Neural Machine Translation: English-Afrikaans

Jean Lucien Randrianantenaina
Applied Mathematics, Machine Learning and Artificial Intelligence
Stellenbosch University

***Abstract*—**

I. Introduction

II. Neural Machine Translation

Neural Machine Translation (NMT) is at the heart of many language translation systems in the current era. We will briefly present the main idea behind this machine learning concept.

Most of these models are based on the encoder-decoder architecture. This architecture is usually introduced as a neural network that learns the identity function, e.g., reproduces its input. However, it can be used in tasks like image processing, computer vision, and Natural Language Processing (NLP).

In the case of NMT, the architecture learns how to translate a language A to a language B. Figure **??** illustrates a high-level representation of such an architecture.

Usually, we use Recurrent Neural Networks (RNN) to create these models, also known as seq2seq (sequence-to-sequence) models. A particularity of these neurons is to take their output as the next input. The three most well-known RNNs are the Vanilla RNN, Gated Recurrent Unit (GRU), and Long Short-Term Memory (LSTM). However, these architectures seem hard to train and computationally slow due to their recurrence.

One way researchers have found to improve the performance of these architectures is the attention mechanism, which aims to conserve the importance of each output of the encoder when decoding its input. This technique was primarily used with RNNs but was later used with classical neural networks (without recurrence), giving rise to what we know as self-attention. The main idea behind this is to rely only on the output of the layers using some weighting system to produce the next layers or outputs.

Lastly, attention and self-attention yielded the so-called transformer block found in the state-of-the-art of many machine learning models today.

III. Methodology

In this task, we will focus on translating English to Afrikaans using a small dataset from the engineering assessment corpus from Stellenbosch University.

To achieve that, we will implement from scratch a vanilla RNN, LSTM, GRU, and their variants with attention. Then, we will use the pre-trained `opus-mt-en-af` model. The goal is to show if we can implement such models and perform comparisons.

A. Dataset

The provided dataset is small and may not have much vocabulary. To increase word coverage, we included the English-Afrikaans dataset from the Tatoeba project. However, our main dataset focus is on the engineering domain, and adding this dataset will increase common words only, which is one of the reasons for using a pre-trained model in the second part.

To clean the dataset, we tried as much as possible to detect the LaTeX environment and compactify it as single words (removing spaces), adding spaces on delimiter characters such as "(),[],{}" to treat them as a single character, finally removing extra spaces. We do not change the case and do not remove special characters. This may limit our model, but that will be close to a real-world task (Another option may be possible for the case, but it will be for future work).

B. From Scratch Implementation

As mentioned earlier, our from-scratch model will have the parameter listed in the table I. The only thing that will differentiate them will be the architecture that we use.

Parameter	Value
Embedding Size	256
Hidden Size	1024
Number of Layers	2

TABLE I: Hyperparameters used in the from-scratch models.

Each model will be trained at most with 50 epochs, using NAdam as optimiser with a learning rate of $10-3$ and the CrossEntropyLoss function.

C. Using a Pre-trained Model

Leveraging a pre-trained model can be a highly efficient approach, and allows us to save substantial time and resources. These models are trained on massive amounts of data, meaning they already possess good parameters and are ready to use. However, they may not fit our specific domain or requirements, so we need to perform additional operations like finetuning.

Since we aim to translate engineering assessments from English to Afrikaans using a relatively small dataset, we fine-tune the opus-mt-en-af model. This model is part of a larger collection of state-of-the-art Neural Machine Translation (NMT) models [8]. These models are based on the Marian-MT framework [7] and are among the leading open NMT systems available.

The opus-mt-en-af model was initially trained on a public dataset from the Open Parallel Corpora [3]. While this dataset provides a strong foundation, it may not contain all the domain-specific technical vocabulary necessary for translating engineering assessments. By fine-tuning the model on our dataset, we expect to improve its performance in handling these specialized terms and phrases.

To implement the task, we use the `transformer` library that allows us to obtain the weights of the model from huggingface [2] using PyTorch. It also allows us to tokenize our data set, using the tokenized provided by opus-mt which is based on `sentencepiece` [5] of Google (with the de-tokenizer as well).

Our dataset was preproprocessed using the same approach for the from-scratch implementation. We use only this corpus because the Tatoeba dataset is already part of the training data of the pre-trained model.

Then for the parameter optimization of the model, AdamW [4] as suggested in the huggingface NLP course for the translation model finetuning. Concerning the loss function, we did not find relevant information on it as it was provided directly by the pre-trained model.

D. Model evaluation

To evaluate all the models by computing the bilingual evaluation understudy (BLEU) metric, which allows us to measure how similar the results of the machine translation are to human translation (references).

For the from-scratch implementation, we evaluate the augmented dataset and the engineering assessment only.

For the finetuning, we evaluate it before the training and after training on the validation set, this is done to see if we effectively able to improve the model on the new domain.

IV. Results and discussion

V. Conclusion

References

[1] Herman Kamper. *NLP817*. https://www.kamperh.com/nlp817/, 2022–2024.

[2] Hugging Face https://huggingface.co/

[3] Open Parallel Corpora https://opus.nlpl.eu/

[4] AdamW https://pytorch.org/docs/stable/generated/torch.optim.AdamW.html

[5] Sentencepiece https://github.com/google/sentencepiece

[6] Helsinki NLP https://blogs.helsinki.fi/language-technology/, https://huggingface.co/Helsinki-NLP

[7] Junczys-Dowmunt, M., Grundkiewicz, R., Dwojak, T., Hoang, H., Heafield, K., Neckermann, T., Seide, F., Germann, U., Aji, A.F., Bogoychev, N. and Martins, A.F., 2018. Marian: Fast neural machine translation in C++. *arXiv preprint arXiv:1804.00344.*

[8] Jörg Tiedemann and Santhosh Thottingal. 2020. OPUS-MT - Building open translation services for the World. *In Proceedings of the 22nd Annual Conference of the European Association for Machine Translation, pages 479–480, Lisboa, Portugal. European Association for Machine Translation.*