Incorporating Attention Mechanisms in RNN-based Encoder-Decoder Models

Patrick Hiemsch
pathi619@student.liu.se

Abstract

In this study, we investigate the performance and behavior of the Attention mechanism proposed by (Bahdanau et al., 2016) in a machine translation task from German to English. We implement an RNN-based Encoder-Decoder network with Attention and visualize the attention weights generated during translation for selected input sequences. Our analysis reveals that the introduced Attention mechanism is able to learn the alignment of source and target words. It provides valuable insights into the model's behavior and the translation process, by highlighting the parts of the input that are most relevant for generating the corresponding output, contributing to the explainability of the model and it's results.

1 Introduction

Machine translation has been a hot topic in the field of natural language processing (NLP) for many years. In recent years, the use of neural networks has revolutionized machine translation and led to the development of more accurate and efficient translation models. One of the key innovations in this field has been the introduction of the attention mechanism, which allows the model to focus on certain parts of the input sequence when generating the output sequence. The attention mechanism has been shown to significantly improve the performance of machine translation models. Especially the transformer architecture innovated this field when Vaswani et al. (2017) published their paper on attention.

In this research project, we will investigate one of the earlier approaches to attention proposed by Bahdanau et al. (2016), which also contributed to the development of the transformer architecture. We implement an Encoder-Decoder neural network with attention for machine translation from German to English, based on the seminal work . We investigate and visualize the attention weights generated by the model for different input sentences to gain a better understanding of how the attention mechanism works in practice.

2 Theory

In the field of Neural Machine Translation, especially before the rise of the Transformer architecture, a widely used method was the training of an Encoder-Decoder architecture as described in Sutskever et al. (2014). In this class of models, both, the encoder and decoder networks are based on Recurrent Neural Networks (RNN), using Gated Recurrent Units (GRU) or Long Short-Term Memory (LSTM) specifically.

The encoder network is unrolled over the source sentence, computing a single hidden state, which is then fed to the decoder. The decoder then takes this vector representation of the source sentence together with the start of sentence <SOS>token, to produce the next word in the target sequence. This process terminates when the end of sentence token <EOS> is produced (or when the length exceeds a certain maximum threshold).

As Bahdanau et al. (2016) describe in their paper, a problem that can arise in the context of these encoder-decoder architectures is that the whole source sentence needs to be compressed into a single vector. It is known, that unidirectional RNNs suffer from a recency bias. This refers to the fact that more recently processed tokens in the sequence will be represented stronger in the final hidden state compared to the ones closer to the beginning of the sequence. This is especially true for longer sequences, which according to Cho et al. (2014) is the reason that these approaches have difficulty translating long sentences. This effect can be mitigated by using a bi-directional architecture and concatenating the last hidden states of both directions, but it still remains a challenge.

Bahdanau et al. (2016) propose a method that frees the encoder network from the need to compress the whole source sentence into one single

vector. They introduce the concept of attention. This proposed part of the network takes all hidden states produced in the encoding phase after unrolling the network over the source sentence and computes a weighted fixed-length summary vector c_i. The weights of this attention layer are trainable parameters, which can be learned as usual by backpropagation.

To compute c_i as in equation 1, all hidden states from the encoder step from each of the source sentence tokens are summed together according to the computed attention weights α_{ij}, which are normalized to sum up to one. So c_i is essentially a weighted mixture of every hidden state of the source sentence. To get the attention weights α_{ij}, each hidden state h_j of the corresponding source token at position j is fed to an alignment model a together with the latest hidden state of the decoder network s_{i-1}. See equation 2 for details. This hidden state is yielded by the decoder after producing the latest prediction, so word s_{i-1} is the hidden state that followed decoding word $i-1$ in the generated target sentence. In the first step this will be the last hidden state of the encoder network. The alignment model computes the energy e_{ij} which scores how important the token at position j is for predicting the next word in the sequence. To get the attention weights from e_{ij}, each of them will be normalized through a Softmax-Layer as showcased in equation 3. This ensures that the α_{ij} values form a probability distribution over all steps in the source sentence. Important to note is also, that the energy scores for padding tokens are manually set to a large negative number (in our implementation `-1e10`) which will lead to a (almost) zero attention score after applying softmax. The alignment model can modeled as a feedforward neural network itself.

Some more details of the exact Encoder-Decoder architecture used in the experiments will be mentioned later when describing the method. For further details on the theoretical aspects of RNNs and attention, have a look at Bahdanau et al. (2016) and Sutskever et al. (2014).

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j \tag{1}$$

$$\alpha_{ij} = \frac{\exp\left(e_{ij}\right)}{\sum_{k=1}^{T_x} \exp\left(e_{ik}\right)} \tag{2}$$

$$e_{ij} = a\left(s_{i-1}, h_j\right) \tag{3}$$

3 Data

In the their paper Bahdanau et al. (2016) use a bilingual text corpus from ACL WMT '14[1], which contains English-French sentence pairs. The total number of words included in their combined corpus is 348M.

For this project, we work with a dataset that originates from the same source, ACL WMT '14, but is targeted towards the translation from German to English and vice versa. It includes the proceedings of the European Parliament from 1996 to 2011. The dataset can be downloaded for free from the ACL WMT website[2].

Without preprocessing each corpus contains roughly 44.6M words in 1,920,209 sentences. Since the scope of the project is more on exploring the network architecture and investigating the effect of attention, the preprocessing is kept simple: Both text sets are combined into one big dataset. Then all sentence pairs are removed which contain none, more than 30 or less than 4 words. This lead to a full corpus of 1,084,301 sentences.

In a final step, the full corpus is randomly split into training, validation and test set with ratio 8:1:1.

4 Method

As described in the theory part, the network architecture used for the experiments in this project are closely related to Bahdanau et al. (2016). The encoder network implemented is a one-layer bidirectional GRU based architecture. Its forward pass not only returns the hidden states for each step token in the source sentence (specifically a stacked vector of each GRU direction) and the summary of the source sentence as the initial hidden state for the decoder (modeled as linear layer), but it also returns a padding mask. This mask has the same dimensions as the computed energy by the attention layer, so `(batch_size, padded_seq_len)`. The padded sequence length refers to the length of each sequence in a data mini-batch, which needs to be the same for one batch in order for the computational graph to be built and executed. Some of the sequences will be padded with <pad> tokens in the end of it, to match this length. It is crucial that the padding is done in the end since we pack the sequences when we feed them to the different layers,

[1] https://www.statmt.org/wmt14/translation-task.html

[2] https://www.statmt.org/wmt13/training-parallel-europarl-v7.tgz

which expects back-padded representations. The mentioned mask will then be used to set the energy scores e_{ij} manually to a large negative number to eliminate their contribution to c_i. The attention layer takes the mask, s_{i-1} and all encoder hidden states h_j as arguments and yields the hidden state c_i after applying attention. The alignment model used is a feedforward network with one hidden layer of equal size as the decoder hidden dimension. The decoder is similar in structure to the encoder, a one-layer GRU but with only one direction. It yields the next word logits and its current hidden state s_i.

For training we use an adapted version of teacher forcing. In general, teacher forcing as originally proposed by Williams and Zipser (1989) refers to always feeding the decoder the true one step before token y_{i-1} instead of $\hat{y}_{i-1}$. In our approach we use a probabilistic version of it that for each next step of the target sequence decides on whether to use the real label y_{i-1} or the prediction $\hat{y}_{i-1}$. This is a simplified version of the method described in Bengio et al. (2015), which lets the teacher forcing probability decay over time as the networks learning process progresses.

Regarding translation, Bahdanau et al. (2016) use beam search in their experiments which is a heuristic approach to maximize the joint probability of the target sentence token by considering some steps in the future (depending on beam size) and comparing the different joint probabilities

$$p(\mathbf{y}) = \prod_{t=1}^{T} p\left(y_t \mid \{y_1, \cdots, y_{t-1}\}, c\right).$$

For simplicity, in our implementation the system will just act greedily with respect to the token with the highest probability for this step.

As the time horizon is limited and the focus of the project is to understand, implement and experiment with the attention layer as part of a RNN-based Encoder-Decoder architecture, only this class of models will be studied rather than comparing them with non-attention based RNN-based Encoder-Decoder networks. This is why we will only use the training set to train the models and use the validation set to measure validation loss in order to early stop and load the best performing model based on this validation metric for attention visualizations.

To build the vocabularies for the source, German, and the target, English, we use the `torchtext` library. It is important to note in this context that especially the objects `Field` and `BucketIterator` are used, which are not longer part of the newest package version. We use version `0.6.0`. For both vocabularies, the vocabulary size is set to 8K. This choice seems reasonable when looking at the results from Gowda and May (2020). They argue that for medium data sizes under 1.3M sentences, a vocabulary size of 8K is reasonable.

The influence of the padding tokens on gradient computations and attention weights is eliminated in our implementation as stated before. This is on achieved by using packed sequences for passing to the GRU in the Encoder model, by specifying to ignore the padding index in the target batch while computing the Cross-Entropy-Loss and by implicitly setting attention weights to 0 for all padding tokens.

5 Results

For our experiments we use the following shared hyperparameters: The vocabulary size is fixed to 8,000, the embedding dimension is set to 256 for both languages, both encoder and decoder have only one GRU layer, no dropout is used and the hidden dimensions are fixed to 512 (only in the last experiment 5 , 1024 was used).

For Experiment 1 the learning rate is set to `1e-4`, batch size is 128 and teacher forcing ratio equals 0.8.

For Experiment 2 the learning rate is set to `5e-4`, batch size is 256 and teacher forcing ratio equals 0.8.

For Experiment 3 the learning rate is set to `5e-4`, batch size is 128 and teacher forcing ratio equals 0.5.

For Experiment 4 the learning rate is set to `1e-4`, batch size is 128 and teacher forcing ratio equals 0.5.

For Experiment 5 the learning rate is set to `1e-4`, batch size is 80 as in the paper and teacher forcing ratio equals 0.5 and hidden sizes are 1024.

We evaluate each of the models based on their training and validation CrossEntropyLoss. The evaluation procedure is performed every after reaching 25% epoch size, so 4 measurements per epoch.

From the plots we can see that all models seem rather unstable with periodical changes in the loss. But in general we can see that the models that use a teacher forcing ratio of 0.5 show better results on the validation data. Due to less teacher forcing they

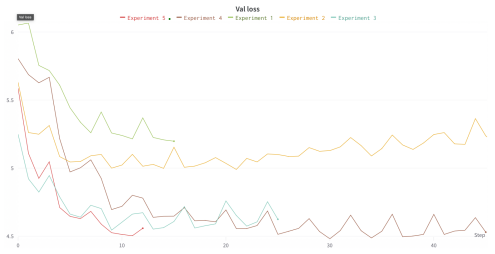

Figure 1: Validation loss curves for experiments

are trained closer to what they will be evaluated on, since during evaluation teacher forcing is set to 0. This improves learning. But as mentioned earlier, her is lies a lot of room for improvement with a teacher forcing schedule that decreases over time. As we can see for the longer trained models, after around 6 epochs they start to overfit the training data indicated by a statically rising validation loss.

Here are a few examples of short sentences from the validation set and their prediction by the best validated model from experiment 4.

Example translations

Ger: "Dennoch ist das ein wichtiges Kriterium für die Europäische Union."

Gold: "Nonetheless, these are important criteria for the European Union."

Model4: "However, this is an important criterion for the european union."

Ger: "Leider war das nicht der Fall."

Gold: "That was not, however, the case."

Model4: "Unfortunately, that was not the case."

Ger: "Vielen Dank für Ihre Antwort, Frau Kommissarin."

Gold: "Commissioner, thank you for your response."

Model4: "Thank you, commissioner, for your reply."

On these short examples, even though the translation is not identical with the gold standard label, they seem to be reasonable. Of course this should be further validated on more and also longer examples while also measuring the BLEU score on (maybe even with some additional gold labels with the same meaning), to really get an unbiased estimate of the translation power.

Attention visualization

Figure 2 shows the attention weights and alignment for the first example sentence from before. We can see, that for example we have a strong alignment of "case" and "Fall" which is a direct translation and expected. But for example for the German word "war" which means "was" in English, there is only a small attention score. Also "the" which would be expected to pair with "der" does only show attention for the noun "Fall".

The second figure shows a more interesting relationship. We can observe that we have string alignments on the diagonal, which is expected since the word ordering in this sentence is pretty similar in both languages.

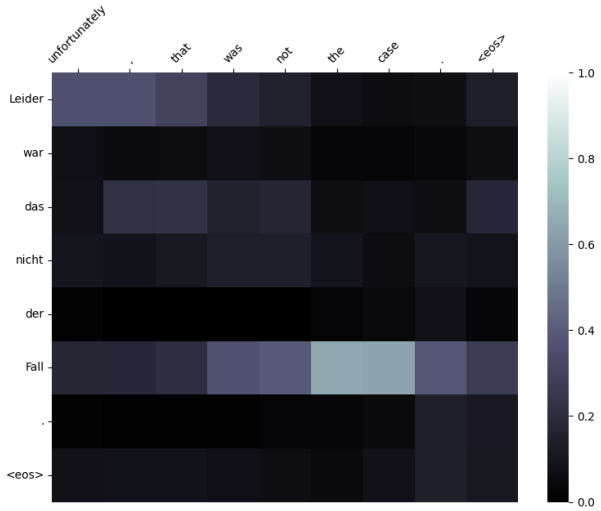

Figure 2: Attention weights for example 1

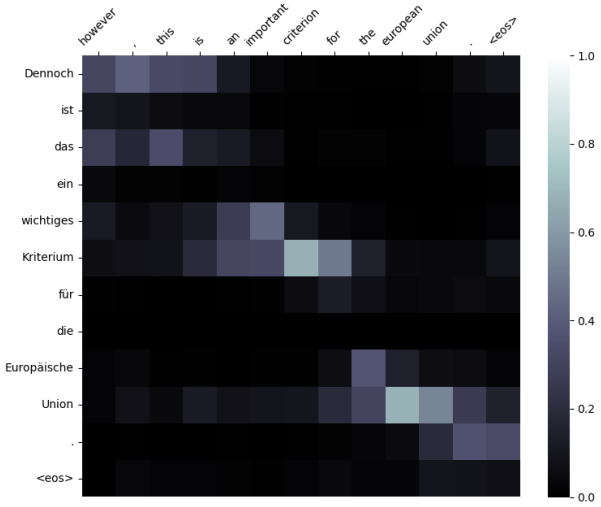

Figure 3: Attention weights for example 2

6 Discussion

As we can see, the model did not show exactly the results we were expecting, as it did not work very well for long sentences and the attention weights, even though showing reasonable alignments, also show some alignment gaps as shown in figure 2. We want to discuss some of the many factors that could influence the results we have seen:

Since the time horizon was limited and a lot of time was spend to implement the pipelines, there was not enough time to explore the vast hyperparameter space in a sufficient way. More experiments with a more sophisticated hyperparameter tuning pipeline could have been beneficial. The corpus of the dataset with over 1M sentences is fairly big given the resources of our project. Even though GPU training was used, the training times and memory requirements of the models are high. In their paper Bahdanau et al. (2016) trained all of their models for over 100 hours. Their best performing model RNNsearch-50 was even trained for over 10 subsequent days (252 hours) on a bigger GPU. As mentioned above, the use of Beam Search could also improve the quality of the translations by heuristically maximizing the joint probability of the translation output words. This would be interesting to include in a further stage of this project.

The hidden state size also holds potential for further investigation. The models in our experiments were mostly initialized with a hidden state size of 512 which is already fairly big. But the training and validation loss show a lot of (periodical) spikes and seem rather unstable. This could be due to the fact that the training loader yields batches of sentences of similar lengths to decrease padding and increase computational efficiency. This could be one if the cases for the seen behaviour of the losses. For example the model adjusts parameters on shorter sentences and then tries to adapt them based on the longer bathes afterwards. But if the model is not big enough to be able to capture the long term relationships, which is especially influenced by the hidden state dimensions, this could lead to undesirable gradient changes, which are then again changed by the smaller sentences, then by the longer ones again, etc. So a more detailed analysis of the hidden dimension sizes, especially larger ones, could be beneficial for the dataset at hand with a high average sentence length of roughly 23.5. Furthermore, Bahdanau et al. (2016) use a Maxout layer as proposed by Goodfellow et al. (2013) before the computation of the Softmax target token probabilities, in our model the target output layer is just a Linear layer without hidden layer which maps the stacked vector of decoder hidden state s_{i-1}, the attention encoder hidden state c_i and the most recent embedded target token (depending on teacher forcing y_{i-1} or $\hat{y}_{i-1}$) to the target vocabulary size. This is also a factor to keep in mind for an extension of the project.

The same is true for dropout regularization which could decrease possible overfitting and stabilize training and generalization performance. This is especially true for large models with high dimensional embeddings/hidden dimensions.

The embedding layer for our model was kept at 256 for all runs. But Bahdanau et al. (2016) map each word to a 620 dimensional vector, which could also help to learn better representations of words. This could clearly also represent an opportunity for improvement of the architecture.

The vocabulary size on the other hand was kept fixed at 8,000 to reduce model parameters. The paper uses 30,000. Even though we presented a reason for the choice of vocabulary size, this is a parameter that should be investigated further.

All the mentioned factors above refer to model architecture and the increase of parameter size. To connect this to our main research interest for the project, the attention mechanism, all these changes present an opportunity for the model to learn more efficiently and more complex long-term relationships in the different vector spaces. This is of course in particular true for the attention layer which relies on good representations of word contexts in the hidden dimensions to be able to align words in a meaningful way.

Something that could be investigated as well is using pretrained embeddings to initialize the source and target embedding layers and either freeze them for training or fine-tune them based on the task at hand. This offers a opportunity for more stable training right from the start.

The example models trained in the project were also not tested properly due to the time horizon accompanying it. Especially a comparison of a regular non-attention based Encoder-Decoder RNN architecture to a similar architecture with the reviewed attention based architecture form the same model class would have been interesting.

Our model was trained and evaluated just by measuring the training and validation loss. This

loss is the CrossEntropyLoss of the top-prediciton of the model compared to the real target token. Even though this represents a valid measurement for predictive accuracy of the model and gives us a feel for the likelihood of the target sentences given the source sentences and our model, it is not always a good indicator of the quality of the generated text. The BLEU score on the other hand proposed by Papineni et al. (2002) is a more holistic measure. It compares the generated text with one or more reference texts, which helps to ensure that the generated text is semantically and syntactically correct and meaningful. So including it in the evaluation procedure could be a valuable direction for an extension of the project.

One more bias we want to point out is the fact that the dataset used contains transcripts of the european parlament, which is biased towards certain topics and a certain style of language. To train a more general model and to especially to learn a better word alignment model with attention, a corpus with more variation should be used.

7 Conclusion

As the experiments have shown, the attention mechanism by Bahdanau et al. (2016) can be an interesting extension to the usual Encoder-Decoder RNN-based architecture. It can potentially help the machine translation network yield better performance since it alleviates the burder of the encoder network to encode the whole source sentence in one single vector representation. By learning to translate and to align at the same time, it also boosts explainability of the translation model since attention weights are a clear indicator for what source sentences influence the networks prediction of the particular output word the most. All in all it is a promising approach but as discussed, there is still great potential for improvement.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2016. Neural Machine Translation by Jointly Learning to Align and Translate.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches.

Ian J. Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. 2013. Maxout Networks.

Thamme Gowda and Jonathan May. 2020. Finding the Optimal Vocabulary Size for Neural Machine Translation. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3955–3964, Online. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Ronald J. Williams and David Zipser. 1989. A Learning Algorithm for Continually Running Fully Recurrent Neural Networks. *Neural Computation*, 1(2):270–280.