\beta_{\text{Standard, Finnish}}$	0.47	-0.98	1.94	✓
$\beta_{\text{Standard, German}}$	0.57	-0.95	1.95	✓
$\beta_{\text{Standard, Uzbek}}$	1.26	-0.38	2.83	✓
$\beta_{\text{Random, Estonian}}$	0.39	-1.13	1.74	✓
$\beta_{\text{Random, Finnish}}$	0.43	-1.00	1.90	✓
$\beta_{\text{Random, German}}$	0.44	-0.92	1.89	✓
$\beta_{\text{Random, Uzbek}}$	0.26	-1.34	1.73	✓
$\beta_{\text{Uniform, Estonian}}$	0.26	-1.18	1.77	✓
$\beta_{\text{Uniform, Finnish}}$	-0.51	-2.05	0.94	✓
$\beta_{\text{Uniform, German}}$	-1.67	-3.04	-0.08	
$\beta_{\text{Uniform, Uzbek}}$	-6.65	-8.29	-5.01	
Merge effects (average)				
$\bar{\gamma}_{2000}$	0.10	-1.06	1.44	✓
$\bar{\gamma}_{5000}$	0.00	-1.24	1.21	✓
$\bar{\gamma}_{32000}$	-0.03	-1.44	1.39	✓
Merge effects (language-specific)				
$\gamma_{2000, \text{Estonian}}$	0.18	-1.11	1.59	✓
$\gamma_{2000, \text{Finnish}}$	0.33	-1.00	1.72	✓
$\gamma_{2000, \text{German}}$	-0.04	-1.40	1.28	✓
$\gamma_{2000, \text{Uzbek}}$	-0.04	-1.43	1.37	✓
$\gamma_{5000, \text{Estonian}}$	0.05	-1.29	1.41	✓
$\gamma_{5000, \text{Finnish}}$	-0.03	-1.39	1.26	✓
$\gamma_{5000, \text{German}}$	0.09	-1.18	1.48	✓
$\gamma_{5000, \text{Uzbek}}$	-0.08	-1.54	1.26	✓
$\gamma_{32000, \text{Estonian}}$	-1.31	-2.67	0.16	✓
$\gamma_{32000, \text{Finnish}}$	-1.83	-3.26	-0.44	
$\gamma_{32000, \text{German}}$	0.59	-0.82	1.96	✓
$\gamma_{32000, \text{Uzbek}}$	2.30	0.79	3.75	

Table 2: Posterior means and 94% posterior highest density intervals for the BLEU model.

5 Discussion

5.1 Limitations and future work

While our results suggest that randomized BPE segmentation algorithms have no consistent deleterious effect on BLEU/chrF across languages, it is possible that further experiments may find differently. There is room for exploration regarding randomization of the BPE algorithm. For example, instead of sampling from the set of observed symbol pairs, merge operations could be chosen by sampling two unigrams independently or using a temperature-augmented sampler.

Although we focus on morphologically rich languages, our experiments still utilize a moderate amount of training data. Many morphologically rich languages that we did not consider may also lack such resources and thus be more impacted by the choice of subword segmentation algorithm. We feel that future work should pay particular attention should to this intersection of morphological complexity and low-resourcedness.

5.2 Conclusion

We introduced two randomized variants of BPE with the expectation that they would have a negative effect on translation performance because the traditional greedy approach should result in better subwords. Instead, our results indicate that subword vocabularies created with randomized BPE yield translation models that perform comparably to those that use subwords created using the standard greedy BPE algorithm. Even when using uniform sampling, performance only degrades substantially for two of the languages we consider. This finding is corroborated by further analysis using a Bayesian linear model which suggests that the effect of uniform sampling is significantly different from zero for only German and Uzbek.

We find this negative result significant, as it suggests that variations on standard BPE can perform reasonably well. We emphasize, however, that it is not clear whether this holds universally, particularly when using Transformer architectures optimized for handling longer sequences or when working with extremely small amounts of training data. We hope that our negative result can motivate further research into the optimal use of subword segmentation algorithms, especially in the context of languages that are both morphologically rich and less-resourced, such as various Indigenous languages.

References

Tomás Capretto, Camen Piho, Ravin Kumar, Jacob Westfall, Tal Yarkoni, and Osvaldo A Martin. 2022. Bambi: A Simple Interface for Fitting Bayesian Linear Models in Python. *Journal of Statistical Software*, 103.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Thamme Gowda and Jonathan May. 2020. Finding the optimal vocabulary size for neural machine translation. In *Findings of the Association for Computa-*

tional Linguistics: EMNLP 2020, pages 3955–3964, Online. Association for Computational Linguistics.

Ximena Gutierrez-Vasques, Christian Bentz, Olga Sozinova, and Tanja Samardzic. 2021. From characters to words: the turning point of BPE merges. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3454–3468, Online. Association for Computational Linguistics.

Barry Haddow, Nikolay Bogoychev, Denis Emelin, Ulrich Germann, Roman Grundkiewicz, Kenneth Heafield, Antonio Valerio Miceli Barone, and Rico Sennrich. 2018. The University of Edinburgh's submissions to the WMT18 news translation task. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 399–409, Belgium, Brussels. Association for Computational Linguistics.

Xuanli He, Gholamreza Haffari, and Mohammad Norouzi. 2020. Dynamic programming encoding for subword segmentation in neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3042–3051, Online. Association for Computational Linguistics.

Matthew D. Hoffman and Andrew Gelman. 2014. The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo. *J. Mach. Learn. Res.*, 15(1):1593–1623.

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.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Sabrina J. Mielke, Zaid Alyafeai, Elizabeth Salesky, Colin Raffel, Manan Dey, Matthias Gallé, Arun Raja, Chenglei Si, Wilson Y. Lee, Benoît Sagot, and Samson Tan. 2021. Between words and characters: A brief history of open-vocabulary modeling and tokenization in NLP. *arXiv preprint 2112.10508*.

Jamshidbek Mirzakhalov, Anoop Babu, Aigiz Kunafin, Ahsan Wahab, Bekhzodbek Moydinboyev, Sardana Ivanova, Mokhiyakhon Uzokova, Shaxnoza Pulatova, Duygu Ataman, Julia Kreutzer, Francis Tyers, Orhan Firat, John Licato, and Sriram Chellappan. 2021. Evaluating multiway multilingual NMT in the Turkic languages. In *Proceedings of the Sixth Conference on Machine Translation*, pages 518–530, Online. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Maja Popović. 2015. chrF: character n-gram F-score for automatic MT evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395, Lisbon, Portugal. Association for Computational Linguistics.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. 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.

Xinying Song, Alex Salcianu, Yang Song, Dave Dopson, and Denny Zhou. 2021. Fast WordPiece tokenization. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2089–2103, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

David Vilar and Marcello Federico. 2021. A statistical extension of byte-pair encoding. In *Proceedings of the 18th International Conference on Spoken Language Translation (IWSLT 2021)*, pages 263–275, Bangkok, Thailand (online). Association for Computational Linguistics.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint 1609.08144*.

A   Additional Tables and Figures

Corpus statistics   Table 3 shows relevant statistics for each translation dataset, including number of sentences, token and type counts, and type-to-token ratios.

Translation performance   Figure 2 shows a visualization of translation performance in terms of BLEU and chrF across languages and number of merge operations.

Language	Split	Sentences	Tokens		Types		Type-to-token ratio	
			English	Non-English	English	Non-English	English	Non-English
Estonian	Train	1,856,236	32,850,284	27,221,588	361,245	713,970	0.01	0.03
	Dev	2,000	45,892	36,333	7,731	12,275	0.17	0.34
	Test	2,000	48,340	38,063	8,085	12,956	0.17	0.34
Finnish	Train	1,754,754	43,898,422	32,012,655	116,620	677,874	0.00	0.02
	Dev	1,500	34,251	24,617	6,251	10,005	0.18	0.41
	Test	1,370	29,183	21,142	5,761	8,958	0.20	0.42
German	Train	4,173,550	99,557,517	94,741,339	881,684	1,805,238	0.01	0.02
	Dev	3,000	67,807	66,412	9,778	12,859	0.14	0.19
	Test	3,003	70,620	66,081	10,607	14,053	0.15	0.21
Uzbek	Train	529,574	11,502,156	9,361,833	120,768	250,629	0.01	0.03
	Dev	2,500	52,963	42,701	8,312	13,847	0.16	0.32
	Test	2,500	54,061	43,945	8,349	13,265	0.15	0.30

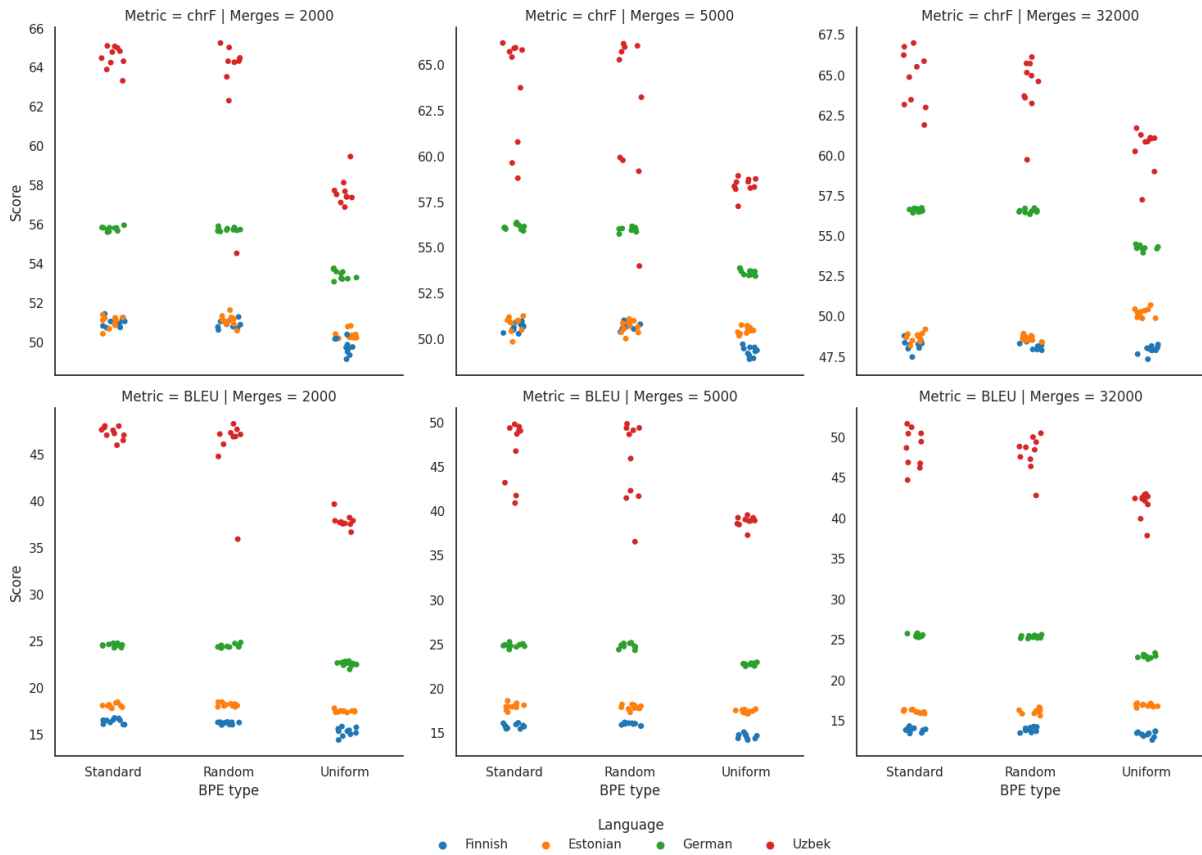Table 3: Counts of sentences, tokens and word types in our corpora.

Figure 2: Translation performance across languages and numbers of merges using BLEU (top) and chrF (bottom).