Neural Machine Translation from Scratch

Josef Kotoun (xkotou06)

December 13, 2023

1 Topic propsal and goals of the project

The project focuses on the implementation of a transformer-based translator. It uses existing implementation of transformer encoder, decoder and positional encoding from Keras library, but is trained from scratch. Greedy search and Beam search decoding algorithms are implemented for evaluation. For training the model from scratch, the europarl dataset was chosen, specifically a pair of languages Czech and English to train the English to Czech translator. This dataset is splitted into training, testing and validation parts. After training, the model is evaluated on the test split of the dataset using chrF2 and bleu metrics for variants using both implemented decoding algorithms. Finally, the inference speed using both implemented decoding algoritms is also compared.

2 Problem definition

Machine translation is a challenging task in natural language processing that enables automatic conversion of text from one language to another. The key challenge lies in producing sentences in the target language that are grammatically correct and capture the semantic meaning of the source sentence. The complexity arises from linguistic differences between languages. Commonly used architecture for this problem is transformers architecture due to its possibility to selectively focus on different parts of the source sentence when generating each word in the target sentence. This mechanism is called attention.

2.1 Transformer Architecture

The transformer architecture is used in many different of sequence-to-sequence tasks including machine translation. Its innovation lies in relying on multi-head attention mechanism instead of traditional recurrent or convolutional structures. The attention mechanism enables the model to weigh the significance of different input tokens dynamically.

The core building blocks of transformer are encoder and decoder. The encoder processes the input sequence, capturing its contextual information using the self-attention mechanism. The decoder generates output sequence utilizing self-attention to capture contextual information but with an additional cross-attention mechanism, allowing it to

consider relevant parts of the input sequence. The decoder generates output sequence by predicting probability distribution over vocabulary for one token at a time. During training process, the most common practice is to use a technique called teacher forcing. Teacher forcing involves providing the true target sequence as input to the decoder, rather than using the previously generated tokens, which helps stabilizing the training process. During evaluation, the model uses an autoregressive approach, generating tokens one at a time and using its own predictions as inputs for subsequent steps.

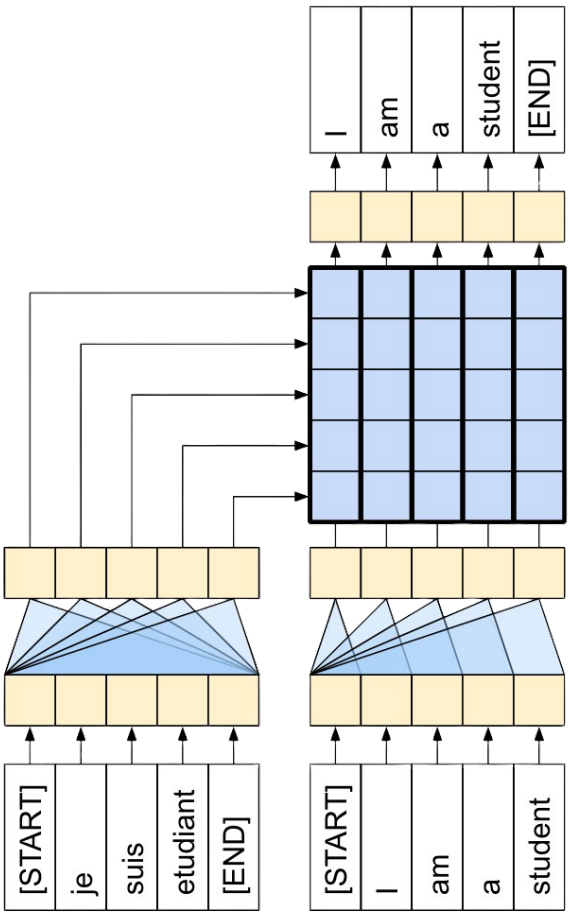

Figure 1: Transformer based translator training process [Ten23]

2.1.1 Decoding algorithms

Decoding in machine translation refers to the process of generating sequence of words in the target language based on the learned representation from the source language. Transformer decoder generates probability distribution over vocabulary for one token at a time. We need to find output sequence of tokens based on tokens distribution such that its joint probability is highest. However, exhaustive search is not efficient for longer sequences, because the complexity of exhaustive search grows exponentially with respect to the length of the output sequence. Therefore, we need a suitable algorithm to find the best possible solution, even though the solution will not be optimal. The two primary decoding algorithms explored in this project are greedy decoding and beam search decoding.

2.1.2 Greedy Decoding

The greedy decoding algorithm is very simple. It selects the token with maximum probability given the current context of already generated tokens. This locally optimal strategy simplifies the decoding process but may overlook global dependencies, potentially resulting in suboptimal translations. However the algorithm is very easy to implement and it is faster than more complex algorithms.

2.1.3 Beam Search Decoding

The beam search maintains a set of top-k possible sequencies at each step, where k determines the beam size. The model generates multiple hypotheses simultaneously and selects the most probable ones. This approach often leads to better translations compared to the simpler greedy decoding, however its computationally more demanding and it is not guaranteed that it finds optimal solution either.

3 Implementation

3.1 Programming Tools

Dataset preprocessing, training and evaluation were implemented using Python. Keras with TensorFlow backend was used to define structure of transformer model, with usage of bultin encoder, decoder and positional embedding from Keras library. Training tokenizers was done with Keras_nlp's wordpiece tokenizer. To check how well the translations worked, Keras_nlp and sacrebleu libraries were used.

3.2 Datasets

The project uses the Europarl dataset[Koe05], consisting of approximately 650,000 English-Czech sentence pairs. This dataset is publicly available at `https://www.statmt.org/europarl/` in the form of an archive, which consists of two files, each dedicated to one of the languages involved in the chosen language pair. Individual lines of file contain sentences in the given language in the same order as in other files for other languages. This dataset was divided into training, validation, and test splits, with a 70:15:15 ratio.

3.3 Training Hyperparameters

The transformer model was trained with a batch size of 16, learning rate of 1e-4 for 20 epochs.The Adam optimizer was used with early stopping based on validation loss, with patience of 3. Sequences were limited to 64 tokens, and the transformer architecture featured 8 attention heads, an intermediate dimension of 2048, and an embedding dimension of 256.

3.4 Used Hardware

Single NVIDIA A40 48GB GPU was used for both training and evaluation. It was provided by Cesnet Metacentrum

3.5 Implementation details

Implementation of this project consists of these python scripts: `create_dataset_splits.py`, `preprocess_dataset.py`, `train_tokenizers.py`, `train.py` and `evaluate.py`. The `create_dataset_splits.py` loads both files with english and czech sentences, shuffles it randomly and splits it to train, validation and test splits in 70:15:15 ratio. The `train_tokenizers.py` creates `tf_dataset` from training splits and trains WordPiece tokenizers from `keras_nlp` for Czech and English. The `preprocess_dataset.py` takes care of tokenizing the training and validation splits and other preprocessing of dataset and saving it in tensorflow dataset format for training. The `train.py` defines the model architecture and trains the model using training hyperparameters described in section 3.3. The `eval.py` contains implementation of Greedy and Beam search decoding algorithms. It runs evaluation on specified number of evaluation samples and outputs chrF2 and BLEU metrics. Running the scripts requires downloading and extracting the europarl cs-en dataset archive to `./datasets/europarl/`.

The `preprocess_dataset.py`, definition of model in `train.py` and `decode_sequences` function from `eval.py` were inspired by code examples from Keras documentation[1].

4 Evaluation and Analysis

4.1 Evaluation metrics

4.1.1 chrF

chrF (Character-level F-score) is metric for machine translation evaluation, which calculates similarity between a translation output and reference translation using n-grams of characters instead of n-grams of words. Metrics based on word n-grams are problematic for high-morphology languages. This project is implementing English to Czech translation, so chrF metric may be better, because Czech language is also considered to be high-morphology language [com23b]. SacreBLEU[2] library was used to compute the chrF2 metric, which is one version of the chrF metric, over the test split.

4.1.2 BLEU

BLEU (BiLingual Evaluation Understudy)[com23a] is metric for machine translation evaluation, which calculates the similarity between a translation output and reference translation using word n-gram precision. SacreBLEU[3] library was used to compute the BLEU metric over the test split.

4.2 Experimental results

Evaluation was performed over the test split of the dataset with both greedy decoding and beam search decoding. Both evaluations results were compared to translation references using chrF2 and BLEU metrics. Sacrebleu library outputs the result scores

[1] https://keras.io/examples/nlp/
[2] https://github.com/mjpost/sacrebleu
[3] https://github.com/mjpost/sacrebleu

in range from 0 to 100. Evaluation with greedy search resulted in chrF2 = 47.03 and BLEU = 42.02. Evaluation with beam search resulted in chrF2 = XXX and BLEU = XXX. Results of chrF2 metric are slightly better than BLEU, probably because Czech is considered high-morphology language, so character level ngrams may represent the quality of translation better. Table 4.2 shows some examples of translated sentences.

English Sentence	**Czech Translation**
This is unacceptable: I shall vote against.	To je nepřijatelné: hlasuji proti.
I voted in favor of this report with conviction.	Hlasoval jsem ve prospěch této zprávy.
We have already heard about China.	O Číně jsme již slyšeli.
That is why, if we enter a new era of bilateral trade agreements, these should include guarantees and principles, some of which are mentioned in Mr. Martin's report.	To je důvod, proč v době, kdy zahájíme novou éru dvoustranných obchodních dohod, by měly být tyto záruky a zásady, z nichž některé jsou zmíněny ve zprávě pana Martina.

4.3 Inference speed

The greedy search was approximately 5 times faster than beam search in inference. This may be effected by effectivity of my implementation of algorithms, which has a lot of room for optimizations, but I didn´t consider it as important for goals of this project.

5 Conclusion

As can be seen in table 4.2, translations in most cases capture the semantics of original sentences quite well and the translations are mostly syntactically correct czech sentences, but there is definitely room for improvement. The experiments also showed, that model is better on formal language than informal, probably due to the used dataset, which consists mainly of highly formal sentences. Creating better dataset consisting of texts from multiple sources, both formal and informal, may lead to better results. There's also room for improvement in optimization of decoding algorithms for better inference speed.

References

[com23a] Machine Translate community. Bleu: Evaluation metric based on n-gram precision. https://machinetranslate.org/bleu, 2023.

[com23b] Machine Translate community. chrf: Character-level f-score. https://machinetranslate.org/chrF, 2023.

[Koe05] Philipp Koehn. Europarl: A parallel corpus for statistical machine translation. Phuket, Thailand, 2005.

[Ten23] Tensorflow authors. Tensorflow documentation. online, 2023.