Assignment 2: Implementing Neural Machine Translation with Attention

Aniruddha Bala
Indian Institute of Science
Bangalore, India
aniruddhab@iisc.ac.in

1 Introduction

In this assignment I have implemented the sequence to sequence model for neural machine translation task with four different types of attention. The translation task has been carried out on two different pairs of languages English to German and English to Hindi.

2 Dataset

To train the model on English to German translation task (henceforth called as task 1) I have used the Europarl-v7 parallel corpus and for English to Hindi translation task (task 2) I have used the HindEnCorp. For validation and testing I have used the newsdev and newstest parallel corpuses. The europarl dataset for task 1 consists of around 2M samples while the HindEnCorp corpus for task 2 consists of around 0.27M samples. For ease of training, in the task 1 I have subsampled the training set and used around 1.3 lakh samples for training after preprocessing. I also found a few inconsistencies in the Europarl data, for instance at some places if there is a blank line in one corpus the corresponding line number in the parallel corpus is non blank and hence sentences one line above and below the blank line are misaligned, therefore, in such cases I have removed all the three lines from both corpuses.

3 Model Architecture

To build the sequence to sequence model I have used an encoder-decoder architecture. In my model the encoder is a 2 layered bidirectional LSTM network and the decoder is a single layered unidirectional LSTM network. The word embeddings fed as an input to the LSTMs are 100 dimensional vectors and are randomly initialized. I have done the experiments with two different settings of hyperparameters. Here I describe the model with LSTM hidden size chosen as 128. The hidden representation formed from the encoder network is 256 dimensional (comprising of hidden units of forward and backward LSTM), it is passed through a neural network which projects it to a 128 dimensional vector before feeding it to the decoder. Without self-attention, if attention is calculated on the encoder states which are 256 dimensional vectors (with hidden states of forward and backward LSTMs concatenated), then the context vector formed is also 256 dimensional. If decoder attention is also used then we also concatenate the attention calculated on the decoder hidden states and hence the context vector is 384 dimensional. If self attention is used then, first one layer of attention is calculated on top of each encoder and decoder states followed by an intra attention on encoder and decoder. To make the prediction at time step t the hidden representation of the decoder at time step t and the context vector concatenated is passed through a neural network, and softmax is applied on the logits obtained to get the probabilities. Below I have mentioned the different formulae used for attention.

Additive attention:

$$f_{att}(\mathbf{h}_i, \mathbf{s}_j) = \mathbf{v}_a{}^\top \tanh(\mathbf{W}_a[\mathbf{h}_i; \mathbf{s}_j])$$

Multiplicative attention:

$$f_{att}(h_i, s_j) = h_i^\top \mathbf{W}_a s_j$$

Scaled dot product attention:

$$f_{att}(h_i, s_j) = (h_i^\top \mathbf{W}_a s_j)/\sqrt{d_k}$$

Key value attention:

$$\mathbf{a}_i = \text{softmax}(\mathbf{v}_a{}^\top \tanh(\mathbf{W}_1[\mathbf{k}_{i-L}; \ldots; \mathbf{k}_{i-1}] + (\mathbf{W}_2 \mathbf{k}_i)\mathbf{1}^\top))$$

Context vector and attention outputs:

$$\begin{aligned} \mathbf{c}_i &= \sum_j a_{ij} \mathbf{s}_j \\ \mathbf{a}_i &= \text{softmax}(f_{att}(\mathbf{h}_i, \mathbf{s}_j)) \end{aligned} \tag{1}$$

Self Attention is calculated by first projecting the hidden states into query and keys as follows:

$$q_{proj} = W^Q q$$

$$k_{proj} = W^K k$$

4 Methodology

The raw data is processed by removing symbols other than the language specific characters and a few punctuation marks. The sentences with length less than certain threshold are removed from the dataset. The vocabulary for a language is obtained from the training set of that language by keeping only k most frequent words. The unknown words in the data are replaced with '<UNK>' token. And to the end of each sentence an end of sentence '<EOS>' tag is appended. To feed the sentences as batches the sentences less than the maximum length sentence are padded with the '<PAD>' tags in the end. All the words in the sentences are then converted to their respective indices. The processed parallel data is then fed to the model in batches and the whole model is jointly trained by minimizing the negative log likelihood loss. Different models are trained for different types of attention and with and without self attention. The trained models are evaluated based on the bleu scores obtained on the validation and test sets. The model is implemented using the Pytorch framework. The code is available at https://github.com/ani555/E1-246-Assignment2.

5 Experimental Setup

I have tried using two different types of hyperparameter settings for for my experiments. Those are shown in tables 1 and 2.

6 Results

The result section contains the results obtained on task1 and task2. I have reported the bleu scores obtained on the test set for different models. The bleu scores are corpus bleu scores calculated on the WMT test set and test set sampled from Europarl data. The two figures below show the variation of loss and variation of validation perplexity with the number of samples seen. Here perplexity is calculated by taking the exponential of average log likelihood loss on validation set.

Hyperparameter	Value
batch_size	128
embed_size	100
hidden_size	128
enc_num_layers	2
dec_num_layers	1
proj_size	64
dropout	0.5
vocab_size	30000
epochs	10
learning_rate	0.01

Table 1: Hyperparameter setting 1

Hyperparameter	Value
batch_size	256
embed_size	100
hidden_size	64
enc_num_layers	2
dec_num_layers	1
proj_size	32
dropout	0.2
vocab_size	20000
epochs	10
learning_rate	0.01

Table 2: Hyperparameter setting 1

7 Observations and Conclusions

In table 3 we observe that the bleu scores are very less, the main reason for this was overfitting as we can see from the loss curves in figure 1. When we decrease the number of parameters in the model the bleu scores significantly improve. Also increasing the training data helped. A comparison of different attention mechanisms show that additive attention works the best when hidden size is 64 while scaled dot product attention works best for larger size hidden units. The implementation of scaled dot product is almost similar to that of multiplicative attention with the only difference of a scaling factor. Scaling factor plays a vital role in cases where there are more number of hidden units where normal dot product will result in high values causing unstability. Self attention, surprisingly was found to be not so effective in this case, though it helped improve performance for hyperparameter setting 2. For self attention I have used only one head, to improve performance multiple headed self attention with better hyperparameter

Attention type	**Bleu score** WMT Test	**Bleu score** Europarl Test
seq2seq + additive attn	4.43	7.18
seq2seq + multiplicative attn	4.64	7.36
seq2seq + scaled dot-product attn	4.76	8.52
seq2seq + key value attn	2.71	3.99
seq2seq + additive + self attn	4.35	6.70
seq2seq + scaled dot product + self attn	2.99	3.76

Table 3: Bleu scores for task 1 on the test set for model trained on 1.1 lakh samples from europarl dataset for hyperparameter setting 1

Attention type	**Bleu score** WMT Test	**Bleu score** Europarl Test
seq2seq + additive attn	7.15	12.03
seq2seq + multiplicative attn	7.10	11.55
seq2seq + scaled dot product attn	5.46	9.75
seq2seq + key value attn	5.62	8.95
seq2seq + scaled dot product + self attention	6.23	8.93
seq2seq + scaled dot product + intra temporal attn	5.72	8.94

Table 4: Bleu scores for task 1 on the test set for model trained on 1.3 lakh samples from europarl dataset for hyperparameter setting 2

Attention type	**Bleu score WMT Test**
seq2seq + additive attn	2.02
seq2seq + scaled dot product attn	1.37

Table 5: Bleu scores for task 2 on the test set for model trained on HindEnCorp dataset for hyperparameter setting 2

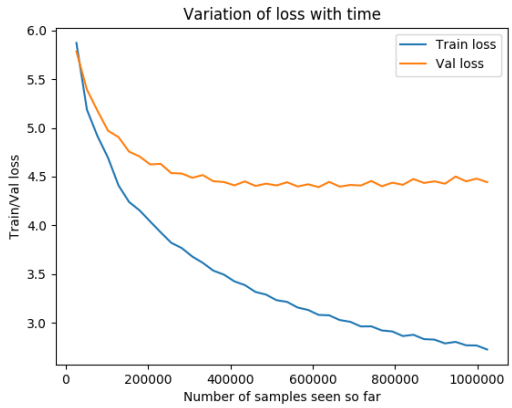

Figure 1: Variation of training and validation loss with time

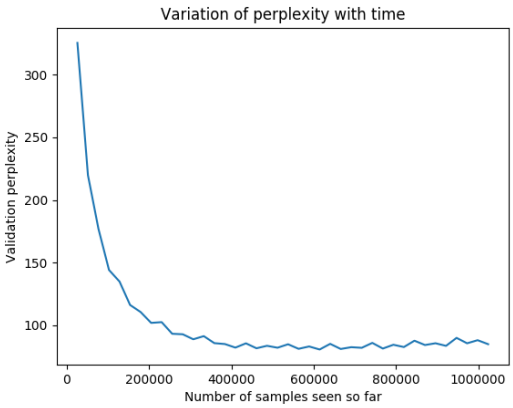

Figure 2: Variation of validation perplexity with time

settings can be explored. I have also included a result with intra temporal attention as proposed by Paulus et al. (2017) which penalizes the input tokes that have obtained high attention in the past. This type of attention helps by avoiding repetitive phrases in the translation.

It is observed that Machine Translation is a data hungry task, the more data we can train our models on the better results we obtain. By training on just 1.3 lakh samples for task 1 we are able to obtain bleu scores around 7.

Source sentence	Translated sentence
The president of the commission agreed to this proposal	auszurufen präsident der kommission legte diesem vorschlag dieses vorschlags der kommission dieses vorschlags .
Why have the staircases not been improved since my accident?	warum haben wir die <UNK>nicht gebessert ?
In our view, information is not enough	informationen sind nicht genug .
Of course we want weapons of mass destruction out of Iraq.	wir wollen also massenvernichtungswaffen errichten .

8 Some Translations and Visualizations

Table shows some translations from source sentence english. The figures below show some attention visualizations obatained while translating a sentence.

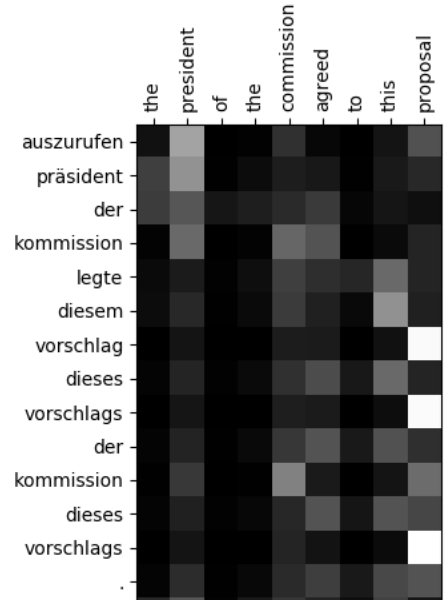

Figure 3: Attention visualization on the encoder states

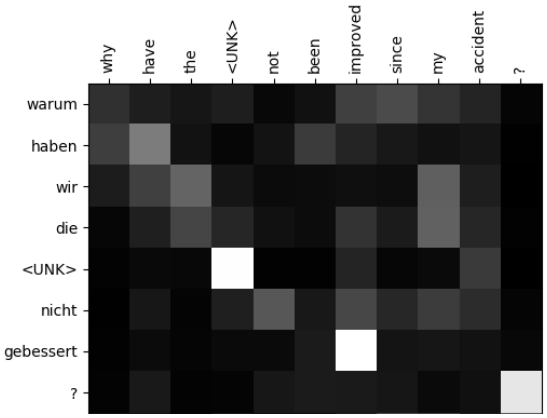

Figure 4: Attention visualization on the encoder states

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate.

Michał Daniluk, Tim Rocktäschel, Johannes Welbl, and Sebastian Riedel. 2017. Frustratingly short attention spans in neural language modeling.

Yang Liu and Mirella Lapata. 2018. Learning structured text representations. *Transactions of the Association for Computational Linguistics*, 6:63–75.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.