Multilingual Sequence Labeling Approach to solve Lexical Normalization

Divesh Kubal
CRIMSON AI
divesh.kubal@crimsoni.ai

Apurva Nagvenkar
CRIMSON AI
apurva.nagvenkar@crimsoni.ai

Abstract

The task of converting a nonstandard text to a standard and readable text is known as lexical normalization. Almost all the Natural Language Processing (NLP) applications require the text data in normalized form to build quality task-specific models. Hence, lexical normalization has been proven to improve the performance of numerous natural language processing tasks on social media. This study aims to address the problem of Lexical Normalization by formulating the Lexical Normalization task as a Sequence Labeling problem. This paper proposes a sequence labeling approach to solve the problem of Lexical Normalization in combination with the word-alignment technique. The goal is to use a single model to normalize text in various languages. This is a shared task in "2021 The 7th Workshop on Noisy User-generated Text (W-NUT)" in which the participants are expected to create a system/model that performs lexical normalization, which is the translation of non-canonical texts into their canonical equivalents, comprising data from over 12 languages. The proposed single multilingual model achieves an overall ERR score of 43.75 on intrinsic evaluation and an overall Labeled Attachment Score (LAS) score of 63.12 on extrinsic evaluation. Further, the proposed method achieves the highest Error Reduction Rate (ERR) score of 61.33 among the participants in the shared task. This study highlights the effects of using additional training data to get better results as well as using a pre-trained Language model trained on multiple languages rather than only on one language.

1 Introduction

Information can be found in abundance on social media. In fact, social media is a significant data resource for many natural language processing (NLP) tasks. Such text is characterized by 'noise', or non-standard language filled with lexical variants, acronyms, hashtags, and mentions. Hence, the performance of conventional NLP tools on social media text, is poor as the data resources they used were standard texts. Although data extracted from social media contains a high degree of noise, the data can be extracted in huge quantities because of fast speed and casual style. Existing natural language processing techniques face numerous challenges as a result of the innovative/creative/slang language usage present on social media. Slang, acronyms, and abbreviated words are all part of this so-called new "online" language. It is demonstrated by (Liu et al., 2011; Schulz et al., 2016) that the performance of NLP tools often drops dramatically when used on social media data. The challenge can be solved by adapting the input text to a more conventional format, a process known as Lexical Normalisation. This paper focuses to solve the problem of Lexical Normalization on a word-by-word or token-by-token basis.

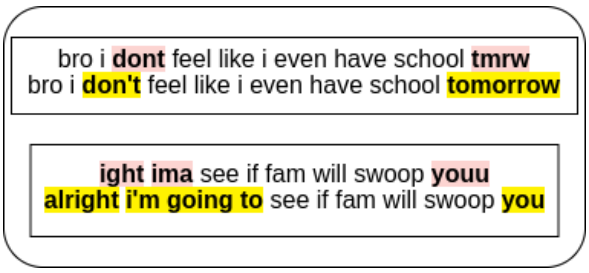

Figure 1: Examples of Lexical Normalization (English Language)

Figure 1 shows examples of un-normalized and the corresponding normalized sentences. The following languages are to be lexically normalized in this task: Croatian (hr), Danish (da), Dutch (nl), English (en), Indonesian-English (iden), German (de), Italian (it), Serbian (sr), Slovenian (sl), Spanish (es), Turkish (tr), and Turkish-German (trde). The proposed model is based on a sequence labeling-based technique in which the input tokens are in an un-normalized form and the target tokens are in normalized forms. To reduce the target labels

Proceedings of the 2021 EMNLP Workshop W-NUT: The Seventh Workshop on Noisy User-generated Text, pages 457–464
November 11, 2021. ©2021 Association for Computational Linguistics

and to make predictions faster, only those tokens for which normalization is required are added. For the tokens which do need not be normalized, their target label fixed to be a single target token (for instance 'O'). This sequence labeling model is finetuned on a pre-trained multilingual model to encompass the information on all the possible languages. Further, a post-processing layer for word-alignment is applied which further help to improve the performance in terms of Error Rate Reduction (ERR - word-level accuracy normalized).

This paper elucidates four essential points to improve lexical normalization systems.:

- the importance of treating Lexical Normalization task as a Sequence Labeling Task.
- the effect of providing the model with additional data to improve performance
- the importance of finetuning a language model pre-trained on a huge multilingual corpus
- the Effect of hyperparameter tuning by using a single model to solve the problem of lexical normalization for all the tasks.
- the effect of word-alignment post-processing layer.

2 Related Work

Previous work on normalization has been disjointed as a number of techniques have been tested against a variety of benchmarks, with a variety of assessment criteria and assumptions. In addition, the majority of normalization systems are neither open-source nor publicly accessible. To solve the problem of lexical normalization, existing systems have mainly used machine translation (Vaghani, 2020; Pennell and Liu, 2011; Ljubešic et al., 2016) or spellchecking (Han, 2014) technique. In earlier systems, the lexical normalization was performed by using the following steps:

1. Detect words that need to be normalized.
2. Generate possible candidates for the detected original un-normalized word.
3. Rank the possible candidates and select the best one.

It was later discovered by (Jin, 2015; van der Goot, 2019) that the detection step can be completely avoided by treating each original word as a candidate for normalization. At present, the state-of-the-art model for almost all the languages is MoNoise (van der Goot, 2019) which is a two-step approach. In this paper, there are different techniques for generating candidates. To get the best or correct candidate, MoNoise uses Random Forest as a classifier that takes input features generated in the first step (candidate-generation step). There are methods using sequence-to-sequence (seq2seq) based architectures (Lourentzou et al., 2019) and contextual words and phrasal embeddings (Muller et al., 2019) which achieved a performance similar to that of MoNoise, especially in the English language. Most of the work in lexical normalization is performed for the English language (Han and Baldwin, 2011; Xu et al., 2015). Nonetheless, research has been conducted on languages other than English by (van der Goot, 2019).

3 Multilingual Sequence Labeling Approach

The proposed system architecture is depicted in figure 2. The main components of Multilingual Sequence Labeling Approach to solve Lexical Normalization Problem System Architecture are as follows:

1. Data acquisition/collection.
2. Data preparation of input for the multilingual model.
3. Finetuning of transformer-based language models for the sequence labeling task of lexical normalization.
4. Hyperparameter tuning and retraining of language models.
5. Word alignment based post-processing to obtain final predictions.
6. Evaluation of development and test datasets.

3.1 Data Acquisition/Collection

The shared task provides the data in 12 languages. The data is then split into training data (train.norm) and development data (dev.norm). Essentially, the data is divided into two broad categories based on the method of assessment, intrinsic or extrinsic. In intrinsic evaluation, Error Reduction Rate

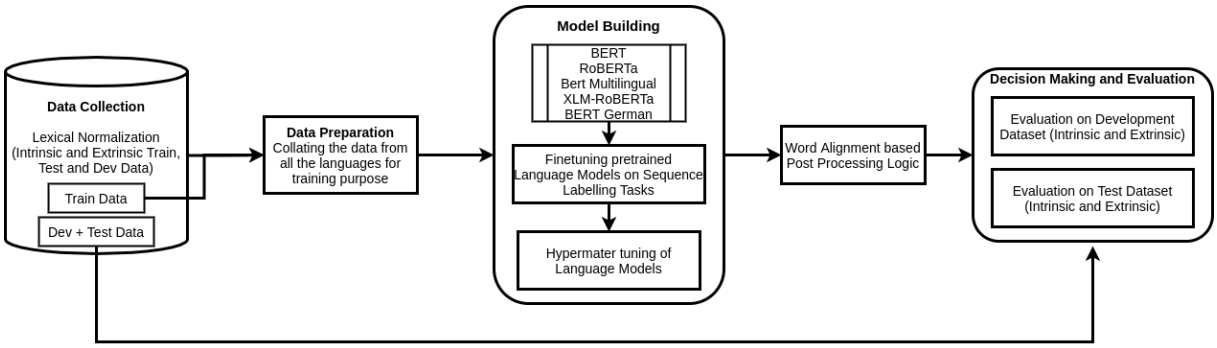

Figure 2: Multilingual Sequence Labeling Approach to solve Lexical Normalization Problem

(ERR) is used as the evaluation measure. Dependency parsing is considered extrinsic evaluation, which focuses on the impact of normalization on the quality of the parsed trees. In extrinsic evaluation, Labeled Attachment Score (LAS) is used as the evaluation measure.

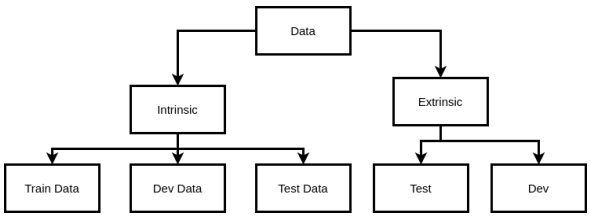

Figure 3: Overview of dataset

Figure 3 depicts the categorization of the data into intrinsic and extrinsic evaluation. The intrinsic type of data is annotated except for the test dataset which contains only the unnormalized text. On the other hand, the extrinsic type of data contains the only unnormalized text. Table 1 shows the actual number of sentences contained in each split (train and dev). The languages in table 1 are arranged in the descending order of the parallel sentences in training data. The dev dataset split is not provided for the following languages: Turkish-German, Italian, Turkish, Spanish and Danish.

Table 2 provides the count of the un-normalized sentences; they are not labeled. This data is used for the extrinsic evaluation. Finally, Table 3 shows the number of sentences used for the intrinsic evaluation of the languages provided.

3.2 DataData preparation of input for the multilingual model

This study focuses on building a single model capable of performing lexical normalization on the 12 languages provided in the shared task. Hence, the training data is prepared by concatenating the parallel sentences of normalized and un-normalized data. This is done in two phases. The first phase consists of only concatenating the training data of all languages from the intrinsic set and then evaluating the model on the intrinsic development data. This is done to finetune the trained model so that it achieves high performance on development data. This step also reduces the loss when evaluated on

Language	Train	Dev
hr	4760	1588
sl	4670	1557
sr	4138	1379
en	2360	590
de	1628	573
nl	907	308
trde	800	NA
it	593	NA
tr	570	NA
es	568	NA
iden	495	165
da	207	NA

Table 1: Number of Parallel Sentences (unnormalized - normalized) provided for different languages [Intrinsic Data]

Name	Sentences
ud-it-twittiro.test.norm.masked	3018
ud-en-aae.test.norm.masked	3322
ud-it-postwita.test.norm.masked	12796
ud-en-tweebank2.test.norm.masked	20296
ud-tr-iwt151.test.norm.masked	48200
ud-en-monoise.test.norm.masked	5738
ud-de-tweede.test.norm.masked	5261

Table 2: Number of unnormalized Sentences provided for different languages [Extrinsic Dev and Test Data]

language	Number of sentences
it	2096
iden	4531
sl	16578
es	7166
de	5665
nl	5721
tr	1782
trde	3964
sr	18496
hr	17281
en	31388
da	3939

Table 3: Number of Test Sentences provided for different languages [Intrinsic Test Data]

development data. After getting an assurance that the model performs as expected on the development data, the final model's architecture and the values of the newly finetuned hyperparameters are noted down. In the next phase, a new set of training data is generated which is the concatenation of intrinsic train + dev data. This data is used to train the architecture which achieved the best performance in the first phase. This trained model is now used to predict un-normalized sentences from the test data. Additional Spanish data was taken from shared task [1] (training data) and was used as training data. After concatenating all the data, a total of 16861 labels are generated. These labels are the unique normalization tokens/phrases for their corresponding un-normalized counterparts.

3.3 Finetuning of transformer-based language models for the sequence labeling task of lexical normalization

The proposed approach attempts to finetune already pre-trained language models. The experiments of finetuning language models on a downstream task of lexical normalization are performed on the following pre-trained language models: BERT (Devlin et al., 2018), RoBERTa (Liu et al., 2019), BERT-multilingual (Devlin et al., 2018), XLM-RoBERTa (Conneau et al., 2019), BERT-German. For the English Language, the variants of RoBERTa performed well as so with BERT (trained on an English corpus). However, these models which were pre-trained on a huge English

Hyperparameter	Value
dataloader_num_workers	0
do_lower_case	False
early_stopping_consider_epochs	False
early_stopping_delta	0
early_stopping_metric	eval_loss
early_stopping_metric_minimize	True
early_stopping_patience	3
eval_batch_size	8
evaluate_during_training	False
evaluate_during_training_steps	2000
evaluate_during_training_verbose	False
evaluate_each_epoch	True
fp16	True
gradient_accumulation_steps	1
lazy_loading	False
lazy_loading_start_line	0
learning_rate	4E-05
logging_steps	50
max_grad_norm	1
max_seq_length	256
model_class	NERModel
model_name	bert-base-multilingual-cased
model_type	bert
multiprocessing_chunksize	-1
n_gpu	1
no_cache	False
no_save	False
num_train_epochs	100
onnx	False
optimizer	AdamW
overwrite_output_dir	False
polynomial_decay_schedule_lr_end	1E-07
polynomial_decay_schedule_power	1
process_count	14
quantized_model	False
reprocess_input_data	True
save_best_model	True
save_eval_checkpoints	True
save_model_every_epoch	True
save_optimizer_and_scheduler	True
save_steps	20000
scheduler	linear_schedule_with_warmup
silent	False
skip_special_tokens	True
train_batch_size	32
use_early_stopping	False
use_multiprocessing	True
use_multiprocessing_for_evaluation	True

Table 4: Final hyperparameters list used to finetune Language models for Lexical Normalization task using Sequence Labeling)

[1] http://komunitatea.elhuyar.eus/tweet-norm/

corpus failed to achieve good performance on the other 11 languages. To make sure that this behavior is consistent, another experiment was performed wherein the unified training data from phase 1 of section 3.2 was taken and this time the finetuning was performed on the BERT-German model which is pre-trained on Open Subtitles, CommonCrawl Wikipedia dump, ParaCrawl EU Bookshop corpus, and News Crawl. This sums up to about 16GB of German text corpus and 2,350,234,427 tokens. This model with BERT-German cased is trained based on the hypothesis that it will perform well for the German language and low in other languages. This hypothesis was not rejected after computing the evaluation results. Hence, it would be a sensible choice to train different models for different languages. However, it would be expensive in terms of resource cost when these models have to be deployed in production. Further, it would have scalability issues to cater to a large number of people.

Hence, the proposed approach finetunes the BERT multilingual base model (Devlin et al., 2018). The multilingual BERT model is pre-trained on the top 104 languages that have voluminous Wikipedia data using the masked language modeling (MLM) objective. The training data for each language was collected from the complete Wikipedia dump (excluding user and discussion pages). The major challenge faced while preparing this enormous dataset of 104 languages is the data imbalance, as different languages had different amounts of data. To deal with this data imbalance, the exponentially smoothed weighting of the data is performed. For instance, let's assume that the percentage of English data is 32% (after concatenating all English data in Wikipedia). In such a case, re-normalization and sampling are applied from that distribution after exponentiating each probability by a factor S (S was chosen to be 0.7 in the paper). As a result, high-resource, widely spoken languages such as English would be under-sampled and low-resource or less spoken languages such as Icelandic would be oversampled, thus solving the challenge of data imbalance. In this paper, the case-sensitive version is selected to retain the case information during the lexical normalization task.

By default, the maximum sequence length (max_seq_length) of the input sentence is fixed to 128. However, it resulted in the truncation of longer sentences which caused a decrease in performance as the ERR score got lowered. Hence, the max_seq_length was kept to 256 to accommodate longer input sentences. Experiments were also carried out on maximum sequence lengths of 128, 256, 300, 400, 450, and 512. It might be a greedy choice to select the highest max_seq_length but the training and inference time increases. Further, with max_seq_length set as 256, 300, 400, 450, and 512, the performance was similar, hence the final max_seq_length was set as 256. The number of epochs was kept to 100 with the early stopping parameter set to 3. It was observed that after the completion of 60-65 epochs, the loss stopped reducing and the model training was stopped automatically. To save the secondary memory, only the latest 3 epochs checkpoints and the best model were saved. This selected multilingual BERT model was finetuned by using the 'Simple Transformers' (Rajapakse, 2020)[2] library and the 'Hugging Face' (Wolf et al., 2019)[3] pre-trained models. All the experiments and model training were carried on NVIDIA Tesla V100 32 GB G-PU with primary memory (RAM) of the system being 128GB.

3.4 Post-processing based word alignment

One of the challenge faced is to align the target or predicted normalized words with the unnormalized input sentence. For this the following steps are followed:

1. Replacing the spaces in multi-gram normalized words with a special character. In this paper, underscore is used.

2. During inference phase, combining the outputs outputted by the model to extract the normalized tokens.

The importance of this step is reflected in the final evaluation results because this step makes sure that the unnormalized tokens are accurately replaced by their predicted normalized counterparts. The addition of special character (in this case underscore) helps to retain the output structure and the combination of independent word-piece token's predictions helps to retain the output.

4 Results and Discussions

This paper used Error Reduction Rate (ERR) (van der Goot et al.) as an evaluation metric for intrinsic evaluation. ERR represents

[2]https://simpletransformers.ai/

[3]https://huggingface.co/

team	avg.	da	de	en	es	hr	iden	it	nl	sl	sr	tr	trde
davda54-2	67.30	68.67	66.22	75.60	59.25	67.74	67.18	47.52	63.58	80.07	74.59	68.58	68.62
davda54-1	66.21	70.25	65.65	73.80	55.93	67.29	66.15	42.57	62.70	79.85	73.55	68.58	68.19
yvesscherrer-2	49.30	5.38	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
yvesscherrer-1	47.47	4.75	58.00	60.76	33.68	51.83	53.26	35.64	43.99	66.02	60.26	49.49	51.97
TrinkaAI (ours)	**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**
TrinkaAI (ours)	43.63	45.89	47.30	64.54	**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
team-2	40.70	48.10	46.06	63.73	21.00	40.39	59.28	13.86	43.72	60.55	46.11	15.88	29.71
learnML-2	40.30	40.51	43.69	61.57	56.55	38.11	56.19	5.94	42.77	58.25	39.99	14.36	25.68
maet-1	40.05	48.10	46.06	63.90	21.00	40.39	59.28	5.94	43.72	60.55	46.11	15.88	29.71
MFR	38.37	49.68	32.09	64.93	25.57	36.52	61.17	16.83	37.70	56.71	42.62	14.53	22.09
thunderml-2	36.48	-4.43	45.95	63.51	21.62	40.98	58.42	12.87	45.00	60.37	46.85	17.40	29.27
team-1	36.48	-4.43	45.95	63.51	21.62	40.98	58.42	12.87	45.00	60.37	46.85	17.40	29.27
cl-monoise	12.05	7.28	16.55	4.13	4.99	26.41	2.41	0.00	16.22	8.77	20.09	17.57	20.16
maet-2	7.33	2.22	4.28	21.73	0.00	9.86	19.24	0.00	2.09	18.39	8.08	0.84	1.23
learnML-1	7.33	2.22	4.28	21.73	0.00	9.86	19.24	0.00	2.09	18.39	8.08	0.84	1.23
bucuram-2	6.73	49.68	-1.91	26.81	-9.36	-10.06	-7.22	-31.68	-2.09	-1.04	42.62	9.97	14.99
bucuram-1	5.22	49.68	-1.91	26.81	-10.19	-9.86	-7.22	-31.68	-2.09	-1.13	42.62	1.01	6.57
LAI	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
machamp	-21.25	-88.92	-93.36	50.99	25.36	42.62	39.52	-312.87	1.49	56.80	39.44	-12.67	-3.42

Table 5: Error Reduction Rate (ERR) scores on test dataset of the teams participating in the lexical normalization task

treebank	avg.	de-tweede	en-aae	en-monoise	en-tweebank2	it-postwita	it-twittiro	tr-iwt151
davda54-2	64.17	73.58	62.73	58.57	59.08	68.28	72.22	54.74
davda54-1	63.98	73.58	62.21	57.90	59.02	68.28	72.15	54.76
yvesscherrer-2	63.72	73.49	60.64	56.24	60.30	68.11	72.29	54.95
yvesscherrer-1	63.72	73.49	60.48	56.26	60.27	68.15	72.43	54.94
monoise	63.44	73.20	62.27	56.83	58.90	67.55	70.69	54.61
TrinkaAI (ours)	**63.31**	**72.86**	**60.32**	**56.74**	**60.31**	**67.34**	**70.72**	**54.89**
TrinkaAI (ours)	63.12	72.86	60.16	56.64	59.87	66.98	71.14	54.20
maet-1	63.09	72.80	59.44	56.64	59.80	67.41	71.07	54.45
team-2	63.03	72.80	59.44	56.64	59.80	67.19	70.86	54.45
Thunderml-2	63.02	72.67	59.57	56.74	59.25	67.34	71.35	54.24
team-1	63.02	72.67	59.57	56.72	59.24	67.34	71.35	54.23
thunderml-1	62.95	72.52	59.31	56.74	59.86	67.09	71.00	54.09
learnML-2	62.88	72.31	58.98	56.16	59.98	66.99	71.24	54.48
cl-monoise	62.71	72.65	60.90	55.26	58.53	66.53	70.10	54.98
bucuram-2	62.53	72.57	59.57	54.20	59.81	66.74	69.99	54.84
bucuram-1	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
maet-2	62.22	72.69	58.50	52.89	60.01	66.47	69.99	55.00
learnML-1	62.22	72.69	58.50	52.89	60.01	66.47	69.99	55.00
machamp	61.89	71.28	60.77	54.61	57.97	64.65	69.82	54.08

Table 6: Labeled Attachment Scores (LAS) on test dataset of the teams participating in the lexical normalization task

accuracy normalized for the number of words that need to be normalized. The main advantage of using ERR as a metric is that it accounts for the difficulty of the task (similar to that of Cohen's Kappa (Cohen, 1968)

$$ERR = \frac{Accuracy_{system} - Accuracy_{baseline}}{1.0 - Accuracy_{baseline}} \quad (1)$$

$$ERR = \frac{TP - FP}{TP + FN} \quad (2)$$

Equation 1 represents the formula to compute

the ERR score. ERR can be calculated by first computing the accuracy of the normalization system ($Accuracy_{system}$) and accuracy of the baseline system which will always return the original word ($Accuracy_{baseline}$). ERR can also be calculated by 2 where,

- TP: Annotators normalized and the proposed system normalized correctly.
- TN: Annotators did not normalize, and the proposed system did not normalize.
- FP: Annotators did not normalize, but the proposed system normalized.
- FN: Annotators normalized, but the proposed system did not find the correct normalization (because the proposed system either kept the original word or proposed a wrong candidate).

For the extrinsic evaluation, dependency parsing is considered. Here, the impact of lexical normalization on the quality of the parsed trees is computed using the LAS, i.e., the percentage of words which have both the correct syntactic head and the correct dependency label.

Table 5 depicts Error Reduction Rate (ERR) scores computed on the test dataset for intrinsic evaluation. The ERR scores of individual languages, as well as average scores, are computed for each team that participated in the lexical normalization shared task. The proposed system named TrinkaAI scored an overall average ERR of 43.75. On the Spanish language, TrinkaAI's ERR score was 61.33, the highest among all the shared task participants. Our result shows that by adding additional training data, the model's performance can be considerably increased. Overall, the proposed system, TrinkaAI, ranked 4th in the lexical normalization shared task per the average ERR scores obtained from all the languages scores.

Table 6 depicts the scores for extrinsic evaluation. TrinkaAI had the highest LAS score of 60.31 on test data en-tweetbank2. This shows the potential of the proposed approach in normalizing un-normalized sentences. The overall average LAS score obtained by the TrinkaAI was 63.12, which stood 5th among other participants.

5 Conclusion and Future Scope

In this paper, we propose a multilingual sequence labeling based approach to solve the problem of lexical normalization. We show that a single model is capable of performing well on all languages. We finetuned multilingual BERT on a downstream task of lexical normalization along with finetuning of some of its hyperparameters. Our proposed system, TrinkaAI, scored an average ERR score of 43.75 in the intrinsic evaluation and an average LAS score of 63.31 in the extrinsic evaluation. Further, TrinkaAI achieved the highest ERR score of 61.33 for the Spanish language. This denotes that the performance of the system can be improved considerably by incorporating additional data during training. TrinkaAI also achieved the highest LAS score of 60.31 for the en-tweetbank2 test data.

The performance of our proposed system can be further improved by training the model with 'large' configurations having a high number of parameters. The accuracy can be improved by additional data and by using language-sensitive embeddings.

References

Jacob Cohen. 1968. Weighted kappa: nominal scale agreement provision for scaled disagreement or partial credit. *Psychological bulletin*, 70(4):213.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Bo Han. 2014. *Improving the utility of social media with natural language processing*. Ph.D. thesis.

Bo Han and Timothy Baldwin. 2011. 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*, pages 368–378.

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.

Xiaohua Liu, Shaodian Zhang, Furu Wei, and Ming Zhou. 2011. Recognizing named entities in tweets. In *Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies*, pages 359–367.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis,

Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Nikola Ljubešic, Katja Zupan, Darja Fišer, and Tomaz Erjavec. 2016. Normalising slovene data: historical texts vs. user-generated content. In *Proceedings of the 13th Conference on Natural Language Processing (KONVENS 2016)*, volume 16, pages 146–155.

Ismini Lourentzou, Kabir Manghnani, and ChengXiang Zhai. 2019. Adapting sequence to sequence models for text normalization in social media. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 13, pages 335–345.

Benjamin Muller, Benoît Sagot, and Djamé Seddah. 2019. Enhancing bert for lexical normalization. In *The 5th Workshop on Noisy User-generated Text (W-NUT)*.

Deana Pennell and Yang Liu. 2011. A character-level machine translation approach for normalization of sms abbreviations. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 974–982.

T Rajapakse. 2020. Simple transformers.

Sarah Schulz, Guy De Pauw, Orphée De Clercq, Bart Desmet, Veronique Hoste, Walter Daelemans, and Lieve Macken. 2016. Multimodular text normalization of dutch user-generated content. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 7(4):1–22.

Kushal Vaghani. 2020. Curating social media data. *arXiv preprint arXiv:2002.09202*.

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.

Rob van der Goot, Alan Ramponi, Arkaitz Zubiaga, Barbara Plank, Benjamin Muller, Iñaki San Vicente Roncal, Nikola Ljubešić, and ˜Çetinoglu. MultiLexNorm: A shared task on multilingual lexical normalization.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, pages arXiv–1910.

Wei Xu, Bo Han, and Alan Ritter. 2015. Proceedings of the workshop on noisy user-generated text. In *Proceedings of the Workshop on Noisy User-generated Text*.