# ÚFAL at MultiLexNorm 2021: Improving Multilingual Lexical Normalization by Fine-tuning ByT5

**David Samuel** and **Milan Straka**
Charles University,
Faculty of Mathematics and Physics,
Institute of Formal and Applied Linguistics
`{samuel,straka}@ufal.mff.cuni.cz`

## Abstract

We present the winning entry to the *Multilingual Lexical Normalization (MultiLexNorm)* shared task at *W-NUT 2021* (van der Goot et al., 2021a), which evaluates lexical-normalization systems on 12 social media datasets in 11 languages. We base our solution on a pre-trained byte-level language model, ByT5 (Xue et al., 2021a), which we further pre-train on synthetic data and then fine-tune on authentic normalization data. Our system achieves the best performance by a wide margin in intrinsic evaluation, and also the best performance in extrinsic evaluation through dependency parsing. The source code is released at `https://github. com/ufal/multilexnorm2021` and the fine-tuned models at `https://huggingface. co/ufal`.

## 1 Introduction

People produce text in natural language every minute of every day. However, in many cases, for example on social media like Twitter, such texts are not conforming to a formal style. Instead, they are in colloquial form, which is perfectly understandable to other people, but challenging for automatic natural language processing. To make the processing of such texts more viable, the task of *lexical normalization* can be used to replace the input forms with their canonical (more formal, lexically normalized) variants.

The aim of the *W-NUT 2021: Multilingual Lexical Normalization (MultiLexNorm)* shared task (van der Goot et al., 2021a) is to evaluate participant lexical-normalization systems on 12 social media datasets in 11 languages, including two code-switching datasets. Both intrinsic and extrinsic evaluation is performed, where the latter is measured through dependency parsing performed on the normalized data.

Recently, large pre-trained models like BERT (Devlin et al., 2019) or T5 (Raffel et al., 2020) have demonstrated superior performance in many NLP tasks when trained in a transfer learning setting. In line with that, we approach the lexical normalization shared task as a fine-tuning of a large pre-trained model, specifically the multilingual byte-level generative language model ByT5 (Xue et al., 2021a).

Our system achieves the best performance in the shared task, both in intrinsic and extrinsic evaluations. In the intrinsic evaluation, our system reaches 67.3% error reduction compared to leave-as-is baseline; the second-best system has 53.6%.

The source code is released at `https://github. com/ufal/multilexnorm2021` and the fine-tuned models are available in HuggingFace Transformers (Wolf et al., 2020) at `https://huggingface. co/ufal`.

## 2 Related Work

Lexical normalization can be considered a simplified variant of a well-studied problem of *grammar error correction* (GEC). Grundkiewicz et al. (2019) approach GEC as a neural machine translation task using the Transformer architecture (Vaswani et al., 2017), which is pre-trained using a vast amount of synthetic data generated by character-level and word-level edits. Recently, Rothe et al. (2021) presented a GEC system based on multilingual mT5 (Xue et al., 2021b), reaching state-of-the-art results on several datasets with the gigantic *xxl* model size with 13B parameters.

While the mentioned GEC systems are autoregressive, lexical normalization can be easily solved in a non-autoregressive way, because the normalizations of different words are relatively independent. Besides, successful non-autoregressive models have been recently proposed for general GEC (Awasthi et al., 2019; Omelianchuk et al., 2020).

Although fine-tuned language models have been successfully used in the state-of-the-art GEC systems, this has not been the case in the field of lexi-
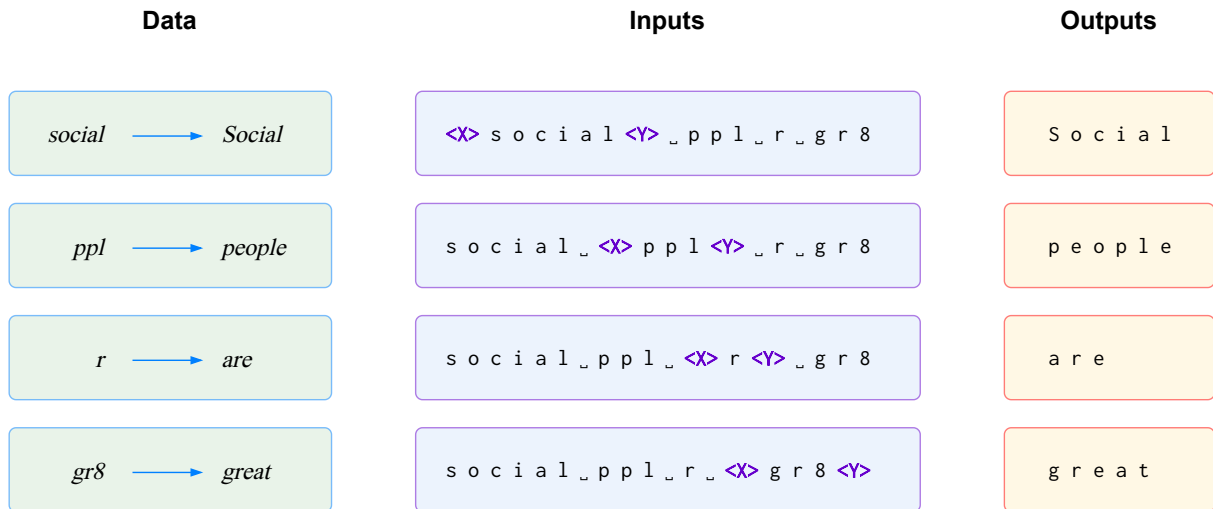
Figure 1: The inputs and outputs of our lexical normalization model. The ByT5 sentinel tokens `<X>` and `<Y>` mark the word to be normalized.

cal normalization (Muller et al., 2019; Lourentzou et al., 2019). MoNoise (van der Goot, 2019) is a publicly available multilingual lexical normalization tool achieving competent performance (an improved version of the system would place third in the shared task intrinsic evaluation). It utilizes Aspell dictionaries, FastText (Bojanowski et al., 2017) embeddings and hand-crafted language features, but no contextualized embeddings.

## 3 Model

Our model is based on a large pre-trained multilingual model. In accordance with Bommasani et al. (2021), we call such models (like BERT or T5) the *foundation* models.

Specifically, we utilize the ByT5 (Xue et al., 2021a) foundation model. It is a byte-level generative sequence-to-sequence model, which processes a sequence of bytes of UTF-8 encoding on input and produces a sequence of UTF-8 encoding bytes on output. ByT5 models were proposed as an alternative to multilingual mT5 foundation models (Xue et al., 2021b) and have been shown to perform remarkably well on noisy text data (compared to subword-based mT5), like TWEETQA (Xiong et al., 2019).

### 3.1 Input and Output Format

We start by recapitulating the pre-training task of the ByT5 model. The input sentence (including spaces) is represented as a sequence of bytes of UTF-8 encoding, and spans of around 20 bytes are
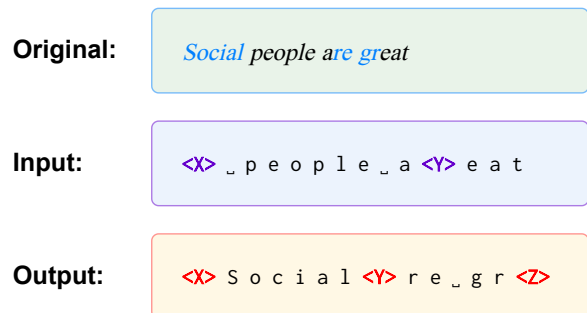


Figure 2: The pre-training task utilized in the ByT5 model (Xue et al., 2021a). The `<X>`, `<Y>` and `<Z>` are the sentinel tokens.

masked using special *sentinel tokens*. The goal of the model is to reconstruct all the masked spans. We illustrate the task visually in Figure 2.

For lexical normalization, we could directly use the unnormalized sentence as input and the normalized sentence as output (an approach used by Rothe et al. (2021) for GEC). However, we were concerned that such an approach would be too different from the ByT5 pre-training, and furthermore, it would not allow to reconstruct the alignment of the normalized tokens when a word is removed during normalization or split into several words.

Instead, we opted for a different approach illustrated in Figure 1. For each input word, we construct a separate ByT5 input, in which we mark the beginning and the end of the word in question using two sentinel tokens. Then we ask the model to produce just the normalization of that word. Such an approach normalizes each input word indepen-
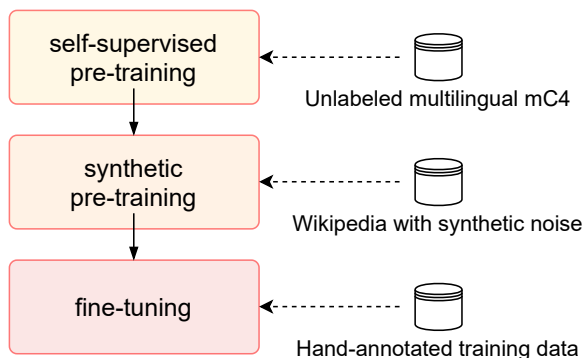
Figure 3: Three stages of training of our models: the original ByT5 pre-training, pre-training on synthetic data and final fine-tuning of authentic data.

dently, and we consider it quite similar to the original ByT5 pre-training task. Unfortunately, it requires to encode different input sequences for every input word, which is considerably inefficient, even if we can perform normalization of all words in parallel.

## 3.2 Pre-training on Synthetic Data

Fine-tuning a ByT5 model directly with supervised training data would not deliver very high performance, given that the normalization task is (despite our efforts) still quite different from the pre-training task, and that the amount of available training data is quite low.[1]

Therefore, before fine-tuning, we first pre-train the ByT5 model using synthetic data, as illustrated in Figure 3. Note that from now on, by *"pre-train"* we mean the training of the ByT5 foundation model using synthetic lexical normalization data.

We construct the synthetic data by modifying clean data in the following way:

- If a word is present in the normalized output in the training data, it is replaced by one of the corresponding (possibly unnormalized) training data inputs, proportionally to the number of occurrences. For example, we change *"people"* to *"ppl"* with 39.7% probability in the English dataset or *"ikke"* to *"ik"* with 4.97% chance in the Danish dataset.
- A large portion of the required changes can be realized as simple character-level modifications. Thus, we synthesize the data by reversing these alterations. These are 1) accent mark removal (e.g., replacing *"š"* by *"s"* with 16.3% chance in Slovenian), 2) changing

capitalization (e.g., lowering the first capitalized letter with 1.0% chance in Turkish), 3) removing apostrophes (46.8% probability in English), and 4) other miscellaneous modifications (e.g., simplifying *"qu"* to *"k"* in Spanish with 2.0% probability).

- The training data naturally contain a large quantity of different typographical errors, usually caused by inaccurate typing on a keyboard. To simulate this behavior, we alter the words by skipping some characters, changing or inserting some characters (more likely to those close on a computer keyboard), and by reversing two consecutive characters. To give one example, the probability of making an artificial typo in Italian is 0.458%.
- Some modifications are unique for a specific language. For example, the plural forms in Indonesian can be created by duplicating the singular forms. As users want to save time when writing in Indonesian, they sometimes indicate the plural by simple appending "2" to the singular form. We reverse this transformation with 33.3% probability (thus, *"laki-lakinya"* becomes *"laki2nya"*).
- 7 datasets in this shared task split or merge words. We synthetically split/merge words on the clean dataset to model these actions. For example, the probability of merging two words in Dutch is 5.99% and of splitting a word is 0.0565%.
- In order to speed up typing, the users modify their language in various ways – e.g. by omitting vowels or shortening words to prefixes. On the other hand, they repeat certain characters to make their messages more expressive. We include these variations in the synthetic datasets to get closer to the real data.

For more details, please consult the source code.

It is important to note that all the probabilities were estimated from the training data. Therefore the synthetic pre-training cannot be regarded as completely unsupervised; one would need expert knowledge to extend this type of pre-training on a language without annotated data of lexical normalization.

We need to make sure that most of the original data (before the synthetic modifications) are clean. Since we also want an ample amount of cross-lingual text, we opted to use publicly avail-

---

[1]This experiment is included in ablations in Section 5.3.

| Code | Language | Original Source | Words | Words split/merged | Caps | % words normed | MFR ERR |
|------|----------|-----------------|-------|--------------------|------|----------------|---------|
| DA | Danish | Plank et al. (2020) | 11 816 | ✓ | ✓ | 9.25 | 49.68 |
| DE | German | Sidarenka et al. (2013) | 25 157 | ✓ | ✓ | 17.96 | 32.09 |
| EN | English | Baldwin et al. (2015) | 73 806 | ✓ | ✗ | 6.90 | 64.93 |
| ES | Spanish | Alegria et al. (2013) | 13 827 | ✗ | ✗ | 7.69 | 25.57 |
| HR | Croatian | Ljubešić et al. (2017a) | 75 276 | ✗ | ✗ | 8.89 | 36.52 |
| ID-EN | Indonesian-English | Barik et al. (2019) | 23 124 | ✓ | ✗ | 12.16 | 61.17 |
| IT | Italian | van der Goot et al. (2020) | 14 641 | ✓ | ✓ | 7.32 | 16.83 |
| NL | Dutch | Schuur (2020) | 23 053 | ✓ | ✓ | 28.29 | 37.70 |
| SR | Serbian | Ljubešić et al. (2017b) | 91 738 | ✗ | ✗ | 7.65 | 42.62 |
| SL | Slovenian | Erjavec et al. (2017) | 75 276 | ✗ | ✗ | 15.62 | 56.71 |
| TR | Turkish | Çolakoğlu et al. (2019) | 7 949 | ✓ | ✓ | 37.02 | 14.53 |
| TR-DE | Turkish-German | van der Goot and Çetinoğlu (2021) | 16 546 | ✓ | ✓ | 24.14 | 22.09 |

Table 1: The MultiLexNorm datasets and their properties – number of words, whether words are split/merged, if capitalization is corrected, relative number of normalized words and performance of the most-frequent-replacement baseline (Section 5).

able dumps of Wikipedia.[2] We remove any lines shorter than 32 characters or ending with a colon (to remove titles), segment the lines into sentences with Stanza (Qi et al., 2020) and tokenize with `CMU-ARK tokenizer`[3] (a tool standardly used for tokenizing text from Twitter).

### 3.3 Fine-tuning

The final fine-tuning of the pre-trained model can be straightforwardly performed on the training data (so called `base` fine-tuning). However, we also consider fine-tuning on a mix of both authentic training data and the synthetic data (with 1:1 ratio) in order to avoid overfitting (`mixed` fine-tuning).

### 3.4 Inference

To predict the normalized sentences during inference, each word-token is processed independently, similarly to training (Figure 1). The ByT5 decoder autoregressively generates each token, either greedily or via beam search.[4] When using the beam search, we generate multiple candidate sequences, and the beam search automatically assigns prediction scores to each of them. These can then be used to aggregate predictions from multiple models in an ensemble.

---

[2] https://dumps.wikimedia.org/backup-index.html
[3] https://github.com/myleott/ark-twokenize-py
[4] For simplicity, we use the default settings of the `greedy_search` and `beam_search` procedures from the Hugging Face library, apart from allowing longer sequences and varying the number of beams.

## 4 Experiments

The MultiLexNorm shared task consists of 12 social media datasets in 11 languages, including two code-switching datasets. All these datasets are based on Twitter data, with the Danish and the Dutch ones also including data from other sources. The datasets, their original sources and some of their properties are listed in Table 1.

We train an independent model for every dataset. The pre-training starts from the *small* variant of ByT5, which contains 300M parameters. We keep all hyperparameters of the model unchanged, including a dropout rate of 10%. The training employs batch size 128 and the AdaFactor optimizer (Shazeer and Stern, 2018) to decrease memory usage.

The pre-training is performed using the synthetic data described in Section 3.2, for at least 100 000 optimizer steps.[5] We utilize inverse square root decay with peak learning rate of $5 \cdot 10^{-4}$ and 4 000 warm-up steps.

The fine-tuning phase is carried out with a constant learning rate $1 \cdot 10^{-4}$ for 50 epochs. We consider two training data configurations:

1. Using only MultiLexNorm training data, respecting the train/dev split (using 10% of training data as development set if there is none).

2. Because the development set, if present, is quite large (usually circa 30% of the training data), we also consider training on development data. Specifically, we concatenate train-

---

[5] We stop the training only after a full epoch, when at least 100k steps were performed.

ing and development data, if any, and take only 3% of the data as development set (just to detect errors, because evaluation on such small set is considerably noisy).

Our competition model is trained on the combined dataset (the latter option), and the synthetic data is mixed in (`mixed` fine-tuning from Section 3.3) if it improves development performance over the `base` fine-tuning.[6]

# 5 Results

The MultiLexNorm participants were allowed to submit two runs. The first run of our team *ÚFAL* is a single model, while the second run is an ensemble of 4 models.[7] We perform ensembling by considering for each word and each model 16 replacements including their probabilities (using a beam-search decoder), and producing the replacement with the highest average probability.

Apart from the participant systems, several baselines are also evaluated: *LAI* (leave-as-is), *MFR* (most frequent replacement based on the training data) and the *MoNoise* tool (van der Goot, 2019).

## 5.1 Intrinsic Evaluation

The intrinsic evaluation is performed using the *Error Reduction Rate* (ERR), which is word-level accuracy normalized to the number of replacements in the dataset. Formally, if we denote the system that does not normalize any word as *leave-as-is*, we can define ERR as

$$ERR = \frac{accuracy_{system} - accuracy_{leave\text{-}as\text{-}is}}{1.0 - accuracy_{leave\text{-}as\text{-}is}}.$$

The final ranking is determined by the ERR macro-averaged over all datasets.

The MultiLexNorm intrinsic evaluation results are provided in Table 2. Our system achieves the best performance by a wide margin – the single model achieves 66.2% macro-averaged ERR, and the model ensemble even a percent point more, 67.3%. That is 13.7% higher than the second-best result of 53.6%. Our ensemble model reaches best results on all datasets, except for Danish, where our single model is better, and for Spanish, where it is outperformed by other system.

## 5.2 Extrinsic Evaluation

To evaluate the effect of lexical normalization on downstream applications, MultiLexNorm considers dependency parsing. First, dependency parsing models are trained using the MaChAmp parser (van der Goot et al., 2021b) on several treebanks from Universal Dependencies 2.8 (Zeman et al., 2021). Treebanks with formal style are used (i.e., not data from social networks), specifically German-GSD, English-EWT, Italian-ISDT and Turkish-IMST. Then, MultiLexNorm participant systems are used to normalize 7 social-media treebanks, which are then parsed using the described parsing models and evaluated using the label attachment score (LAS) metric. For details, please see the MultiLexNorm shared task overview paper.

The results of the MultiLexNorm extrinsic evaluation are presented in Table 3. Our system also achieves the best performance in the overall macro-average and in 4 out of the 7 treebanks. Generally, the LAS score differences are much smaller than the ERR intrinsic evaluation metric, but the rankings in both evaluations show a lot of similarity. A notable difference is the MFR baseline, which performs remarkably well in the extrinsic evaluation.

## 5.3 Ablation Study

To quantify the effect of various hyperparameters of our system, Table 4 presents intrinsic evaluation of several ablation experiments.

**The Foundation Model** We compare the *small* variants of mT5 and ByT5 foundation models when only fine-tuning (and not pre-training on synthetic data) is used. In this setting, the ByT5 model reaches substantially better results (59.2% average ERR compared to 33.6%). We therefore did not experiment further with the mT5 model.[8]

**Pre-training and Fine-tuning Phases** The pre-training phase considerably improves results, reaching 64.8% ERR compared to 59.2% ERR without pre-training. We also evaluate the model after the pre-training phase only – the resulting 31.3% ERR is quite low, worse than the MFR baseline and most submitted systems.

**Fine-tuning Data** First, we consider the effect of fine-tuning purely on the MultiLexNorm training data (`base` fine-tuning in Table 4), mixing in the

---

[6]Ablations of all training data configurations are evaluated in Section 5.3.
[7]The ensemble models start from a single pre-trained checkpoint and only the fine-tuning is independent.

[8]Preliminary experiments on synthetic pre-training of English and Italian also demonstrated considerably worse results of mT5 compared to ByT5.

| Team | Average | DA | DE | EN | ES | HR | ID-EN | IT | NL | SL | SR | TR | TR-DE |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **ÚFAL (ensemble)** | **67.30** | 68.7 | **66.2** | **75.6** | 59.2 | **67.7** | **67.2** | **47.5** | **63.6** | **80.1** | **74.6** | **68.6** | **68.6** |
| **ÚFAL (single)** | 66.21 | **70.2** | 65.7 | 73.8 | 55.9 | 67.3 | 66.2 | 42.6 | 62.7 | 79.8 | 73.5 | **68.6** | 68.2 |
| HEL-LJU | 53.58 | 56.6 | 59.8 | 62.0 | 35.5 | 56.2 | 55.3 | 35.6 | 45.9 | 67.0 | 66.4 | 51.2 | 51.2 |
| *MoNoise* | *49.02* | *51.3* | *47.0* | *74.3* | *45.5* | *52.6* | *59.8* | *21.8* | *49.5* | *61.9* | *59.6* | *28.2* | *36.7* |
| TrinkaAI[*] | 43.75 | 45.9 | 47.3 | 66.0 | **61.3** | 41.3 | 56.4 | 15.8 | 45.7 | 59.5 | 44.5 | 15.5 | 25.8 |
| thunderml[*] | 43.44 | 46.5 | 46.6 | 64.1 | 60.3 | 40.1 | 59.1 | 11.9 | 44.0 | 59.3 | 44.5 | 15.9 | 29.0 |
| team | 40.70 | 48.1 | 46.1 | 63.7 | 21.0 | 40.4 | 59.3 | 13.9 | 43.7 | 60.6 | 46.1 | 15.9 | 29.7 |
| learnML | 40.30 | 40.5 | 43.7 | 61.6 | 56.5 | 38.1 | 56.2 | 5.9 | 42.8 | 58.2 | 40.0 | 14.4 | 25.7 |
| maet | 40.05 | 48.1 | 46.1 | 63.9 | 21.0 | 40.4 | 59.3 | 5.9 | 43.7 | 60.6 | 46.1 | 15.9 | 29.7 |
| *MFR* | *38.37* | *49.7* | *32.1* | *64.9* | *25.6* | *36.5* | *61.2* | *16.8* | *37.7* | *56.7* | *42.6* | *14.5* | *22.1* |
| CL-MoNoise | 12.05 | 7.3 | 16.5 | 4.1 | 5.0 | 26.4 | 2.4 | 0.0 | 16.2 | 8.8 | 20.1 | 17.6 | 20.2 |
| BLUE | 6.73 | 49.7 | -1.9 | 26.8 | -9.4 | -10.1 | -7.2 | -31.7 | -2.1 | -1.0 | 42.6 | 10.0 | 15.0 |
| *LAI* | *0.00* | *0.0* | *0.0* | *0.0* | *0.0* | *0.0* | *0.0* | *0.0* | *0.0* | *0.0* | *0.0* | *0.0* | *0.0* |
| MaChAmp | -21.25 | -88.9 | -93.4 | 51.0 | 25.4 | 42.6 | 39.5 | -312.9 | 1.5 | 56.8 | 39.4 | -12.7 | -3.4 |

Table 2: The results of MultiLexNorm intrinsic evaluation. Each team could submit two systems, we show the best of the two. More detailed results are available in the overview paper (van der Goot et al., 2021a) and in the participant papers: HEL-LJU (Scherrer and Ljubešić, 2021), TrinkaAI (Kubal and Nagvenkar, 2021), CL-MoNoise (van der Goot, 2021) and BLUE (Bucur et al., 2021). [*] denotes late submissions.

| Team | Average | DE tweede | EN aae | EN monoise | EN tweebank2 | IT postwita | IT twittiro | TR iwt151 |
|---|---|---|---|---|---|---|---|---|
| **ÚFAL (ensemble)** | **64.17** | **73.6** | **62.7** | **58.6** | 59.1 | **68.3** | 72.2 | 54.7 |
| **ÚFAL (single)** | 63.98 | **73.6** | 62.2 | 57.9 | 59.0 | **68.3** | 72.2 | 54.8 |
| HEL-LJU | 63.72 | 73.5 | 60.6 | 56.2 | 60.3 | 68.1 | **72.3** | **55.0** |
| *MoNoise* | *63.44* | *73.2* | *62.3* | *56.8* | *58.9* | *67.5* | *70.7* | *54.6* |
| *MFR* | *63.31* | *72.9* | *60.3* | *56.7* | ***60.3*** | *67.3* | *70.7* | *54.9* |
| TrinkaAI[*] | 63.12 | 72.9 | 60.2 | 56.6 | 59.9 | 67.0 | 71.1 | 54.2 |
| maet | 63.09 | 72.8 | 59.4 | 56.6 | 59.8 | 67.4 | 71.1 | 54.5 |
| team | 63.03 | 72.8 | 59.4 | 56.6 | 59.8 | 67.2 | 70.9 | 54.5 |
| thunderml[*] | 63.02 | 72.7 | 59.6 | 56.7 | 59.2 | 67.3 | 71.3 | 54.2 |
| learnML | 62.88 | 72.3 | 59.0 | 56.2 | 60.0 | 67.0 | 71.2 | 54.5 |
| CL-MoNoise | 62.71 | 72.7 | 60.9 | 55.3 | 58.5 | 66.5 | 70.1 | 55.0 |
| BLUE | 62.53 | 72.6 | 59.6 | 54.2 | 59.8 | 66.7 | 70.0 | 54.8 |
| *LAI* | *62.45* | *72.7* | *59.2* | *53.6* | *60.0* | *66.5* | *70.1* | ***55.0*** |
| MaChAmp | 61.89 | 71.3 | 60.8 | 54.6 | 58.0 | 64.7 | 69.8 | 54.1 |

Table 3: The results of MultiLexNorm extrinsic evaluation through dependency parsing evaluated via label attachment score (LAS). Each team could submit two systems, we show the best of the two. [*] denotes late submissions.

synthetic data with 1:1 ratio (`mixed` fine-tuning), or selecting the best option according to the development data (`best` fine-tuning). The results reveal that, unfortunately, our strategy of choosing the best variant based on development performance is actually worse than the pure `base` fine-tuning. On the other hand, training also on the development data improves the performance substantially (from 64.8% to 66.2% average ERR).

**Beam-search Decoding** Using beam-search de-

coding with a beam size 16 has virtually no effect compared to greedy decoding (66.21% average ERR for both options). We hypothesize there is no difference because we generate each normalization independently, so greedy decoding on the *small* target sequences recovers the optimal solution with very high probability. In practice, it is therefore enough to utilize greedy decoding and avoid the higher runtime requirements of a beam search.

**Ensembling** Finally, utilizing an ensemble of

| Treebank | Foundation | mT5 | ByT5 | ByT5 | ByT5 | ByT5 | ByT5 | ByT5 | ByT5 | ByT5 |
|---|---|---|---|---|---|---|---|---|---|---|
| | Pre-training | ✗ | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| | Fine-tuning | base | base | ✗ | base | mixed | best | best | best | best |
| | Training Data | train | train | train | train | train | train | trn+dev | trn+dev | trn+dev |
| | Beam size | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 16 | 16 |
| | Ensemble | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | 4 |
| **Average** | | 33.62 | 59.23 | 31.28 | 64.88 | 63.52 | 64.77 | 66.21 | 66.21 | **67.30** |
| Danish | | 31.65 | 67.41 | 49.37 | 65.82 | 67.72 | 65.82 | **70.25** | **70.25** | 68.67 |
| German | | 42.91 | 59.35 | 49.10 | 63.40 | 62.50 | 63.40 | 65.77 | 65.65 | **66.22** |
| English | | 61.27 | 70.40 | 40.15 | 73.28 | 72.68 | 73.28 | 73.88 | 73.80 | **75.60** |
| Spanish | | -0.21 | 43.87 | 9.15 | 57.59 | 56.96 | 56.96 | 55.93 | 55.93 | **59.25** |
| Croatian | | 38.11 | 55.15 | 43.31 | 63.18 | 63.03 | 63.03 | 67.29 | 67.29 | **67.74** |
| Indonesian-English | | 50.86 | 63.75 | -3.95 | 65.12 | 63.92 | 63.92 | 66.15 | 66.15 | **67.18** |
| Italian | | -7.92 | 43.56 | -12.87 | 46.53 | 35.64 | 46.53 | 42.57 | 42.57 | **47.52** |
| Dutch | | 43.18 | 55.88 | 43.45 | 62.03 | 62.30 | 62.03 | 62.70 | 62.70 | **63.58** |
| Slovenian | | 56.48 | 71.62 | 57.21 | 78.08 | 77.14 | 78.08 | 79.89 | 79.85 | **80.07** |
| Serbian | | 43.29 | 60.99 | 59.22 | 71.10 | 71.83 | 71.83 | 73.42 | 73.55 | **74.59** |
| Turkish | | 11.82 | 59.80 | 21.79 | 66.55 | 65.03 | 66.55 | 68.41 | **68.58** | **68.58** |
| Turkish-German | | 31.99 | 58.98 | 19.46 | 65.82 | 63.45 | 65.82 | 68.27 | 68.19 | **68.62** |

Table 4: The intrinsic evaluation of the ablation experiments – we consider various foundation models, whether pre-training is performed, what data do we use for fine-tuning, the decoder beam size and finally whether an ensemble of models is used.

| GPU | Batch size | Words per sec | Slowdown |
|---|---|---|---|
| GeForce RTX 3090 | 1 | 15.45 | 5.92× |
| GeForce RTX 3090 | 2 | 28.05 | 3.26× |
| GeForce RTX 3090 | 4 | 42.15 | 2.17× |
| GeForce RTX 3090 | 8 | 57.55 | 1.59× |
| GeForce RTX 3090 | 16 | 84.19 | 1.09× |
| GeForce RTX 3090 | 32 | 90.20 | 1.01× |
| GeForce RTX 3090 | 64 | 88.01 | 1.04× |
| GeForce RTX 3090 | 128 | 91.42 | 1× |
| GeForce RTX 3090 | 256 | 87.77 | 1.04× |
| GeForce RTX 3090 | 512 | 78.94 | 1.16× |
| GeForce RTX 1080 Ti | 128 | 55.69 | 1.64× |
| GeForce RTX 2080 Ti | 128 | 58.19 | 1.57× |
| Quadro P5000 | 128 | 43.88 | 2.08× |
| Quadro RTX 5000 | 128 | 66.81 | 1.37× |

Table 5: Inference speed of greedy decoding, measured on all English evaluation data (56 999 words in total) as words per second for various GPUs and batch size.

4 models improves the performance by a percent point from 66.2% average ERR to 67.3%.

### 5.4 Inference Speed

To inspect the runtime performance of our model, we measure the inference speed of a single model using greedy decoding on all English evaluation data (56 999 words in total). The results for various batch sizes and GPUs are listed in Table 5. Overall, with a batch size of 128 our model processes 43-91 words per second, depending on a GPU used. For comparison, the MoNoise system is reported to normalize 29-62 words/sec without a GPU, based on candidate filtering (van der Goot, 2019, Table 3).

### 6 Conclusion

We presented the winning system of the W-NUT 2021: Multilingual Lexical Normalization (Multi-LexNorm) shared task, which is based on a ByT5 foundation model. In intrinsic evaluation, the system performance is superior by a very wide margin, and our system also delivers the best performance in extrinsic evaluation. We release both the source code at https://github.com/ufal/multilexnorm2021 and the fine-tuned models at https://huggingface.co/ufal.

In future work, we would like to change the model architecture to encode input sentences only once, either by decoding the whole sentence, or

by separating all input words with different sentinel tokens and then decoding individual words by initializing the decoder with the corresponding sentinel tokens.

## Acknowledgements

## References

Inaki Alegria, Nora Aranberri, Víctor Fresno, Pablo Gamallo, Lluis Padró, Inaki San Vicente, Jordi Turmo, and Arkaitz Zubiaga. 2013. Introducción a la tarea compartida Tweet-Norm 2013: Normalización léxica de tuits en Español. In *Tweet-Norm@ SEPLN*, pages 1–9.

Abhijeet Awasthi, Sunita Sarawagi, Rasna Goyal, Sabyasachi Ghosh, and Vihari Piratla. 2019. Parallel iterative edit models for local sequence transduction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4260–4270, Hong Kong, China. Association for Computational Linguistics.

Timothy Baldwin, Marie Catherine de Marneffe, Bo Han, Young-Bum Kim, Alan Ritter, and Wei Xu. 2015. Shared tasks of the 2015 workshop on noisy user-generated text: Twitter lexical normalization and named entity recognition. In *Proceedings of the Workshop on Noisy User-generated Text*, pages 126–135, Beijing, China. Association for Computational Linguistics.

Anab Maulana Barik, Rahmad Mahendra, and Mirna Adriani. 2019. Normalization of Indonesian-English code-mixed Twitter data. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, pages 417–424, Hong Kong, China. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri Chatterji, Annie Chen, Kathleen Creel, Jared Quincy Davis, Dora Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Kohd, Mark Krass, Ranjay Krishna, Rohith Kuditipudi, Ananya Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir Mirchandani, Eric Mitchell, Zanele Munyikwa, Suraj Nair, Avanika Narayan, Deepak Narayanan, Ben Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, Julian Nyarko, Giray Ogut, Laurel Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghunathan, Rob Reich, Hongyu Ren, Frieda Rong, Yusuf Roohani, Camilo Ruiz, Jack Ryan, Christopher Ré, Dorsa Sadigh, Shiori Sagawa, Keshav Santhanam, Andy Shih, Krishnan Srinivasan, Alex Tamkin, Rohan Taori, Armin W. Thomas, Florian Tramèr, Rose E. Wang, William Wang, Bohan Wu, Jiajun Wu, Yuhuai Wu, Sang Michael Xie, Michihiro Yasunaga, Jiaxuan You, Matei Zaharia, Michael Zhang, Tianyi Zhang, Xikun Zhang, Yuhui Zhang, Lucia Zheng, Kaitlyn Zhou, and Percy Liang. 2021. On the opportunities and risks of foundation models.

Ana-Maria Bucur, Adrian Cosma, and Liviu P. Dinu. 2021. Sequence-to-sequence lexical normalization with multilingual transformers. In *Proceedings of the 7th Workshop on Noisy User-generated Text (W-NUT 2021)*, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Talha Çolakoğlu, Umut Sulubacak, and Ahmet Cüneyd Tantuğ. 2019. Normalizing non-canonical Turkish texts using machine translation approaches. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 267–272, Florence, Italy. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Tomaž Erjavec, Darja Fišer, Jaka Čibej, Špela Arhar Holdt, Nikola Ljubešić, and Katja Zupan. 2017. CMC training corpus Janes-Tag 2.0. Slovenian language resource repository CLARIN.SI.

Roman Grundkiewicz, Marcin Junczys-Dowmunt, and Kenneth Heafield. 2019. Neural grammatical error correction systems with unsupervised pre-training on synthetic data. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 252–263, Florence, Italy. Association for Computational Linguistics.

Divesh Kubal and Apurva Nagvenkar. 2021. Multi-lingual sequence labeling approach to solve lexical normalization. In *Proceedings of the 7th Workshop on Noisy User-generated Text (W-NUT 2021)*, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Nikola Ljubešić, Tomaž Erjavec, Maja Miličević, and Tanja Samardžić. 2017a. Croatian Twitter training corpus ReLDI-NormTagNER-hr 2.0. Slovenian language resource repository CLARIN.SI.

Nikola Ljubešić, Tomaž Erjavec, Maja Miličević, and Tanja Samardžić. 2017b. Serbian Twitter training corpus ReLDI-NormTagNER-sr 2.0. Slovenian language resource repository CLARIN.SI.

Ismini Lourentzou, Kabir Manghnani, and ChengXiang Zhai. 2019. Adapting sequence to sequence models for text normalization in social media. In *International Conference on Web and Social Media*. AAAI.

Benjamin Muller, Benoit Sagot, and Djamé Seddah. 2019. Enhancing BERT for lexical normalization. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, pages 297–306, Hong Kong, China. Association for Computational Linguistics.

Kostiantyn Omelianchuk, Vitaliy Atrasevych, Artem Chernodub, and Oleksandr Skurzhanskyi. 2020. GECToR – grammatical error correction: Tag, not rewrite. In *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 163–170, Seattle, WA, USA → Online. Association for Computational Linguistics.

Barbara Plank, Kristian Nørgaard Jensen, and Rob van der Goot. 2020. DaN+: Danish nested named entities and lexical normalization. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6649–6662, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A Python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Sascha Rothe, Jonathan Mallinson, Eric Malmi, Sebastian Krause, and Aliaksei Severyn. 2021. A simple recipe for multilingual grammatical error correction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 702–707, Online. Association for Computational Linguistics.

Yves Scherrer and Nikola Ljubešić. 2021. Sesame Street to Mount Sinai: BERT-constrained character-level Moses models for multilingual lexical normalization. In *Proceedings of the 7th Workshop on Noisy User-generated Text (W-NUT 2021)*, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Youri Schuur. 2020. Normalization for Dutch for improved pos tagging. Master's thesis, University of Groningen.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. *CoRR*, abs/1804.04235.

Uladzimir Sidarenka, Tatjana Scheffler, and Manfred Stede. 2013. Rule-based normalization of German Twitter messages. In *Proc. of the GSCL Workshop Verarbeitung und Annotation von Sprachdaten aus Genres internetbasierter Kommunikation*.

Rob van der Goot. 2019. MoNoise: A multi-lingual and easy-to-use lexical normalization tool. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 201–206, Florence, Italy. Association for Computational Linguistics.

Rob van der Goot. 2021. CL-MoNoise: Cross-lingual lexical normalization. In *Proceedings of the 7th Workshop on Noisy User-generated Text (W-NUT 2021)*, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Rob van der Goot and Özlem Çetinoğlu. 2021. Lexical normalization for code-switched data and its effect on POS tagging. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Association for Computational Linguistics.

Rob van der Goot, Alan Ramponi, Tommaso Caselli, Michele Cafagna, and Lorenzo De Mattei. 2020. Norm it! lexical normalization for Italian and its downstream effects for dependency parsing. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 6272–6278, Marseille, France. European Language Resources Association.

Rob van der Goot, Alan Ramponi, Arkaitz Zubiaga, Barbara Plank, Benjamin Muller, Iñaki San Vicente Roncal, Nikola Ljubešić, Özlem Çetinoğlu, Rahmad Mahendra, Talha Çolakoğlu, Timothy Baldwin, Tommaso Caselli, and Wladimir Sidorenko. 2021a. MultiLexNorm: A shared task on multilingual lexical normalization. In *Proceedings of the 7th Workshop on Noisy User-generated Text (W-NUT 2021)*, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Rob van der Goot, Ahmet Üstün, Alan Ramponi, Ibrahim Sharaf, and Barbara Plank. 2021b. Massive choice, ample tasks (MaChAmp): A toolkit for multi-task learning in NLP. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 176–197, Online. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Wenhan Xiong, Jiawei Wu, Hong Wang, Vivek Kulkarni, Mo Yu, Shiyu Chang, Xiaoxiao Guo, and William Yang Wang. 2019. TWEETQA: A social media focused question answering dataset. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5020–5031, Florence, Italy. Association for Computational Linguistics.

Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. 2021a. Byt5: Towards a token-free future with pre-trained byte-to-byte models.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021b. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online. Association for Computational Linguistics.

Daniel Zeman, Joakim Nivre, et al. 2021. Universal dependencies 2.8. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.