# MUNI-NLP Systems for Lower Sorbian-German and Lower Sorbian-Upper Sorbian Machine Translation @ WMT22

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

## Abstract

We describe our neural machine translation systems for the WMT22 shared task on unsupervised MT and very low resource supervised MT. We submit supervised NMT systems for Lower Sorbian-German and Lower Sorbian-Upper Sorbian translation in both directions. By using a novel tokenization algorithm, data augmentation techniques, such as Data Diversification (DD), and parameter optimization we improve on our baselines by 10.5-10.77 BLEU for Lower Sorbian-German and by 1.52-1.88 BLEU for Lower Sorbian-Upper Sorbian.

## Introduction

This paper describes our Machine Translation (MT) systems for the WMT22 shared task on "Unsupervised MT and Very Low Resource Supervised MT"[1], which features translation between Lower Sorbian, Upper Sorbian, and German. Lower (*dsb* and Upper Sorbian (*hsb*) are Slavic minority languages spoken in the Eastern part of Germany with 7.000 and 30.000 native speakers respectively. Text data for these languages collected and made available by the Sorbian Institute and the Witaj Language Centre (Libovický and Fraser, 2021).

We submit systems for Lower Sorbian-German and Lower Sorbian-Upper Sorbian in both translation directions. We focused on the supervised approach, using only the parallel data made available by the task organizers for all the above languages.

We were able to improve on our baselines by: i. employing a new tokenization algorithm, High Frequency Tokenizer (HFT) (Signoroni and Rychlý, 2022); ii. augmenting the original parallel data with the Data Diversification (DD) technique by (Nguyen et al., 2020); iii. tuning the architecture and the parameters of the models, such as encoder/decoder depth, number of attention heads, dropout, batch size, etc.

[1]https://statmt.org/wmt22/unsup_and_very_low_res.html

We employed HFT since it aims to obtain more meaningful subword dictionaries, while DD was chosen because it does not involve additional data apart from the original parallel corpus. Both this techniques are relevant when working with a limited amount of data.

This paper is structured as follows: Section 1 summarizes the data used in training; Section 2 outlines our methodology, introducing our novel tokenizer and the models we used; Section 3 sums up our final systems, while Section 4 relates and discusses the results of our experiments; Section 5 contains some final remarks.

## 1 Data

We experiment with Lower Sorbian-German and Lower Sorbian-Upper Sorbian translation, using only the parallel data provided for each pair.

The parallel data for the dsb-de and the dsb-hsb pairs consist of ∼40k and ∼62k sentences respectively. We use only these data, as the approach we decided to follow does not need additional data. After applying this method, our final corpus size for training is ∼360k sentences for dsb-de, and ∼560k for dsb-hsb.

## 2 Methodology

In this section, we first present briefly our novel tokenizer, High Frequency Tokenizer, or HFT. Then, we describe the architecture of our models and how we trained them.

### 2.1 High Frequency Tokenization

Sennrich and Zhang (2019) showed that a meaningful subword tokens vocabulary is crucial for achieving good performance in low-resource NMT. While they experiment with BPE, we employ our novel tokenization algorithm, High Frequency Tokenizer, or HFT. This word segmentation methods aims to provide more meaningful subword dictionaries by obtaining more frequent, and thus better



*Proceedings of the Seventh Conference on Machine Translation (WMT)*, pages 1111–1116
Abu Dhabi, December 7–8, 2022. 

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

Figure 1: Special characters in the pretokenization and tokenization.

represented subwords. Given the importance of tokenization for low-resource NMT, we argue that is an important point to consider.

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`[2] command . Both the beginning and the end of each token is explicitly annotated, differently from previous systems such as `subword-nmt` BPE and `sentencepiece`.

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 1 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 and 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.

We evaluated HFT against Byte-Pair Encoding (BPE) (Sennrich et al., 2016) and Unigram (Kudo, 2018) on the metrics described by (Gowda and May, 2020) and on a weighted average of the frequencies of the tokens in the vocabulary. HFT performed well, providing better results for almost all test cases. Preliminary data on HFT's impact on downstream NMT also showed promising results. (Signoroni and Rychlý, 2022)

Moreover, during the experimentation for this task, we further confirmed that using HFT-tokenized data leads to better translation quality, calculated with sacreBLEU (Post, 2018), against a `subword-nmt` BPE baseline.

### 2.2 Data Diversification

For our final models we follow the Data Diversification (DD) approach of Nguyen et al. (2020). While most of the research in low-resource NMT avails itself of external data by employing techniques such as backtranslation and transfer learning, this simple, yet effective method does not need any external data, but only the original parallel corpus. The DD procedure is the following:

1. Train $k$ different models on the authentic parallel corpus, in both the forwards and backwards directions;

2. Infer the translations with all the trained models, so to obtain $k$ synthetic source and target data;

3. merge the translations to create a new parallel dataset, which comprises the original parallel data, plus an authentic source to synthetic target, and a synthetic source to authentic target section;

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

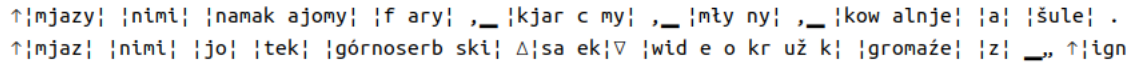


Figure 2: Sample of Lower Sorbian text tokenized with HFT.

4. Train new models on these augmented parallel data;

5. repeat for *n* rounds.

Since Nguyen et al. (2020) reports that additional rounds of DD do not boost the performance of the resulting systems significantly, our systems were trained after just one round of DD. We have a level of diversification *k*=4, since we included all the previous experiments models' output, plus the original parallel data.

### 2.3 Preprocessing

We use both training and development test data as provided, without cleaning of any kind. We tokenize the data with two subword tokenization algorithms, BPE and HFT. For the former, we use the `subword-nmt` implementation. For HFT, we use our own implementation. We experiment with different vocabulary sizes, and our results are in line with previous research, such as Sennrich and Zhang (2019). They showed that a smaller vocabulary sizes improves the performance of low-resource NMT. In line with these findings, our experiments with vocabulary size of 12k and 10k, even if well below the standard 32k, performed worse than our final choice of 4k tokens.

We train a tokenizer for each language on the train split of the datasets, and share the dictionaries during all the stages of training.

### 2.4 Models

Table 1 gives details about the architecture and training parameters of each model we trained.

We experiment with two different model architectures, both based on the Transformer (Vaswani et al., 2017). The first, which we dubbed t-[tok][3], is a standard Transformer (Vaswani et al., 2017); while the second, called and t-opt-[tok], is a Transformer with optimized parameters for the size of the dataset.

We use Fairseq (Ott et al., 2019) for training the models, generating translations, and evaluating them.

As our baseline, we train a Transformer (Vaswani et al., 2017) with default hyper parameters and BPE tokenization, which we refer to as t-bpe. [4] As a first experiment, we train t-hft, a standard Transformer trained on data tokenized with HFT. We use *adam* as optimizer and we maximize BLEU score on the validation set at each epoch. For the BPE models, we use detokenized BLEU, but for HFT this was not implemented during training. We train both t-[tok] models for 100 epochs with dropout of 0.1, and 10240 maximum tokens for each batch. We use a learning rate of 0.0005 and the inverted square root scheduler for all of our models.

Secondly, we train another Transformer with the optimized hyper parameters found by (Araabi and Monz, 2020) for a dataset of 40k sentence pairs: 5 encoder/decoder layers, 2 attention heads, and a feed-forward dimension of 2048. During our experiments, however, we observed that a feed-forward dimension of 1024 gives better results. We do this with data tokenized with both methods, obtaining t-opt-bpe and t-opt-hft. These models are trained for 100 epochs, with dropout of 0.3, label smoothing of 0.5, encoder and decoder word dropout of 0.1, activation dropout of 0.3, and a maximum batch size of 4096 tokens.

Lastly, we build the DD parallel corpus by collating the outputs of all previous systems in both directions, beginning with the authentic parallel data, and adding both combinations of original and synthetic source and target data. We then use the DD-data to train both t-bpe-dd and t-hft-dd, which share the same architecture the t-[tok] models. The training is also similar, just differing in the number of epochs, which for these last models is 50.

### 2.5 Evaluation

To find our best candidates for submission, we generate translation on a development test set of unseen sentence pairs, either provided by the task organizers, or set aside from the train portion of the data. We produce translations with the standard settings (beam search with a beam of 5) using the best

[3]We trained models on data tokenized both with BPE and HFT. Since they share the same architecture and training parameters, [tok] stands either for bpe or hft.

[4]We still use a vocabulary size of 4k, which is already an improvement on the standard size of 32k.

| PARAMETER | MODEL | | |
|---|---|---|---|
| | t-[tok] | t-opt-[tok] | t-[tok]-dd |
| Vocabulary size | 4000 | 4000 | 4000 |
| feed-forward dimension | 2048 | 1024 | 2048 |
| attention heads | 8 | 2 | 8 |
| dropout | 0.1 | 0.3 | 0.3 |
| enc/dec layers | 6 | 5 | 6 |
| label smoothing | 0.1 | 0.5 | 0.1 |
| enc/dec word dropout | 0.0/0.0 | 0.1/0.1 | 0.0/0.0 |
| activation dropout | 0.0 | 0.3 | 0.0 |
| max tokens | 10240 | 4096 | 10240 |

Table 1: Architecture details and training parameters for each model.

| | DSB-DE | DE-DSB | DSB-HSB | HSB-DSB |
|---|---|---|---|---|
| t-bpe | 27.92 | 22.74 | 72.01 | 69.71 |
| t-hft | 34.20 | 30.86 | 72.21 | 70.71 |
| t-opt-bpe | 29.75 | 25.06 | 71.37 | 69.50 |
| t-opt-hft | 35.46 | 31.12 | 71.83 | 68.95 |
| t-bpe-dd | 33.02 | 28.54 | 73.47 | 71.98 |
| t-hft-dd | 38.42 | 33.53 | 73.53 | 71.59 |

Table 2: Trained models and BLEU score during inference on development test data

checkpoint of each model and evaluate the detokenized output with sacreBLEU (Post, 2018). For the BPE models, we detokenize with the provided argument, while for HFT we use our own plug-in script. We use the same settings to translate the test set for our submissions. Table 2 gives BLEU scores, computed on the development test sets, for each system we experimented with.

During inference on the development test set, models trained on HFT data outperformed the BPE baseline by 4.99 to 8.12 BLEU for the dsb-de pair, while for dsb-hsb the difference in score is minimal. t-opt-[tok] was better than the corresponding t-[tok] model in the dsb-de pair. For dsb-hsb, this does not hold true, with t-opt-[tok] always performing worse than the baseline. t-[tok]-dd improves on both the baseline and t-opt-[tok] for every language pair and direction.

### 2.6 Inconclusive and Negative Results

During DD, we collated data from all four experimental models for each pair, both t-[tok] and t-opt-[tok], regardless of their performance. This later resulted in our best systems. For the dsb-de pair, we also tried to ensemble data from the four best performing systems to create the DD train set, all from HFT data and ranging from 34.99 to 37.20 BLEU on the dsb-de side, and 31.12 to 30.42 on the reverse direction. This was done with the intuition that better train data should result in better performance. However, after training t-[tok]-dd on these data, the resulting system that performed worse by -0.58, giving 37.84 BLEU on the development test set. In contrast, the final t-[tok]-dd gave us 38.42 BLEU, and was trained on data generated with systems ranging from 27.92 to 35.46 BLEU on the dsb-de side, and from 22.74 to 31.12 for the reverse direction.

While this small difference in BLEU score may not be significative, further investigation should be conducted as this may indicate that a more diverse dataset is better than one with a higher quality for training with NMT systems with DD. Our initial hypothesis for why this happens is that being the systems' performance closer, the translations they generate are similar. This leads to worse generalization potential for the resulting final system.

## 3 Final Systems

Table 3 gives BLEU and chrF scores for our final submissions.

For all pairs and directions we worked on, our best systems was t-hft-dd, a Transformer trained on a single NVIDIA A40[5] for 50 epochs on DD

[5]Previous systems were always trained on a single GPU,

| | | t-hft-dd |
|---|---|---|
| DSB-DE | BLEU | 49.5 |
| | chrF | 73.0 |
| DE-DSB | BLEU | 50.5 |
| | chrF | 74.1 |
| DSB-HSB | BLEU | 72.2 |
| | chrF | 87.5 |
| HSB-DSB | BLEU | 72.3 |
| | chrF | 87.5 |

Table 3: BLEU and chrF scores for our best systems on the final test set.

HFT-tokenized data, with a vocabulary size of 4k, feed-forward dimension of 2048, dropout 0.3, and 10240 for each batch.[6]

Our hypotheses on why HFT leads to improvements on these datasets are the following. On top of providing more frequent and better defined token for the model to learn, it also explicitly marks both beginning and end of the words during pretokenization. This could be relevant for morphologically complex languages, such as the ones in this task, since it provides more information to the model on possible prefixes and suffixes. Contrast this with the fact that `subword-nmt` BPE only explicitly mark continuation with the @@ marker. This kind of tokenization thus makes no distinction between full words and word endings. Moreover, it seems to struggle with capitalized words and punctuation, which can also be informative, if handled optimally. HFT's pretokenization seems to help with this issue. Investigating these topics will be further addressed by future work.

## 4 Conclusions

This paper described our submission for the WMT22 shared task on Unsupervised MT and Very Low Resource Supervised MT. We presented systems for Lower Sorbian-German and Lower Sorbian-Upper Sorbian translation in both direction under supervised training conditions. To train our best systems we employed a novel tokenization algorithm, HFT, to obtain more meaningful subword vocabularies, contrasted to a BPE baseline; and Data Diversification (Nguyen et al., 2020) to augment the training data using only the parallel dataset provided for each language pair.

During our experiments we confirmed that optimizing not only the Transformer's hyper parameters, but also the subword vocabulary quality and size, are crucial steps for low-resource NMT. Choosing the appropriate vocabulary size for the dataset, could lead to significant improvements in BLEU score even with a small amount of parallel data. These, however, are still open and complex problems, since previously proposed settings or, even more so default ones, did not always provide the best results.

## Limitations and Future Work

While providing NMT systems for Lower Sorbian-German and Lower Sorbian-Upper Sorbian that perform reasonably well, some other methods, some of which were used by other submissions, could provide better results. These should be taken into account, if a system for these language pairs will be deployed for language conservation, revitalization, or everyday use. Such system may also be hampered by the limited scope of the training data, which is inherent in their size. As for other low-resource and, especially for endangered languages, documentation ventures such as those of the Sorbian Institute and the Witaj Language Centre are vital to create bigger, more comprehensive datasets, which are still needed for the current NLP methodologies to work at their best.

## Ethics Statement

NMT systems are, as every other data-driven technology, sensible to biases and other shortcomings in their training data. De-biasing datasets and NLP systems' output is a scope of research that lies outside the scope of this shared task and thus, from the scope of this paper. If these systems were to be employed in a real-world scenario such as language conservation, revitalization or everyday translation, we advise caution as to the limitations mentioned above.

Following Lacoste et al. (2019), we report that the experiments and the research that led to the results presented in this paper were conducted on a private server infrastructure consisting of a NVIDIA Tesla T4, A40, and A100 for around 500 hours of training at an efficiency of 0.59 kg/kWh[7] for a total of 20.39 kg $CO_2$ eq.

variably on a A40, A100 or a Tesla T4. Training times did not exceed 12 hours for both t-[tok]-dd systems.

[6]Every other unmentioned parameter was left at the default setting.

[7]The Czech Republic's country average as reported in https://www.carbonfootprint.com/docs/2018_8_electricity_factors_august_2018_-_online_sources.pdf

## Acknowledgements

We thank the reviewers for their useful comments. The work of the authors is partly supported by the Internal Grant Agency of Masaryk University, by Lexical Computing and by the Ministry of Education of the Czech Republic within the LINDAT-CLARIAH-CZ project LM2018101.

## References

Ali Araabi and Christof Monz. 2020. Optimizing transformer for low-resource neural machine translation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3429–3435, Barcelona, Spain (Online). International Committee on Computational Linguistics.

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.

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.

Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres. 2019. Quantifying the carbon emissions of machine learning. *arXiv preprint arXiv:1910.09700*.

Jindřich Libovický and Alexander Fraser. 2021. Findings of the WMT 2021 shared tasks in unsupervised MT and very low resource supervised MT. In *Proceedings of the Sixth Conference on Machine Translation*, pages 726–732, Online. Association for Computational Linguistics.

Xuan-Phi Nguyen, Shafiq Joty, Wu Kui, and Ai Ti Aw. 2020. Data diversification: A simple strategy for neural machine translation. In *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc.

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.

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.

Edoardo Signoroni and Pavel Rychlý. 2022. HFT: High frequency tokens for low-resource NMT. In *Proceedings of the Fifth Workshop on Technologies for Machine Translation of Low-Resource Languages (LoResMT 2022)*, pages 56–63, Gyeongju, Republic of Korea. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.