Lexical Normalization as a Machine Translation problem

-documentation-

Constantin Gabriel-Adrian
gabriel.constantin13@s.unibuc.ro

Sociu Daniel
daniel.sociu@s.unibuc.ro

Abstract

Transformers are one of the go-to models used in solving tasks sequence-to-sequence. In this project, we explore how a text-to-text T5 model can be used for machine translation for multiple languages. Similarly with nanoT5, we explore different T5-based pretrained architectures. We fine-tune them on MultiLexNorm, a lexical normalization dataset containing various languages, and compare and discuss the results.

1 Introduction

Lexical normalization involves correcting the data to the usual canonical form. This usually means correcting abbreviation or mistaken words to the correct grammatical or dictionary form.

We chose this topic for our research because such an approach can prove very useful as a preprocessing method for various datasets that contain comments or posts from social media sources, given that a major problem in NLP is the lack of quality annotated data. In this instance, the challenge goes beyond just correction when considering a multi-language approach.

Aside from reading related research and understanding the models which were specific to both of us, here is a summary of contributions to the project of each member:

Constantin Gabriel-Adrian

1. elaborated the scripts for plotting the result, processing the dataset and constructing the files needed in order to leverage the nanoT5 implementation
2. proposed and experimented with T5-small model baselines on English language
3. proposed and experimented with reduce learning rate on plateau and early stopping
4. elaborated the introduction, theory and description of the approach and possible limitations of the research paper

Sociu Daniel

1. forked the implementation from T5 and adapted it for our task
2. proposed and experimented various experimental setups with multilingualT5-small and different batch sizes
3. elaborated on the software tools used for implementation and performance comparisons sections of the research paper

In order to develop this kind of correction model, the initial phase entails the representation of tokens (words) from the dataset as (numerical) embeddings. This can be accomplished fairly easy with the help of a transformer. In this study, we have explored the T5 model (Text-to-Text Transfer Transformer) (Raffel et al.) which is pretrained on diverse and way bigger datasets from multiple languages: "C4" dataset ("Colossal Clean Crawled Corpus").

A similar goal could also be achieved with the help of Mixtral (Jiang et al., 2024). Mixtral follows a similar shape to a classic transformer. What sets it apart is the fact that, after the attention heads, instead of using a feed forward network and thus the same set of weights for each attention head output, it proposes using multiple "experts". These experts follow the same idea that inspired the splitting of the attention heads in a sparse manner. As such, each output can be passed to a different expert i.e. FFN. In order to decide which expert should be applied for current input, they use a routing neural network. This not only provides the benefit of running different experts, but also helps with paralelization on multiple GPUs.

For the dataset, we are using Lexical Normalization Dataset (van der Goot et al., 2021). This data represents texts extracted from posts originating mainly from Twitter. These messages have been collected from 12 languages.

2 Approach

As previously mentioned, we will be using various T5-based models for the aforementioned dataset.

In order to use the texts in our model, we created multiple language-based json files for train and test set containing a pair of positive-negative example for context for the model and several pairs of given input - corrected output.

Considering that the input statements have different lengths (number of tokens), to pass them through a transformer we would have to make all of them a specific length. This was done by splitting the longer statements in multiple parts of max length 1024, and the rest that didn't reach the set length, we padded them. For the output, we set the maximum sequence length to 128.

As previously mentioned, we used T5 as the overall architecture of our model.

A transformer is a deep learning model whose goal is solving "sequence to sequence" tasks i.e. encode-decoder. Transformers are similar with recurrent neural networks (RNN), the main difference being that they don't process the data in a specific order. Thus, a given sequence is analyzed in parallel and the influence or importance of each part of the sequence is determined at every step. This sequence is passed to the encoder part which uses the self-attention mechanism. Given the fact that by paralelizing this task the initial order is forgotten, the transformer adds for each embedding a positional encoding.

The second part, the decoder follows a similar patter, but in this case using casual self-attention, which enables the model to only use past outputs. Each attention block is split into multiple heads which are independent, the output being the concatenation of the results from each head. As such, the model receives overall a text for context, this being the input data (the message, such as the positive-negative pairs we previously mentioned), and based on this it generates the required output in the form of text. This enables the T5 model to be used on a variety of tasks, such as text classification, question answering, text sumarisation and even machine translation, tasks on which the model proposed has already been pretrained.

Regarding the actual training process, we relied on nanoT5 (Nawrot, 2023), which facilitates fine-tuning of T5-style models. This represents training the model starting from the weights of the pre-trained model and also updating the output head to be corresponding for our task. In our case, the weights are sourced from the T5 model pretrained on the C4 dataset.

We have explored several models:

- T5-small
- mT5-small (Multilingual T5-small)

We started off with the T5-small model, which was trained and tested only on the English dataset. This was done such that we would have not only a sanity-check result in order to ensure that the code was working properly and the model was learning, but also to have a baseline and see how the performance changes when switching to the multilingual task.

We used AdamW as an optimizer and reduced the learning rate when encountering a plateau region. This represents a sequence of 5 epochs when the loss on the validation dataset hasn't improved. When this situation occurs, the initial learning rate is gradually reduced by a factor of 0.5.

The previously mentioned scheduler replaced the scheduler that the nanoT5 was using on the finetuning task (LambdaLR scheduler), which improved the results slightly. Though, to achieve this imporovement, reducing the batch size was also requried.

We are also stopping the training if the validation accuracy hasn't improved, and we save the model with the best performance.

We started from the nanoT5 github repository (Nawrot and Pei) which is specifically optimized to fine-tune big models efficiently and also maintain a very close accuracy to the original model, which is all implemented using PyTorch. For loading the transformers model architecture, tokenizer and other various utils for batch encoding, we used the 'transformers' framework. Therefore we used this highly optimized repository to train each model for 25 epochs, the training on the EN subset takes 10 minutes, whereas the one on the whole dataset (multilingual) takes approximately 40 minutes. They all use the initial learning rate of 5e-5.

In order to evaluate the performance of the models, we used accuracy and Rouge-L, which measures the number of matching n-grams between the reference text and the generated output text.

The results we obtained can be seen in Table 1 and the plots for the best model can be viewed in Fig. 1.

As we can see, the best performing model on the multilingual task was the T5-small, with an accuracy on validation dataset of 55.1 and a Rouge-L of 71.3

All the code, documentation and presentation is available on our github repository here (Gabriel-Adrian and Daniel).

3 Limitations

One major limitation in our experiments was the lack of computational resources, notably GPU processing power. This not only reflects in the training and inference time, but also in the number of parameters that we used inside the model. This is why we used the small variants of the T5 and Multilingual T5 models. We have also tried to experiment with Mixtral, however, as previously mentioned, it uses multiple experts as FNN. According to the original paper (Jiang et al., 2024), the number of experts needed is 8, but the model uses the top-2. This means that all these experts need to be loaded in memory, which results in very high VRAM usage.

Another limitation is given to the relatively small size of the dataset. It contain 1200 sentences on the training set and another 1200 on the test set. Also, as stated by the author of the Lexical Normalization Dataset (van der Goot et al., 2021), there are differences between the annotations of the 12 languages since they used both manual and automatic methods. These can potentially have an impact on the ability of the model to generalize on multiple languages as our task requires.

4 Conclusions and Future Work

We demonstrated that a multilingual approach for lexical normalization is possible using transformers, and that finding the right set of parameters can yield interesting results even an small models. We would have loved to have more computational resources such that we could train at least on medium models, or on models that require a lot of parameters such as Mixtral.

One way we could have ameliorated this issue and improve the experimentation would have been maybe to freeze some layers and only train with those left, however this might have had a significant impact on the performance, since the models have not been pretrained on tasks with similar goals.

All in all, we are glad that we were able to learn about new models, experiment with different setups and write about it in conference-styled report.

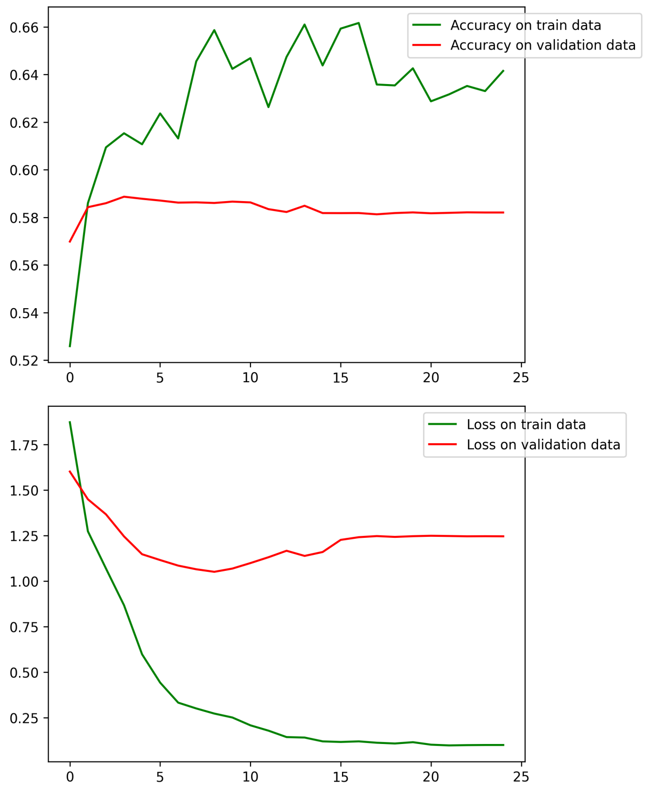

Figure 1: The accuracy and loss plots of our best multilingual model.

Model	Dataset Lang.	Batch size	Scheduler	Train Acc.	Val Acc.	Rouge-L
T5-small	EN	16	LambdaLR	58.5	53.5	50.2
T5-small	Multilingual	16	LambdaLR	42.5	41.8	60.5
mT5-small	Multilingual	8	LambdaLR	48.0	45.9	70.8
T5-small	EN	4	ReduceLROnPlateau	*65.9*	*58.8*	*54.6*
T5-small	Multilingual	4	ReduceLROnPlateau	**59.7**	**55.1**	**71.3**
mT5-small	Multilingual	4	ReduceLROnPlateau	58.2	53.8	67.8

Table 1: Results of training on MultiLexNorm dataset using nanoT5. In **bold**, the best performance on multilingual dataset and in *italic* the best performance on English

References

Constantin Gabriel-Adrian and Sociu Daniel. The repository we used for our implementation.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2024. Mixtral of experts.

Piotr Nawrot. 2023. nanot5: A pytorch framework for pre-training and fine-tuning t5-style models with limited resources. *ArXiv*, abs/2309.02373.

Piotr Nawrot and Qizhi Pei. The original nanot5 repository.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. [1910.10683] exploring the limits of transfer learning with a unified text-to-text transformer. `https://arxiv.org/abs/1910.10683`. (Accessed on 01/17/2024).

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. 2021. 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.