ZPJa: Neural Machine Translation from Scratch - English to Latvian

Sandis Studers
xsandi01@fit.vut.cz

Santosh Kesiraju
kesiraju@fit.vut.cz

Abstract

This technical report details the creation of a sequence to sequence translation model that translates sequences from English to Latvian. The model is using a transformers architecture and is trained with 320 000 sentence pairs from the OPUS Europarl English-Latvian dataset.

The resulting model achieves a BLEU score of 0.103 and a COMET score of 0.702 on 5000 sample sentence pairs from the test set. The report describes the difficulty involved in creating translations to Latvian language which has a rather flexible sentence structure and the importance of an appropriate dataset which can better display the possible structures and help the model make sense from them. The limits of the current dataset in evaluating the model are also noted, as the model provides only one reference to evaluate against while many possibilities may exist.

1 Task Definition

The main task described in this technical report is the creation of a neural network based translator which translates textual sequences from English to Latvian language. To achieve this goal the subtasks are:

- The preparation of a ready dataset for usage in a neural network. The dataset includes sentence pairs in English and Latvian and must be transformed to be able to serve as an input to a neural network.
- Implementation of a neural network based on a transformer architecture that can accept a list of input source and target tokens and output the probabilities for the next token.
- Implementation of a neural network training process which is capable of training the implemented neural network with the prepared dataset.
- Implementation of an inference procedure which produces a Latvian translation when fed a source sequence in English.
- Implementation of a beam search and greedy decode algorithms.
- Evaluation of the trained neural network with BLEU and COMET metrics.

2 Method

For this task a dataset of source sentences in English and target sentences in Latvian is used. Firstly, separate vocabularies of tokens are created for English and Latvian sentences with each token being a sub-word level text string. Then the dataset is tokenized each piece according to its vocabulary and transformed into a list of token ids. Each sequence is prepended by a start-of-sequence token and appended by an end-of-sequence token.

Given that the input sequences and target sequences are of variable length padding masks are utilized, so that multiple sequences can be batched together and processed in parallel. Padding is used on a batch level.

In the end each sequence is of structure:

$$\text{Start-of-sequence token} + \text{sequence tokens} + \text{end-of-sequence token} + n \times \text{padding-token} \quad (1)$$

Where n is the difference between the sequence length and its batch's longest sequence's length.

Masks are also used to hide the subsequent tokens of the target sequence, so that during the prediction of the next token the model can see only the source sequence tokens and the preceding target sequence tokens.

In this task the translation was performed by a neural network model based on transformer architecture. The input to the encoder is a matrix of size $B \times Ns$ where B is the batch size and Ns is the length of each source sequence, the input to the decoder is a matrix of size $B \times Nt$ where Nt is the length of each target sequence. Ns and Nt varies for each batch.

Inference uses beam search decode algorithm which also covers greedy decode, since beam search with a single beam is equivalent to greedy decode. Evaluation of the neural network for both BLEU and COMET were performed on the entire test set (16 000 sentence pairs).

3 Experimental Setup

The dataset used for this task was a set of English-Latvian sentence pairs taken from OPUS' provided Europarl dataset collection. The dataset contains 621 325 sentence pairs in total and its content consists of speeches and discussions that have taken place during Europarlament sessions.

Here is a sample sentence pair from the dataset:

Source sentence in English: Future action in the field of patents (motions for resolutions tabled): see Minutes

Target sentence in Latvian: Turpmākās darbības patentu jomā (iesniegtie rezolūcijas priekšlikumi) (sk. protokolu)

Firstly, it was observed that the dataset contains quite many sentence pairs where the target sentence was not in Latvian but instead in another language such as Greek, Italian, German or others. These sentences concerned not the discussions or speeches made during the session but the details of the session itself and were always contained in parentheses. Here is an example of an erroneous sentence pair:

Source sentence in English: (The sitting was closed at 11.55 p.m.)

Target sentence which is actually in Italian not Latvian: (La seduta è tolta alle 23.55)

To deal with this issue all sentence pairs that begin with parentheses were simply removed from the dataset.

After that a certain number of the shortest sentences from the dataset were taken for the model's dataset. Initially 150 000 sentence pairs were taken. But as the training with this much data was quick while the results were not very good, the model's dataset was extended to 320 000 sentence pairs which yielded a model with better performance.

The vocabulary and tokens were created with SentencePiece library. A vocabulary size of 32 000 was chosen, since it provided a good performance while not being too computationally expensive and seemed like a good fit for the given amount of data. A separate vocabulary for English and Latvian was created and then the source and target sentences were transformed into the token ids based on their respective vocabulary.

At the core of the neural network model was PyTorch's provided nn.Transformers class which is based on nn.Module superclass and implements the transformers architecture as seen in the "Attention is all you need" paper. However PyTorch's implementation lacks the embedding and positional encoding layers that transform the initial source and target tensors, and it also lacks the final linear layer that transforms the final layers results into logits. (pyt, 2023)

The dataset was split in 90% of data (288 000 sentence pairs) for training set and 5% (16 000 sentence pairs) each for validation and test splits.

To implement these layers and connect them with the nn.Transformers layer a separate Transformer class was created with nn.Module class as its superclass. The source and target embedding layers were instances of PyTorch's nn.embedding class and the final linear layer was an instance of PyTorch's nn.Linear class. (pyt, 2023)

For positional encoding a ready implementation was taken from Annotated Transformer's internet post by Austin Huang and others. In this implementation each sequence position gets a unique positional encoding vector based alternating sine and cosine functions. (Huang et al., 2022)

All these layers were initiated in the created transformers class and sequentially executed during the forward pass to obtain the logits. A variety of hyperparameter configurations were tried to get a feel for where the model is not powerful enough or where the model is so large that the data starts overfitting it. The configuration of hyperparameters used in the final model is:

- Learning rate: 0.0001
- Embedding size: 256
- Number of attention heads: 8
- Hidden layer size: 256
- Encoder layers: 3
- Decoder layers 3
- Dropout: 0.1

The model was trained on the training set for 10 epochs. Each epoch took about 5 minutes, so in total the training was about 50 minutes long. The training was done with Google Colab's provided Nvidia's T4 GPU which has 15 gigabytes of video RAM. To take advantage of the available VRAM the batch size was set to 64.

BLEU evaluation was done using PyTorch's offered BLEU metrics library. (pyt, 2023)

COMET evaluation was performed with Unbabel's COMET library using a general translation model "COMET-22" presented in 2022 at The Seventh Conference on Machine Translation (WMT22). (Rei et al., 2022)

4 Results and Analysis

The described model was trained on the Nvidia T4 GPU for 10 epochs. With each epoch taking about 300 seconds to train, the total training length was about 50 minutes.

The training results can be seen in the table below.

Epoch	Training loss	Validation loss	Validation loss decrease
1	6.086	5.188	-
2	4.897	4.446	0.742
3	4.320	4.004	0.442
4	3.941	3.714	0.29
5	3.672	3.518	0.196
6	3.467	3.375	0.143
7	3.304	3.265	0.11
8	3.177	3.187	0.078
9	3.069	3.123	0.064
10	2.974	3.067	0.056

Table 1: Training and validation losses over epochs

As can be seen from the table the validation loss was still steadily and substantially decreasing during the last epochs, therefore the model could benefit and provide better performance with even more training.

The initial model training was done with a dataset consisting of 150 000 sentence pairs, 135 000 of them being used for training. This model provided considerably worse results with validation loss plateauing at around 3,35 under various model configurations. Increasing the dataset to 320 000 sentence pairs with 288 000 of them being used for training yielded a much better performance with validation loss reaching 3,067 after 10 epochs and possibly decreasing more in further training.

I believe increasing the dataset more would yield even higher performance. Latvian language is particular in that its sentences don't have definite structures and words within a sentence can be combined in many different ways with the sentence having the same meaning semantically. For example:

Source sentence: Implementing measures (level 2) of the Transparency Directive (vote)

Target sentence: Direktīvas "Pārredzamība" īstenošanas līdzekļi (2. līmenis) (balsojums)

Literal translation of target sentence: Directive's "Transparency" implementation measures (2. level) (vote)

The words here are in a very different order than in the source sentence while they could actually have been written in the same order as source sentence and still made sense. Such decisions were probably made subjectivelly by the translator and can be inconsistent across the dataset.

Furthermore, some of the sentences didn't have exact but more so approximate translations. For example:

Source sentence: Programme of support for the European audiovisual sector (MEDIA 2007) (vote)

Target sentence: Atbalsta programma audiovizuālajā nozarē (MEDIA 2007) (balsojums)

Literal translation of target sentence: Support programme in audiovisual sector (MEDIA 2007)

It can be seen that the word "European" has been ommitted from the target translations. Such differences can cause confusion in the model that might be fixed with a larger dataset.

The final loss on the test set was equal to 2.609. Evaluation of the final model was performed with BLEU and COMET metrics on the test set using beam search with both size 1 (which equals to greedy search) and size 5. The results can be seen in the table below.

Evaluation method	Greedy decode	Beam search decode
BLEU	0.103	0.087
COMET	0.702	0.687

Table 2: Evaluation of decoding methods

As can be seen from the table the greedy decode actually slightly outperforms beam search decode on both evaluation metrics.

Both algorithms achieved their lowest individual comet scores (0.149 and 0.163 respectively) on the same sample:

Source sentence: Buckfast wine should, apparently, be banned because it contains both alcohol and caffeine.

Target sentence: Bakfastas vīnu ir paredzēts aizliegt tāpēc, ka tas satur gan alkoholu, gan kofeīnu.

Greedy prediction: Tulffffffffffffffffffffff

Beam search prediction: ffffffffffffffffffffforforf

It's possible that the model became confused by the word "Buckfast" and wasn't able to deal with it meaningfully.

Inference speeds were also compared between both decoding algorithms during the 5000 test set predictions. Greedy decoding algorithm managed to infer the 5000 examples in 360 seconds, yielding an average inference time of 0.072 seconds per sample while beam search decoding algorithm took 2184,3 seconds, yielding a much slower average inference time of 0,437 seconds per sample.

Comparing the both times greedy decoding algorithm is 0.437 / 0.072 = 6.07 times faster than beam search decoding algorithm.

5 Conclusion

In conclusion, I believe that the created model can serve as a good starting point or a prototype for a more advanced and better performing translation model. Further increasing the dataset and the model size could lead to better results. Optimizing the beam search algorithm or using a different decoding algorithm could also lead to better evaluation results than the plain greedy decoding algorithm.

I believe the main difficulty in training this translation model is the different sentence structures seen in English and Latvian languages and the very flexible sentence structuring of the Latvian language. On one hand it makes it more difficult for the model to learn what are the allowed and legitimate sentence structures and ways to translate an English sentence into a Latvian sentence and to display all of these requires a larger dataset. On the other hand, the possibility of several legitimate translations severely reduces the usability of the existing dataset for the model's evaluation as it provides only one reference to measure the candidate solution against, even though many more may exist.

References

2023. Pytorch documentation. Accessed: 04.01.2023.

Austin Huang et al. 2022. The annotated transformer. Accessed: 04.01.2023.

Ricardo Rei et al. 2022. COMET-22: Unbabel-IST 2022 Submission for the Metrics Shared Task. In *Proceedings of the Seventh Conference on Machine Translation (WMT22)*. Accessed: 04.01.2023.