Lexical Normalization Using Generative Transformer Model (LN-GTM)

Mohamed Ashmawy1, Mohamed Waleed Fakhr2, and Fahima A. Maghraby3

1College of Computing and Information Technology, Arab Academy for Science & Technology

2College of Computer Engineering Arab Academy for Science & Technology

3College of Computing and Information Technology, Arab Academy for Science & Technology

Corresponding author: Mohamed Ashmawy (e-mail: ashmawyonline@gmail.com).

# Abstract

Lexical Normalization (LN) aims to normalize a nonstandard text to a standard text. This problem is of extreme importance in natural language processing (NLP) when applying existing trained models to user-generated text on social media. Users of social media tend to use non-standard language. They heavily use abbreviations, phonetic substitutions, and colloquial language. Nevertheless, most existing NLP-based systems are often designed with the standard language in mind. However, they suffer from significant performance drops due to the many out-of-vocabulary words found in social media text. In this paper, we present a new (LN) technique by utilizing a transformer-based sequence-to-sequence (Seq2Seq) to build a multilingual characters-to-words machine translation model. Unlike the majority of current methods, the proposed model is capable of recognizing and generating previously unseen words. Also, it greatly reduces the difficulties involved in tokenizing and preprocessing the nonstandard text input and the standard text output. The proposed model outperforms the winning entry to the Multilingual Lexical Normalization (MultiLexNorm) shared task at W-NUT 2021 on both intrinsic and extrinsic evaluations.

# **Keywords** Lexical Normalization, NLP, Transformers, Machine Translation

# **Introduction:**

Every day, social media generates a large amount of textual data, and it has been demonstrated that this data can be useful in a variety of fields, including healthcare [1], psychology [2], politics [4], crisis management [5], and marketing [6]. Natural Language Processing (NLP) systems can play a significant role in these fields and assist in advancing the effectiveness and efficiency of the tasks related to the field by utilizing the unprecedented stream of textual data generated from social media platforms [7] [8] [9]. These systems typically perform worse due to the noisy nature of social media writing [10], caused by users' propensity to employ nonstandard terms [11] and [12]. When dealing with noisy or non-standard text, NLP models tuned for tasks with a noisy and inconsistent format but trained on clean or curated data have demonstrated significant challenges. A potential solution for this problem is normalization [13]. The W-NUT 2021 Multilingual Lexical Normalization (MultiLexNorm) [14] aims to assess lexical normalization systems on 12 social media datasets in 11 languages, including two datasets for codeswitching. The evaluation process includes both internal and extrinsic components; the latter is quantified using dependency parsing on the normalized data.

There are several different types of lexical errors that can be found in social media-generated text that led to a tokenization problem in most (NLP) systems. One lexical error is a typographical error, or the mistyping of user input (e.g., bidge →bridge). Another common error is the missing apostrophe, which is usually ignored by social media users (e.g., im→i'm). There are also "split" errors, when splitting one word into many words or merge errors when joining many words into one big word, both of which happen unintentionally (e.g., pre order→preorder, screen shot→screenshot). One further error is phrasal abbreviation, which happens when the user uses the first letters of many words to create a single word to represent all of them (e.g., lol → laughing out loud). One more error is the repetition error, which is often used by social media users to emphasize a word or feeling by repeating letters (e.g., soooo →so). Another type of error is shortening, which occurs when a user writes a word without its vowels or without the final syllable or letter (e.g., pls please, rmx remix). The last error is transformation error, which occurs when the social media user replaces similar letters or ignores word endings or (e.g., hackd → hacked, wateva → whatever) [15]. Table 1 demonstrates examples of common lexical error types; the main observation is that in the majority of error types in nonstandard words, the words are altered by adding, deleting, or replacing characters. Recent lexical normalization methods have focused on word-level [16], sub-word level [17], syllable-level [18], or sentence level [19]. Based on the previous observation, we focus on characters at word-level's lexical normalization. To do that, we investigated a method to translate a sequence of characters into a sequence of words. In machine translation, a sequence of the input tokens is converted to a sequence of the output tokens, where the length of the input sequence may differ from the length of the output sequence, and the relations between the input and output tokens vary from one to one, one to many, many to one, and many to many. This problem is also known as alignment [20]. The same alignment problem applies to characters-to-words translation for the lexical normalization task: a sequence of characters can be translated to one word (e.g., bidge →bridge) or to multiple words (e.g., lol → laughing out loud).

**Table 1**: Common lexical error types found in social media text [15].

|  |  |
| --- | --- |
| Category | Example |
| Typographical error | throwg → throw |
| Missing apostrophe | doesn’t → doesn't |
| Split | screen shot → screenshot |
| Merge | ima → i'm going to |
| Phrasal abbreviation | lol → laughing out loud |
| Repetition | soooo → so |
| Shortening | pls → please |

Fortunately, there is plenty of research on machine translation with remarkable results. We chose the state-of-the-art transformer [21] to build a characters-to-words translation model. One idea was to use a character-based word representation, meaning that the translation will be done at the word or sub-word level, and we are going to lose any character-level relation information. Rather than words-to-words or subword-to-subword translation, the transformer-based model is used for aligning and translating a sequence of characters into a sequence of words. The encoder transformer accepts input words that are tokenized into a sequence of character tokens, and the decoder transformer accepts word tokens besides the encoded characters and outputs a sequence of word tokens, with a small tweak by using an extra token added to the training labels to classify the input. A token with value (oov) for out-of-vocabulary samples and a token with value (IV) for in-vocabulary samples. This way, the model uses the token prediction for both translating and classifying the input character sequence. The translation method acts as a denoising autoencoder and tries to construct a word or more from a given sequence of characters. The main contribution of this study is a lexical normalization generative model based on character-to-word translation. We also demonstrated how a generative model can be used for classification using special tokens.

# **Related Research:**

A lot of research has been conducted on lexical normalization tasks with different approaches. Most of the previous work took a two-phase approach: first, it generated a set of normalization candidates, and then it selected the best of the candidates as the normalization results. One of the first attempts to use the two-phase approach was to use a statistical machine translation model for character-level translation to generate candidates for the first phase, and in the second phase, a domain-specific language model is used for prediction [22]. Compared to word-level translation, character-level translation is more robust for unseen words. Another model was to first generate a set of normalization candidates for words, employ a Support Vector Machine (SVM) for unnormalized word detection and classification, and then an n-gram lookup for candidate selection [23]. Another character-focused model used two-step statistical machine translation [24], first detecting non-standard words using conditional random fields (CRF) sequence labeling, then translating the character sequence to a phonetic sequence, and finally translating this new sequence to words, with the words segmented based on their phonetic meaning, symbol or pronunciation improves the characters' alignment with parts of a word. Another promising two-phase approach is based on contextual graph random walks [25]. The model generates a bipartite graph with contexts on the left and unnormalized and normalized words on the right. The edges are the co-occurrence counts of a word and its context, and Markov Random Walks to locate pairings of unnormalized and normalized words that may be considered normalization equivalency. Rather than the supervised methods, [26] proposed an unsupervised statistical model for text normalization. They used edit-distance, the longest-common sub-sequence, and word pair co-occurrence as features to define a log-linear model that describes the relationship between normalized and unnormalized words, and they trained it using a unique sequential Monte Carlo algorithm. Another approach was to use the Hidden Markov Model (HMM) [27], in which each standard and nonstandard word was first divided into orthographic syllable segments. In order to do this, characters were changed into sounds, and the word-to-word transition was handled using the standard (HMM). The HMM has four layers in total. The nonstandard words make up the first layer, followed by the standard word syllables in the second layer, the standard word syllables in the third layer, and the standard words in the last layer. In [28] MoNoise model was introduced. They used a range of techniques, such as Word2Vec, Aspell, and the feature-based Random Forest, for candidate generation and ranking. The model obtained state-of-the-art results with an F1-Score of 86.39%. One other unsupervised method first generated a set of candidate morphological variations and a similarity graph of morphological variants, clustered the morphological variants, and then searched for the minimum edit distance within the clusters to obtain the normalized version [29]. Most of the models mentioned required hand-crafted features and also didn't provide a way to handle unseen abbreviations or any other forms of word substitution used on social media platforms.

One of the earliest attempts to employ a deep neural network to tackle (LN) problem was [17], which demonstrated the importance of a pre-trained language model and enhanced the Bidirectional Encoder Representations from Transformers for Language Understanding (BERT) by adding word piece tokenization and word piece alignment. Their performance on the lexnorm2015 dataset yielded an F1-score of 79.28%. [30] uses mBART [25] to translate a full sentence with unnormalized tokens to a normalized one. The main advantage of using mBart is that it can scale to multiple languages without increasing computational demands. The model obtained an average Error Reduction Rate (ERR) across languages of 10.65, where (ERR) is a proposed evaluation metric that can be used to compare the performance of systems across multiple datasets [14]. ÚFAL [31] achieved 67.3% (ERR) on the Multilingual Lexical Normalization (MultiLexNorm) shared task at W-NUT 2021 [14] by training the multilingual byte-level generative language model ByT5 [32] on synthetic data and subsequently fine-tuning it. Unlike the previous research, the aim of this study is to find a small yet generative model to solve the (LN) task. We investigate a word-to-character translation as remedy for lexical normalization solution since various word normalizations frequently work independently of one another while there is a major reliance on word characters while normalizing.

# **Background**

A typical neural machine translation system consists of an encoder that converts all input sequences to a fixed-length representation, a decoder that generates the output sequence using the encoded fixed-length representation, and an attention layer that addresses word alignment and improves translation performance. The first suggested sequence to sequence model used Long Short-Term Memory (LSTM) [33] for both encoder and decoder [34], and the first attention mechanism was additive attention [35]. It computes the attention score using a single-hidden-layer feedforward neural network with a softmax function, then generates the requisite fixed-length vector via weighted sum of all the input sequences; this fixed-length vector becomes the input to the decoder at a particular time. Multiplicative attention is another attention technique [36]. It, like additive attention, uses all encoder output with the decoder's current state and computes the score using simple matrix multiplication.

While both attention mechanisms are equivalent in complexity, multiplicative attention is faster in practice due to well optimized matrix multiplication software libraries, and it is more space efficient. Following this study, the transformer model [37] is presented; it is based on multi-head scaled dot product self-attention. The transformer allows parallelization and avoids vanishing and exploding gradients for long sequences since it analyses the full sequence at once rather than word by word.

The main building blocks of the transformer are the multi-head scaled dot product self-attention, position encoding, position-wise feed-forward network, residual connection, and layer normalization. In the following section, we discuss those building blocks in more detail.

## **Scaled Dot Product Self-Attention**

The closest example of self-attention is seen in information retrieval systems, where a user query is mapped against a set of keys to discover the best-matched data. Similarly, the self-attention function maps a query vector against a set of key vectors, then applies the Softmax function to get a score that can be applied to a value vector .

Figure 1 shows a scaled dot product self-attention with vectorization by attending on a whole sequence at once using matrix multiplication, with masking employed to prevent the attention layer from attending future output positions. Equation (1) summarizes the operations of the scaled dot product self-attention layer.

|  |  |
| --- | --- |
|  | (1) |

For a set of queries each query is mapped to a set of keys ; the output is then scaled by the model size square root , then Softmax to get the score. And then apply the score to the value matrix , The and matrices are created from the input data, with a trainable weight for each and the scale utilized to reduce performance decline when training bigger sequences. The idea is to use multiple self-attention heads with different weights to provide output with more information. As seen in Figure 2, multi-head attention is the concatenation of numerous scaled dot product self-attention. (2) demonstrates an extension for (1) that performs numerous self-attention operations and concatenates the output into a single representation.

MatMul

SoftMax

Mask (opt.)

Scale

MatMul

V

K

Q

**Fig 1**. dot product self-attention given query Q and key K and value V.

|  |  |
| --- | --- |
|  | (2) |

The is computed using (2) and all heads is concatenated and multiplied with the weight matrix .

Linear

Concat

Scaled Dot-Product Attention

Linear

Linear

Linear

Q

K

V

**Fig 2**. Multi-head scaled dot product self-attention by concatenating multiple attentions heads.

## **Position Encoding**

In recurrent neural networks, the position of a token affects the generated output due to the recurrency nature of the network.

The transformers use the positional embedding as replacement of recurrence as following:

|  |  |
| --- | --- |
|  | (3) |
|  | (4) |

Where pos is the position and i is the element index, and dmodel is the dimension of the embeddings. in equation (3) is used to compute elements of even indices and in equation (4) is used to compute the elements of odd indices.

## **Position-Wise Feed-Forword Network**

Attention layer is followed by a fully connected layer applied to each token position.

**Table 2**: The characteristics of the MultiLexNorm datasets, such as word count, word splitting and merging, capitalization correction, and percentage of normalized words, along with instances of noise in each language and their canonical forms.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Code | Language | Words | %normed | 1-N/N-1 | Caps | Example raw | Example gold | |
| HR | Croatian | 75,276 | 8.98 | - | + | interesuje | interesira | |
| DA | Danish | 11,816 | 8.66 | + | + | vaerelse | | værelse |
| NL | Dutch | 23,053 | 26.49 | + | + | wss | | waarschijnlijk |
| EN | English | 73,806 | 6.90 | + | - | wanna | | want to |
| DE | German | 25,157 | 8.90 | + | + | gepruft | | geprüft |
| ID-EN | Indonesian-English | 23,124 | 12.16 | + | - | luv | love | |
| IT | Italian | 14,641 | 7.36 | + | + | Rassegnarsi | Rassegnar si | |
| SR | Serbian | 91,738 | 7.73 | - | + | iznosenje | iznošenje | |
| SL | Slovenian | 75,276 | 15.66 | - | + | razdaje | razdalje | |
| ES | Spanish | 13,827 | 7.69 | - | - | mooovil | móvil | |
| TR | Turkish | 7,949 | 36.60 | - | + | alip | alıp | |
| TR-DE | Turkish-German | 16,546 | 24.25 | + | + | Kadinlar | Kadınlar | |

|  |  |
| --- | --- |
|  | (5) |

In equation (5) a two layer fully connected feed forward network with Rectified Linear Units (ReLU) activation [38].is the first layer weights and max function used for RelU activation function. are the last layer weights.

## **Residual Connection and Layer Normalization**

Each layer in the transformer is followed by residual connection in addition to the layer output and layer normalization [39].

|  |  |
| --- | --- |
|  | (6) |
|  | (7) |

In equations (6) and (7) is the size of the layer to be normalized, is the element of the layer, in equation (6) the mean is computed, in equation (7) the mean used to normalize the layer.

# **Dataset and Baseline**

The assessment methods and the dataset used for this task are covered in more depth in this section. The experiments conducted on the MultiLexNorm shared task [14], which includes texts in 12 languages, including Croatian, Danish, Dutch, English, German, Italian, Serbian, Slovenian, Spanish, Indonesian-English, and Turkish-German, as displayed in Table 2 with examples from the dataset’s training data, showing the characteristics of the MultiLexNorm datasets, such as word count, word splitting and merging, capitalization correction, and percentage of normalized words, along with instances of noise in each language and their canonical forms.

For several languages in the dataset, words are separated or merged (1-N/N-1 column), and capitalization (Caps column) is fixed. The dataset includes tweets in all languages; however, some languages also include information from other sources. Texts from Arto, Denmark's first significant social media platform, are also included in Danish [40], while texts from Dutch forums and SMS messaging are also included [41]. The organizers of the W-NUT workshop propose two evaluation methods: intrinsic, word-level, and extrinsic, downstream task performance. There are two baselines given, 1- Leave-As- Is (LAI): The result is identical to the initial raw input; no normalization is carried out. 2- Most- frequent-Replacement (MFR): depending on the training data, uses the most common replacement. It returns the input word if the training data does not contain the input word.

The introduced error reduction rate (ERR) in [14] served as an intrinsic assessment.

|  |  |
| --- | --- |
|  | (8) |

In (8) is the true positive and is the baseline leave-as-is accuracy. An ERR value of 0.0 indicates that a system always retains the raw words, whereas an ERR of exactly 1.0 indicates that a system is faultless. If the system normalizes more words with an incorrect form than with the proper canonical form, the ERR has a negative value.

MultiLexNorm also provided a secondary evaluation for the lexical normalization on downstream applications effectiveness. To determine whether utilizing normalized data is more performant than using the original data, A dependency parser is trained on both raw and canonical data to assess whether using normalized data is more performant than using the original data, and it is shown that lexical normalization enhances the performance for this task [14].

# **Model**

The objective of this study is to design a simple yet effective generative neural machine translation model. The model leverages the vanilla transformers-based sequence to sequence introduced in [21] for the neural machine translation problem, but with a custom preprocessing and tokenization for both input and output.

As shown in Figure 3, the input for the model is one or more words tokenized into sequence of characters, and the output is a sequence of generated words that is a normalized version of the input. We also included the actual raw data in the input to generate a feature-rich input, giving the model an opportunity to establish a useful relationship between the raw input and its characters as needed.

Processed Inputs

Raw Input

Raw Output

ima

i'm going to

[<bos>, i, m, a, <org>, ima, <eos>]

Processed Output

[<bos>, i’m, going, to, <oov>, <eos>]

**Fig 3**: example of processed input and the output of the proposed system.

To do this, we update the input with two types of data, the actual input words, and a separation token <org>, to separate the characters sequences and the actual input. So, the input sequence [l, o, l] becomes [l, o, l, <org>, lol]. To create a class-aware model. The model learns to predict one extra token with the output. This extra token will be one of two tokens; the <oov> token for words that need to be normalized, and the <iv> token for standard words. During training, the extra token is added to the output sequence label before the End-Of-Sequence token <eos> by checking the output. For example, consider the model trained to map the input sequences [ l, o, l] to [ laughing, out, loud, <oov>] and [ l, a, u, g, h, I, n, g, <space>, o, u, t, <space>, l, o, u, d] to [laughing, out, loud, <iv>]. In this way, the model can learn the difference between the (OOV) input and the (IV) input. And the inference engine looks at the last token to decide whether to use the normalized output if the predicted token is [oov] or return the same input if the token is [iv]. During the training phase, we train a single translation model for each language, all models consisting of two transformer layers with 512 embedding sizes.

As shown in Figure 4, the translation model receives a processed input sequence and output sequence, the input sequence is passed to the encoder, the encoder should generate a fixed representation of the input, the output with encoded sequence is fed to the decoder, and the decoder then attempts to generate tokens in sequential order to match the given output.

ima

[<bos>, i, m, a, <org>, ima, <eos>]

i'm going to

[<bos>, i’m, going, to, <oov>, <eos>]

Transformer

Encoder Transformer

Transformer

Decoder Transformer

Evaluate and Update Weights

**Fig 4**: illustration of training phase of the proposed system

During the inference phase, the model receives a sequence of words to normalize; the given words' language is automatically detected using a language detection library; we utilized the "langdetect" package for this task. The detected language is used to select and load the trained language model. The selected model will then normalize the input word by word. Similarly, to the training phase, the Model ignores the normalization emails, Websites, usernames, and hashtags and returns them as-is.

As illustrated in Figure 5, the proposed model generates the output from the training mode using greedy decoding. The output is supposed to end with the class token, either the <oov> token or the <iv> token. If the <oov> token appears in the output, the input is considered for normalization, and we manually replace the <oov> token from the string before returning the rest of the prediction. Otherwise, we return the input as-is.

Transformer

ima

[<bos>, i, m, a, <org>, ima, <eos>]

Greedy decoding

Has OOV class token

Return Input As IS

Encoder Transformer

Transformer

Decoder Transformer

Remove class token, Return System Output

Yes

No

**Fig 5**: the inference phase of a trained language model.

# **Results And Evaluation**

* 1. **TRAINING**:

We train a 12 model one for the 12 languages. The training is done on a personal laptop with core i5 12th CPU and RTX 1650 GPU. As shown in Table 3, all language models are trained for 5 epochs with 512 embedding size and 8 attention heads, and 2 encoder layers with 2 decoder layers.The model performs translation at the word level, but not all words in the training data are considered for training; the model ignores emails, websites, usernames, and hashtags. Table 4 shows the actual sample lines selected for training as well as the size of the source and target vocabulary. This resulted in slightly smaller training samples as well as Source and Target vocabulary. This reduces training time without affecting the final output.

**Table 3**: Training parameters for all language models

|  |  |
| --- | --- |
| Embedding | 512 |
| Heads | 8 |
| Batch Size | 8 |
| Encoder | 2 |
| Decoder | 2 |
| Epochs | 5 |

**Table 4**: The vocabulary size of the selected training samples from the original samples.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Lang | Vocab (SRC) | Vocab (TGT) | Selected lines | Original lines |
| da | 4610 | 3973 | 16442 | 17167 |
| de | 4648 | 3914 | 14977 | 16634 |
| en | 8342 | 7548 | 35203 | 37576 |
| es | 2854 | 2361 | 7189 | 7757 |
| hr | 16832 | 14713 | 54416 | 59176 |
| iden | 4562 | 3769 | 13923 | 14444 |
| it | 3693 | 3440 | 12645 | 13238 |
| nl | 4833 | 3201 | 12374 | 13288 |
| sl | 12949 | 10429 | 44944 | 49614 |
| sr | 16275 | 13891 | 56823 | 60961 |
| tr | 4757 | 3236 | 6334 | 7013 |
| trde | 6776 | 5305 | 12670 | 13573 |

**Fig 6**: the change of the training loss for each language model over training the five epochs.

Figure 6 shows the learning curve represented as loss over the five training epochs for all the languages models.

And Figure 7 presents the training time for each language. The learning progress is affected by the size and the symbols diversity in each language.

**Fig 7**: the training time for each language model in minutes.

* 1. **Intrinsic Evaluation:**

As previously mentioned, for intrinsic assessment, we utilise the Error Reduction Rate (ERR), which measures word-level accuracy normalised to the dataset's replacement rate.

The ERR macro averaged across all datasets is used to establish the final ranking. The result generated using “normEval” script provided in the dataset repository.

Table 5 shows the generated evaluation terms for each language as the output of the evaluation script. The first row shows the LAI baseline accuracy, the second and third rows show the accuracy and the ERR value generated from the evaluation script.

Table 6 shows the results of the intrinsic evaluation for the MultiLexNorm. The table shows the results of the proposed model for each language compared to multiple teams from the workshop challenge including the winning team.

By a significant margin, the proposed model outperforms the baselines. The model achieved an average ERR across languages of 88.41, corresponding to a distinct model trained for each language. That is 21.11% improvement over the second-best performance of 67.3%. Except for Croatian and Serbian, our model achieves the best results across all datasets.

* 1. **EXTRINSIC EVALUATION:**

Extrinsic evaluation aims to evaluate the effect of the trained model on downstream tasks. The workshop challenge provides 7 social-media treebanks to be normalized with the trained model for dependency parsing task using models trained on standard text.

The MaChAmp parser [42] is used to train the dependency parsing models in a standard language and normal words from a treebank from Universal Dependencies 2.8 [43]. The label attachment score (LAS) metric is used to evaluate the dependency parsing task.

In Table 7 We showcase the results for our best model on the dependency parsing downstream task from the workshop challenge on each treebank. Our model achieves the best performance of all treebanks and the overall macro-average.

## **Ablation Study**

In order to gain a better understanding of the model’s behavior, several ablation experiments conducted to evaluate the effectiveness of multiple training settings, due to the physical hardware limitation we mainly focused our experiments to the preprocessing of the input data, the batch size, and the numbers of layers. Table 8 shows intrinsic evaluation of these ablation experiments.

**Input preprocessing**: We tried to use raw character sequence as an input without our extra tokens, but we found that adding the non-standard word extra to the training input character sequence led to significantly better performance on all language models from 75.44 average ERR to 86.94. As a result, since the preprocessing was necessary, we did not conduct any further experiments without preprocessing.

**Batch size**: We tried 8 and 32 as batch size, the smaller batch size considerably improves results in all language from average 86.76 ERR to 88.41 in our best model.

**Number of Layers**: We tried 2 and 3 layers for both encoders and decoders, and we found that the 2 layers architecture achieved a good result in a short time.

Both batch size and number of layers parameters have small effects on the overall model performance compared to the preprocessing. By giving more information in the input, the model had a chance to collect more evidence to support the prediction decision.

**Table 5**: generated evaluation terms of “normEval” script after executed on each language model.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Term | da | de | en | es | hr | iden | It | nl | sl | sr | tr | trde |
| Baseline | 92.18 | 89.08 | 92.1 | 92.75 | 87.14 | 86.67 | 98.4 | 75.02 | 85.27 | 90.46 | 71.08 | 75.72 |
| accuracy | 99.63 | 99.94 | 99.21 | 99.89 | 93.71 | 99.91 | 99.9 | 99.98 | 96.42 | 96.02 | 99.88 | 99.99 |
| err | 95.24 | 99.46 | 90.06 | 98.54 | 51.09 | 99.31 | 93.75 | 99.93 | 75.69 | 58.3 | 99.58 | 99.99 |

**Table 6**: the result of intrinsic evaluation for the MultiLexNorm

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Team | avg | da | de | en | es | hr | iden | It | nl | sl | sr | tr | trde |
| LN-GTM (Ours) | **88.41** | **95.24** | **99.46** | **90.06** | **98.54** | 51.09 | **99.31** | **93.75** | **99.93** | **75.69** | 58.3 | **99.58** | **99.99** |
| ÚFAL2 | 67.3 | 67.3 | 66.2 | 75.6 | 59.2 | **67.7** | 67.2 | 47.5 | 63.6 | 80.1 | **74.6** | 68.6 | 68.6 |
| HEL-LJU-2 | 53.58 | 56.65 | 59.80 | 62.05 | 35.55 | 56.24 | 55.33 | 35.64 | 45.88 | 66.97 | 66.44 | 51.18 | 51.18 |
| MoNoise | 49.02 | 51.27 | 46.96 | 74.35 | 45.53 | 52.63 | 59.79 | 21.78 | 49.53 | 61.91 | 59.58 | 28.21 | 36.72 |
| TrinkaAI-2 | 43.75 | 45.89 | 47.30 | 65.96 | 61.33 | 41.28 | 56.36 | 15.84 | 45.74 | 59.51 | 44.52 | 15.54 | 25.77 |
| thunderml-1 | 43.44 | 46.52 | 46.62 | 64.07 | 60.29 | 40.09 | 59.11 | 11.88 | 44.05 | 59.33 | 44.46 | 15.88 | 29.01 |

**Table 7**: extrinsic evaluation results in LAS score.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Team | Avg | DE  tweede | EN  aae | EN  monoise | EN  tweebank2 | IT  postwita | IT  twittiro | TR  iwt151 |
| LN-GTM (Ours) | **66.55** | **73.92** | **64.43** | **69.33** | **61.93** | **69.74** | **72.31** | **56.11** |
| ÚFAL-2 | 64.17 | 73.58 | 62.73 | 58.57 | 59.08 | 68.28 | 72.22 | 54.74 |
| MFR | 63.31 | 72.86 | 60.32 | 56.74 | 60.31 | 67.34 | 70.72 | 54.89 |
| mBart | 62.53 | 72.57 | 59.57 | 54.20 | 59.81 | 66.74 | 69.99 | 54.84 |
| LAI | 62.45 | 72.71 | 59.21 | 53.65 | 59.99 | 66.49 | 70.06 | 55.00 |

**Table 8**: Ablation research results

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Preprocessing |  |  | **🗸** | **🗸** |
| Batch size | 8 | 32 | 8 | 8 |
| Layers | 2 | 3 | 3 | 2 |
| AVG | | 75.44 | 72.24 | 86.94 | **88.41** |
| da | | 70.94 | 70.66 | 92.35 | **95.24** |
| de | | 76.63 | 75.4 | 96.75 | **99.46** |
| en | | 96.63 | 87.85 | **90.66** | 90.06 |
| es | | 71.73 | 70.41 | 96.27 | **98.54** |
| hr | | 47.58 | 40.33 | 50.5 | **51.09** |
| iden | | 77.98 | 74.46 | 98.61 | **99.31** |
| it | | 82.78 | 81.03 | 92.82 | 93.75 |
| nl | | 87.56 | 86.7 | 99.15 | **99.93** |
| sl | | 74.26 | 71.55 | 73.01 | **75.69** |
| sr | | 56.94 | 54.8 | 57.47 | **58.3** |
| tr | | 75.58 | 71.5 | 96.37 | **99.58** |
| trde | | 86.66 | 82.2 | 99.28 | **99.99** |

# **Conclusion**

This study shows that the lexical normalization task can be viewed as a character-to-word alignment problem observed from the common lexical error types, and to solve this problem we used attention-based transformers. The proposed model is an end-to-end system trained from scratch without any pretrained Language Model. Also, the normalization process requires two steps: detecting the text to be normalized, applying the normalization, and generating the normalized text. The transformers with attention mechanisms can both detect and generate in a single process; the classification can be done with an extra label that determines the nature of the input. And the generated output can be used if the input word is classified as OOV. We used the encoder-decoder transformer model to perform character-to-word translations to solve the lexical normalization task in an end-to-end system. The character to word translation improves the performance with unseen OOV, with minimum feature engineering and a relatively small parameters size to speed up the processing. The character-to-word translation improves the performance with unseen OOV, with minimum feature engineering and a relatively small parameter size to speed up the processing.

# **References**:

[1] Pai, Rajesh & Alathur, Sreejith. (2018). Assessing mobile health applications with twitter analytics. International Journal of Medical Informatics. 113. 10.1016/j.ijmedinf.2018.02.016.

[2] Roland, Damianm & Spurr, Jesse & Cabrera, Daniel. (2017). Preliminary Evidence for the Emergence of a Health Care Online Community of Practice: Using a Netnographic Framework for Twitter Hashtag Analytics. Journal of Medical Internet Research. 19. e252. 10.2196/jmir.7072.

[3] Guntuku, Sharath Chandra & Schneider, Rachelle & Pelullo, Arthur & Young, Jami & Wong, Vivien & Ungar, Lyle & Polsky, Daniel & Volpp, Kevin & Merchant, Raina. (2019). Studying expressions of loneliness in individuals using twitter: an observational study. BMJ Open. 9. e030355. 10.1136/bmjopen-2019-030355.

[4] Bahrami, Mohsen & Findik, Yasin & Bozkaya, Burcin & Balcisoy, Selim. (2018). Twitter Reveals: Using Twitter Analytics to Predict Public Protests.

[5] A, Blanford & And, J & MacEachren, Alan & Robinson, Anthony & Pezanowski, Scott & Savelyev, Alexander & Blanford, Justine & Mitra, Prasenjit. (2011). Geo-Twitter Analytics: Applications in Crisis Management.

[6] Sheth, Jagdish. (2021). New areas of research in marketing strategy, consumer behavior, and marketing analytics: the future is bright. Journal of Marketing Theory and Practice. 29. 1-10. 10.1080/10696679.2020.1860679.

[7] Pai, Rajesh & Alathur, Sreejith. (2018). Assessing mobile health applications with twitter analytics. International Journal of Medical Informatics. 113. 10.1016/j.ijmedinf.2018.02.016.

[8] Roland, Damian & Spurr, Jesse & Cabrera, Daniel. (2017). Preliminary Evidence for the Emergence of a Health Care Online Community of Practice: Using a Netnographic Framework for Twitter Hashtag Analytics. Journal of Medical Internet Research. 19. e252. 10.2196/jmir.7072.

[9] Kudliskis, Voldis & Burden, Robert. (2009). Applying ‘what works’ in psychology to enhancing examination success in schools: The potential contribution of NLP. Thinking Skills and Creativity. 4. 170-177. 10.1016/j.tsc.2009.09.002.

[10] Eisenstein, Jacob. (2013). What to do about bad language on the internet. Proceedings of NAACL-HLT. 359-369.

[11] Kumar, Ankit & Makhija, Piyush & Gupta, Anuj. (2020). Noisy Text Data: Achilles’ Heel of BERT. 16-21. 10.18653/v1/2020.wnut-1.3.

[12] Agarwal, Sumeet & Godbole, Shantanu & Punjani, Diwakar & Roy, Shourya. (2007). How Much Noise Is Too Much: A Study in Automatic Text Classification. Proceedings - IEEE International Conference on Data Mining, ICDM. 3-12. 10.1109/ICDM.2007.21.

[13] Jacob Eisenstein. 2013. What to do about bad language on the internet. In Proceedings of the 2013 conference of the North American Chapter of the association for computational linguistics: Human language technologies, pages 359–369.

[14] Rob van der Goot, Alan Ramponi, Arkaitz Zubiaga, Barbara Plank, Benjamin Muller, Iñaki San Vicente Roncal, Nikola Ljubeši´c, Özlem Çetino˘glu, Rahmad Mahendra, Talha Çolako˘glu, 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

[15] R. van der Goot, R. van Noord, en G. van Noord, “A Taxonomy for In-depth Evaluation of Normalization for User Generated Content”, in Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018), 2018.

[16] R. van der Goot en G. van Noord, “MoNoise: Modeling Noise Using a Modular Normalization System”, Computational Linguistics in the Netherlands Journal, vol 7, bll 129–144, 2017

[17] B. Muller, B. Sagot, en D. Seddah, “Enhancing BERT for Lexical Normalization”, in Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019), 2019, bll 297–306.

[18] Ning Jin. 2015. Ncsu-sas-ning: Candidate generation and feature engineering for supervised lexical normalization. In Proceedings of the Workshop on Noisy User-generated Text, pages 87–92.

[19] Bucur, A.-M., Cosma, A., & Dinu, L. P. (2021). Sequence-to-Sequence Lexical Normalization with Multilingual Transformers. doi:10.48550/ARXIV.2110.02869

[20] D. Bahdanau, K. Cho, en Y. Bengio, “Neural Machine Translation by Jointly Learning to Align and Translate”, arXiv [cs.CL]. 2016.

[21] A. Vaswani et al., “Attention is All you Need”, in Advances in Neural Information Processing Systems, 2017, vol 30.

[22] D. Pennell en Y. Liu, “A Character-Level Machine Translation Approach for Normalization of SMS Abbreviations”, in Proceedings of 5th International Joint Conference on Natural Language Processing, 2011, bll 974–982.

[23] B. Han en T. Baldwin, “Lexical Normalisation of Short Text Messages: Makn Sens a #twitter”, in Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, 2011, bll 368–378.

[24] C. Li en Y. Liu, “Improving Text Normalization using Character-Blocks Based Models and System Combination”, in Proceedings of COLING 2012, 2012, bll 1587–1602.

[25] H. Hassan en A. Menezes, “Social Text Normalization using Contextual Graph Random Walks”, in Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2013, bll 1577–1586.

[26] Y. Yang en J. Eisenstein, “A Log-Linear Model for Unsupervised Text Normalization”, in Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, 2013, bll 61–72.

[27] K. Xu, Y. Xia, en C.-H. Lee, “Tweet Normalization with Syllables”, in Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), 2015, bll 920–928.

[28] R. van der Goot en G. van Noord, “MoNoise: Modeling Noise Using a Modular Normalization System”, Computational Linguistics in the Netherlands Journal, vol 7, bll 129–144, 2017.

[29] A. Roy, S. Ghosh, K. Ghosh, en S. Ghosh, “An Unsupervised Normalization Algorithm for Noisy Text: A Case Study for Information Retrieval and Stance Detection”, J. Data and Information Quality, vol 13, no 3, Apr 2021.

[30] A.-M. Bucur, A. Cosma, en L. P. Dinu, “Sequence-to-Sequence Lexical Normalization with Multilingual Transformers”, arXiv [cs.CL]. 2021.

[31] Samuel, David & Straka, Milan. (2021). \'UFAL at MultiLexNorm 2021: Improving Multilingual Lexical Normalization by Fine-tuning ByT5.

[32] 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.

[33] S. Hochreiter en J. Schmidhuber, “Long Short-term Memory”, Neural computation, vol 9, bll 1735–1780, 12 1997.

[34] I. Sutskever, O. Vinyals, en Q. V. Le, “Sequence to Sequence Learning with Neural Networks”, in Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2, Montreal, Canada, 2014, bll 3104–3112.

[35] D. Bahdanau, K. Cho, en Y. Bengio, “Neural Machine Translation by Jointly Learning to Align and Translate”, arXiv [cs.CL]. 2016.

[36] T. Luong, H. Pham, en C. D. Manning, “Effective Approaches to Attention-based Neural Machine Translation”, in Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, 2015, bll 1412–1421.

[37] A. Vaswani et al., “Attention is All you Need”, in Advances in Neural Information Processing Systems, 2017, vol 30.

[38] V. Nair en G. E. Hinton, “Rectified Linear Units Improve Restricted Boltzmann Machines”, in Proceedings of the 27th International Conference on International Conference on Machine Learning, Haifa, Israel, 2010, bll 807–814.

[39] J. L. Ba, J. R. Kiros, en G. E. Hinton, “Layer Normalization”, arXiv [stat.ML]. 2016.

[40] 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.

[41] Youri Schuur. 2020. Normalization for Dutch for improved pos tagging. Master’s thesis, University of Groningen.

[42] 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.

[43] 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.