HW3 : NMT

Noah Golowich and Jesse Zhang

March 2, 2018

1 Introduction

In this note we describe our implementation of a Seq2seq recurrent neural network to perform neural machine translation, as well as a model incorporating attention. We implement beam search and visualize the attention distribution for the latter model.

We use the IWSLT German-English dataset of TED talks, and perform translation from German to English. In addition to implementing a baseline Seq2seq model as in Sutskever et al. (2014) and a model using attention somewhat similar to that in Bahdanau et al. (2014), we implement many variations and extensions, some of which are used in Luong et al. (2015).

2 Problem Description

We consider the following problem: given a source vocabulary $\mathcal{V}_{src}$ (of German words), a target vocabulary $\mathcal{V}_{trg}$ (of English words), and a source sentence $\boldsymbol{x} = (x_1, \ldots, x_S) \in \mathcal{V}_{src}^S$, we aim to produce a translation $(y_1, \ldots, y_T) \in \mathcal{V}_2^T$.

In evaluating our translations, for simplicity, we do not use the BLEU score Papineni et al. (2002), as is standard, but instead evaluate based on perplexity. In particular, for each word $1 \leq i \leq T'$ of the target sentence, our network receives words $y_1, \ldots, y_{i-1}$ (in addition to the source sentence $(x_1, \ldots, x_T)$) and based solely on this information must predict y_i:

$$p(y_t | \boldsymbol{x}, y_1, \ldots, y_{t-1}) = f(y_t; \boldsymbol{x}, y_1, \ldots, y_{t-1}),$$

where f is a function computed by two recurrent neural networks (to be described in subsequent sections). We train to minimize the expected negative log-likelihood of the correct word y_t, and report perplexity values $\text{PPL}(f)$, where

$$\text{PPL}(f) := \exp\left(\frac{1}{T}\sum_{t=1}^{T} -\log f(y_t; \boldsymbol{x}, y_1, \ldots, y_{t-1})\right). \tag{1}$$

Throughout this note we use $1 \leq s \leq S$ to index into words in the source sentence, adn $1 \leq t \leq T$ to index into words in the target sentence.

3 Model and Algorithms

In this section we describe a baseline seq2seq model, as well as the version incorporating attention.

3.1 Seq2seq

For our Seq2seq model, we mostly follow the implementation of Sutskever et al. (2014).

Our Seq2seq network consists of an encoder and a decoder, each implemented with a multi-layered LSTM:

- The encoder LSTM, upon receiving an input sentence $(x_1, \ldots, x_S) \in \mathcal{V}_{src}^S$, uses the words to index into a $D \times |\mathcal{V}_{src}|$ word embeddings matrix $W^{(E)}$ (which is learned), giving a sequence $(W_{src}^{(E)} x_1, \ldots, W_{src}^{(E)} x_S) \in \mathbb{R}^D$. (Here x_t denotes the 1-hot encoding.) It then computes the output $(v_1, \ldots, v_S) \in (\mathbb{R}^H)^S$ and final hidden state $(h_S, c_S) \in (\mathbb{R}^{HL})^2$ of a standard L-layer LSTM, where H denotes the hidden layer size of the LSTM. The vector $c_S \in \mathbb{R}^{HL}$ denotes the final cell state, whereas h_T denotes the final hidden state of the LSTM (note that this terminology overloads the meaning of "hidden state", so we will limit it from here on). In particular,

  $$(v_1, \ldots, v_S), (h_S, c_S) = LSTM_{enc}(W_{src}^{(E)} x_1, \ldots, W_{src}^{(E)} x_S), \tag{2}$$

  where H denotes the hidden layer size of the LSTM. The above notation means that in the s-th time step $W_{src}^{(E)} x_s$ is fed as input into $LSTM_{enc}$.

  As in Sutskever, we experiment with feeding the input sentence (equivalently, the embeddings) into the LSTM in reverse order, so that the LSTM instead computes $LSTM_{enc}(W_{src}^{(E)} x_S, \ldots, W_{src}^{(E)} x_1)$.

- The decoder LSTM has the same hidden size H and number of layers L as the encoder LSTM, and uses as its initial hidden layer the final hidden layer (h_S, c_S) of the encoder. We feed as the first word to the decoder LSTM a special token $\langle \text{sos} \rangle$ denoting start-of-sentence.

  The decoder then computes, for $1 \leq t \leq T$, the embedding $W_{trg}^{(E)} y_{t-1}$ of the $t-1$-th word in the target sentence, and then runs the LSTM,

  $$z_t, (h_t, c_t) = LSTM_{dec}(W_{trg}^{(E)} y_{t-1}, (h_{t-1}, c_{t-1})),$$

  where y_{t-1} is the true target word and h_{t-1}, c_{t-1} represent the hidden state of the previous iteration. Note that, by the definition of the LSTM, we will have $z_t = h_t$ at each time step, but we use separate letters to emphasize the difference in roles of z_t and h_t.

  We then predict the probability of the next target word as:

  $$p(y_t | \boldsymbol{x}, y_1, \ldots, y_{t-1}) = Softmax(W[z_t; h_S] + b)_{y_t},$$

  ? where W is a learned matrix of parameters. Note that one may include the last cell state c_S into this prediction as well, but we chose not to experiment with this option.

3.2 Attention model

Next we describe how we incorporated attention into our baseline Seq2seq model. We use soft attention, as in Sutskever et al. (2014); Luong et al. (2015).

The encoder for our attention model is identical to the encoder for the non-attention model, except we experiment with a bidirectional LSTM encoder in addition (which means that the size of each of $v_1, \ldots, v_S, h_S, c_S$ are all doubled).

The probabilities produced by the decoder for the attention model are given by:

$$p(y_t|\boldsymbol{x}, y_1, \ldots, y_{t-1}) = Softmax(W_1 \cdot [z_t, d_t] + b_1), \tag{3}$$

where $W_1 \in \mathbb{R}^{|\mathcal{V}_{trg}| \times 2H}, b_1 \in \mathbb{R}^{|\mathcal{V}_{trg}|}$ are learned parameters[1], and z_t is the output of the decoder LSTM, given by

$$z_t, (h_t, c_t) = LSTM_{dec}((W_{trg}^{(E)} y_{t-1}, (h_{t-1}, c_{t-1})),$$

where h_t, c_t denote the hidden and cell states of the LSTM at each time step. Finally, d_t in (3) is a **context** vector, computed as follows: d_t is a weighted linear average over source states, $d_t = \sum_{s=1}^{S} \alpha_{st} v_s$, where $v_1, \ldots, v_S$ are as in (2). The weights α_{st} are given by a score function,

$$\alpha_{st} = \frac{\exp(score(v_s, z_t))}{\sum_{s'=1}^{S} \exp(score(v_{s'}, z_t))},$$

where

$$score(v_s, z_t) = z_t^T (W v_s)$$

is a bilinear form on the source state v_s at time step s and the hidden state z_t at time step t. In our unidirectional model we fix $W = Id$, which is possible, since v_s and z_t have the same dimensionality (both are in $\mathbb{R}^H$, where H denotes the hidden size). When the encoder is bidirectional, however, we have $W \in \mathbb{R}^{H \times 2H}$, and we make W a learned parameter.

Differences from Bahdanau et al. (2014); Luong et al. (2015). The final layer of our decoder in (3) is simpler than that in Bahdanau et al. (2014); Luong et al. (2015): the former uses the embedding of the previously generated word in addition to the context d_t and the output state z_t to predict the next target word, and also adds a single maxout hidden layer. The latter, on the other hand, adds an additional tanh layer, so that it computes $Softmax(W_2 \cdot \tanh(W_1 \cdot [z_t, d_t] + b_1) + b_2)$ as opposed to (3); we experimented with such a layer but found that it failed to improve performance on our dataset. Bahdanau et al. (2014) also use a MLP to compute the attention scores $score(v_s, z_t)$, as opposed to our dot-attention.

4 Experiments

4.1 Hyperparameters for Seq2seq

For our baseline (no attention) Seq2seq model, we set the hidden size of the hidden layers of both the encoder and decoder LSTM to 500, and the number of word embeddings to be 500. We use stochastic gradient descent on the negative log likelihood for training, with an initial learning rate of 0.7, and which was decayed by a factor of 2 for each epoch, starting at the 6th epoch. Unlike Sutskever et al. (2014), we do not reverse the input sentence when training; we tried doing so, but the validation accuracy did not significantly improve. We hypothesize that this may be because our sentences are generally shorter than those in Sutskever et al. (2014). (Moreover, we spent more effort tuning parameters in our attention model.)

We trained the Seq2seq model for a total of 6 epochs, which was chosen by using early stopping (on the validation set).

[1]For a bidirectional encoder, we will have $W_1 \in \mathbb{R}^{|\mathcal{V}_{trg} \times 3H|}$ since the context d_t is then $2H$-dimensional.

Attention Model Hyperparameters		
Choice	ATTN-1	ATTN-2
Bidirectional encoder	False	True
Hidden Layer Size	500	650
Number of Layers	4	4
Word Embeddings Size	500	650
Dropout	0.2	0.35
Initial learning rate	1	1.2

Table 1: Architectural differences between ATTN-1 and ATTN-2.

4.2 Hyperparameters for attention models

We report results on two attention models (ATTN-1 and ATTN-2): the second is larger and uses a bidirectional encoder, and is intended to be our final Kaggle submission.

4.2.1 Differences between ATTN-1 and ATTN-2

Architectural differences between our ATTN-1 and ATTN-2 models are shown in Table 1. The number of epochs with which to train each model was determined by early stopping on the held-out validation set. For ATTN-1, we trained for 13 epochs, which took approximately 2 hours. For ATTN-2, we trained for 11 epochs, which took approximately 3 hours (as the model is significantly bigger).

4.2.2 Common features

For both models ATTN-1 and ATTN-2, we used stochastic gradient descent on the negative log likelihood for training, decaying the learning rate by a factor of 2 after 8 epochs (as in Bahdanau et al. (2014)). Moreover, we used "tied weights", as in Press and Wolf (2016) for the decoder network. In particular, we set the $|\mathcal{V}_{trg}| \times H$-submatrix of W_1 in (3) that multiplies the decoder output d_t to be equal to the transpose of the target embedding matrix $W_{trg}^{(E)}$. We found that doing so achieved a significant performance boost, larger than that reported in Press and Wolf (2016) for NMT models. This may be so since we are training on a relatively small dataset, so the model cannot take advantage of the additional flexibility in having these matrices' weights un-tied.

We use dropout in the hidden layers of both the encoder and decoder LSTM for our attention models, as suggested in Zaremba et al. (2014), to combat against overfitting.

4.3 Hyperparameters for all models

All of our models had the following training-related features in common: sentences were fed in batches of 32, which were arranged so that similar-length sentences were in the same batch. All model parameters were initialized uniformly in the interval $[-0.05, 0.05]$. For each gradient step, the norm of the gradient vector was computed, and if greater than 5, it was scaled down to be of norm 5.

4.4 Results

The perplexities of our models as evaluated on the held-out validation set are shown in Table 2.

4.4.1 Attention distribution

In Figure 1, we show the attention scores $score(v_s, z_t)$ for the model ATTN-2 on a selection of 3 sentences in the validation set. Lighter squares indicate higher weights. The rough diagonal across the middle confirms that, as one would generally expect, the decoder is attending to words in the source sentence in order while decoding the target sentence. Moreover, one can observe particularly strong correlations between English-German word pairs which are direct translations of each other, such as "campaign" and "Kampagne". Figure 2 shows the progress of attention over training time (epochs) for another sample sentence.

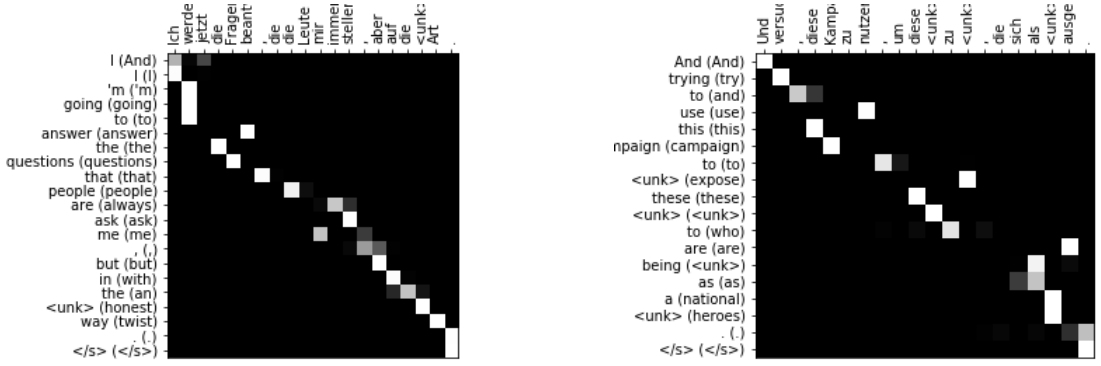

Figure 1: Sample attention distribution for two relatively long sentences. English words in parentheses show true word, whereas those not in parentheses show predicted word.

Model	PPL
SEQ2SEQ	7.049
ATTN-1	4.806
ATTN-2	4.294

Table 2: Perplexity of our models on the validation set.

4.5 Sample Translations

We implemented beam search based on word-level negative log-likelihoods, using a beam size of both 100 and 200 for our Kaggle submissions (the beam size did not end up affecting our Kaggle scores). In particular, at each iteration of the beam search, given a current list of b (b = beam size) candidate sentences, we consider the sentence-level negative log-likelihoods of all possible sentences obtained by extending each of the b sentences by a single word, and take the top b resulting sentences for the next iteration. We also artificially removed EOS-token-containing sentences from our Kaggle submission by setting the NLL of an EOS token to $-\infty$.

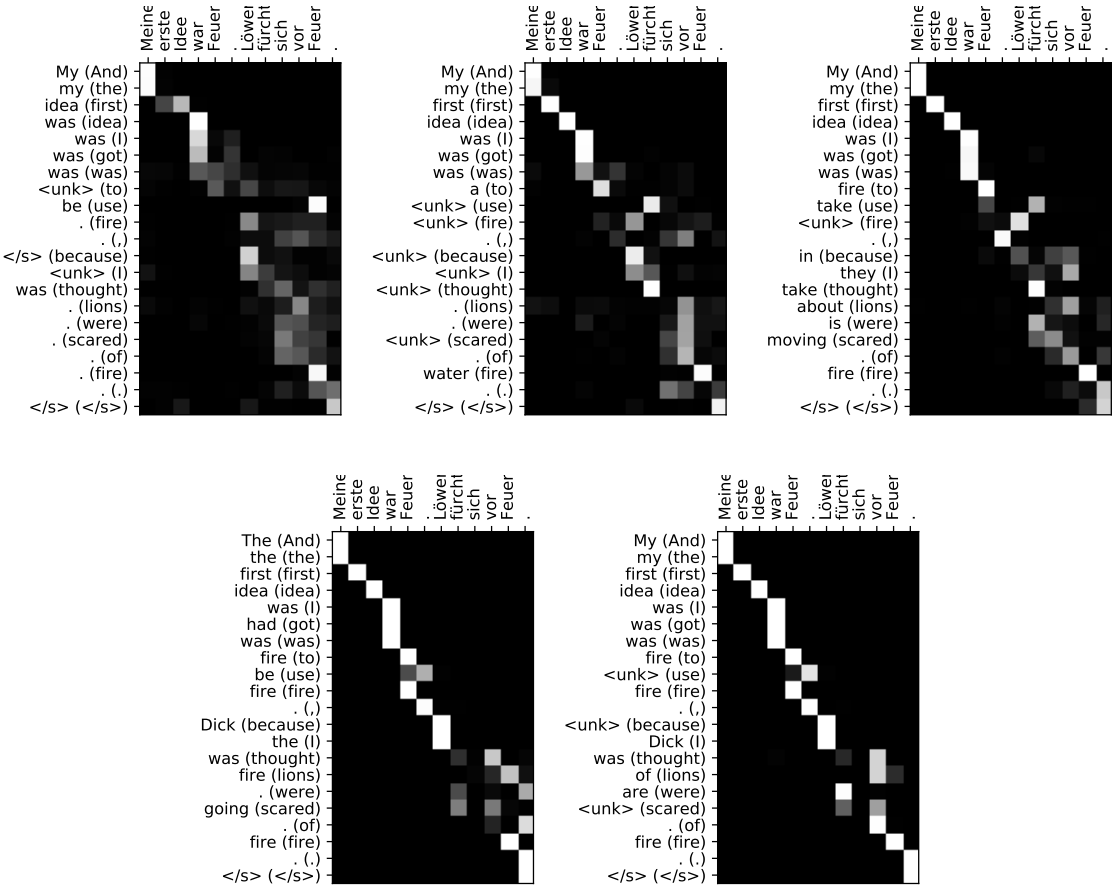

Figure 2: From left to right: attention tensor after epoch 0, 1, 2, 6, 10 for a sample sentence. English words in parantheses show true word, whereas those not in parantheses show predicted word. Note that the attention distribution becomes sharper over training time. Also note, for instance, that whenever the trained model predicts the word "fire", it attends to the German word "Feuer".

In Figure 3, we report some German sentences, their English translation as given by the validation set, and translations our model produces, as determined by beam search (in particular, the produced sentences are not fed any words from the true English translation). These sentences are produced by beam search with a beam size of 20. Moreover, whenever an iteration of the beam search produces a sentence with an EOS token, we remove that sentence and add it to a list containing candidate translations. We report the first 5 translations in this list for each of the sample sentences. Note that this strategy favors sentences of shorter length, which is generally opposite to what is standard in this area, since additional words only decrease the log likelihood of a sentence (so one generally wants to add a penalty favoring longer sentences, such as dividing log probabilities by $C(5 + \text{sent-len})^{\alpha}$, as is done in Wu et al. (2016). However, this strategy is solely for the purpose of exposition, so it was chosen for its simplicity.

German: ist ein Bauplan fr Lnder wie China und den Iran .
True English: This is a blueprint for countries like China and Iran .
Predicted English (top 5):
[unk] for countries like China and Iran .
That ’s a [unk] for countries like China and Iran .
This is a [unk] for countries like China and Iran .
That ’s a stereotype for countries like China and Iran .
It ’s a [unk] for countries like China and Iran .
German: Handy kann ein Leben verndern und einem persnliche Freiheit geben .
True English: A mobile phone can change your life , and gives you individual freedom .
Predicted English (top 5):
A cell can change a life and give a happy freedom .
A phone can change a life and give a happy freedom .
A telephone can change a life and give a happy freedom .
A cell can change a life and give a personal freedom .
A cell can change a life and give your own freedom .
German: mssen Sie ihnen die absolute Wahrheit ber das Unternehmertum verraten .
True English: And then you have to tell them the truth about entrepreneurship .
Predicted English (top 5):
Then you have to tell them the latest truth of the primate .
Then you have to tell them the latest truth about the primate .
Then you have to tell them the absolute truth about the primate .
Then you have to tell them the latest truth about the receptor .
Then you have to tell them the absolute truth of the primate .

Figure 3: Candidate translations for Germany sentences, as determined by beam search. Note that one of the top 5 sentences for the first sample uses a rough synonym of “blueprint”, namely “stereotype”. Note also that the model mis-translated the German word “Unternehmertum” (entrepreneurship) for the English word “primate.”

5 Conclusion

Upon implementing the standard Seq2seq model, and the attention model, we saw the expected boost that the attention features provided. For our dataset, our various experiments yielded some changes that boosted the validation perplexity for the attention models. At first, when we tried making the ATTN-1 model bigger and adding a bidirectional LSTM encoder, results did not improve, but by increasing dropout and maintaining tied weights for the ATTN-2 model, we were able to regularize sufficiently to further improve validation PPL.

References

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv:1409.0473 [cs, stat]*. arXiv: 1409.0473.

Luong, M.-T., Pham, H., and Manning, C. D. (2015). Effective Approaches to Attention-based Neural Machine Translation. *arXiv:1508.04025 [cs]*. arXiv: 1508.04025.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). BLEU: a method for automatic evaluation of machine traslation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia.

Press, O. and Wolf, L. (2016). Using the Output Embedding to Improve Language Models. *arXiv:1608.05859 [cs]*. arXiv: 1608.05859.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, ., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., and Dean, J. (2016). Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *arXiv:1609.08144 [cs]*. arXiv: 1609.08144.

Zaremba, W., Sutskever, I., and Vinyals, O. (2014). Recurrent Neural Network Regularization. *arXiv:1409.2329 [cs]*. arXiv: 1409.2329.