Problem Set 3: Machine Translation

Luke Melas-Kyriazi
lmelaskyriazi@college.harvard.edu

George Han
hanz@college.harvard.edu

March 4, 2018

1 Introduction

In this assignment, we explore recurrent neural network models with attention for machine translation between English and German sentences. We find that a deep LSTM network with a dot-product attention mechanism performs well with regards to BLEU, validation perplexity, and subjective translation quality. We conduct a substantial hyperparameter search to optimize our model's performance on the International Workshop on Spoken Language Translation (IWSLT) dataset. We modify and test parameters including attention type (dot product, additive, or none), dropout rate, depth (number of layers), and using a bidirectional encoder. We also find that during prediction, beam search (with a beam size of around 5-10) provides a significant performance increase over greedy search.

We counter overfitting the IWSLT dataset by employing dropout and early stopping. Although the IWSLT dataset is larger than the Penn Treebank and SST datasets used previous problem sets, it is still a relatively small dataset for machine translation, so we find regularization to be an important aspect of our model. Unlike in previous assignments, we do not compare our model with non-deep-learning based alternatives, as traditional phrased-based machine translation models are complex to implement. Indeed, the simplicity and end-to-end nature of neural sequence to sequence models is among their most attractive qualities (along with their strong performance on a variety of tasks).

Overall, our results demonstrate the effectiveness of deep LSTM networks and end-to-end training for sequence-to-sequence tasks, along with the importance of hyperparameter tuning and regularization for optimizing model performance.

2 Problem

In this problem set, we tackle a language translation task: given a sequence of previous words in German $\boldsymbol{x}_{1:n}$, we seek to predict a probability distribution over possible translations in English, $p(\boldsymbol{y}_{1:m}|\boldsymbol{x}_{1:n})$. However, since this search space is very large (in the order of $|V_{DE}|^n \times |V_{EN}^m|$) we instead opt to predict a probability distribution during train time over each word in the translation given the previous word from the target sentence. This probability distribution is a matrix with size m, the size of our translated sentence, times V_{EN}, the size of our English vocabulary, which is more tractable.

We approach this problem in a fully supervised manner, training on a dataset of paired sentences from the IWSLT dataset of TED talks in English and German. For this dataset, we chose to use the standard IWSLT train-val-test split built into the PyTorch Torchtext library. We additionally employ fastText word embeddings in both English and German, which were pre-trained by Facebook on text from Wikipedia.

2.1 Evaluation

We evaluate our model using two different language translation evaluation metrics.

Our primary evaluation metric is the $BLEU$ score, or the bilingual evaluation understudy score, developed by (Papineni et al., 2002). This metric is widely used in the field of language translation to score predictions against reference translations and has been empirically shown to correlate well with human evaluations. To compute the BLEU score, we take a geometric average of n-gram similarity scores for $n = 1, 2, 3$ and 4, as computed in the PERL script written for Moses.

Our secondary evaluation metric is *perplexity*, the exponential of the cross-entropy of our predictions. Denoting our predicted probability distribution over words as $p(x_i|x^*_{i-1})$, for a prediction $x_1, \ldots, x_N$ and a target reference $x^*_1, \ldots, x^*_N$, the perplexity PPL is given by:

$$PPL = \exp\left(-\sum_{i=1}^{N} \log p(x_i|x^*_{i-1})\right)$$

This metric roughly corresponds to the number of words from which our model would have to choose, if it chose words uniformly, to achieve similar performance on the test set.

3 Models

3.1 LSTM Seq2Seq

Our first seq2seq model, as described in (Sutskever et al., 2014), consists of a LSTM encoder and a LSTM decoder that were collectively trained end-to-end. We first project our words (as onehot vectors) through a fastText German embedding to reach a lower-dimensional embedding space before passing them in sequential order as input into our LSTM encoder. We then take the final hidden state of the encoder as our context vector, and pass it into the decoder as its initial hidden state. Throughout the training process, we use teacher-forcing and feed the previous word from our target sentence as the next input into the decoder. During validation, we instead feed the previous word from our prediction into the decoder. This previous word is chosen using either beam search with a beam width of $k = 10$ or via a simple argmax over our computed probability distribution of possible words.

We extend further on this model by making the LSTM encoder bidirectional (and correspondingly halving the number of layers). Now, instead of having one hidden state vector to use as our context vector, we effectively have two hidden states (one from the forward pass of the LSTM, and one from the backwards pass), which we add together to form the final context vector used by the decoder for translation.

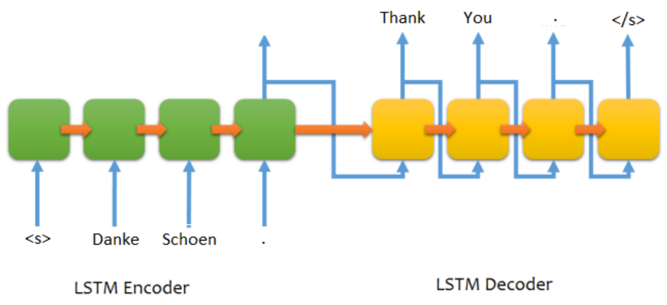

Figure 1: LSTM Seq2Seq

3.2 Bidirectional LSTM with Attention

To build the next iteration of our model, we experimented with two different attention mechanisms. First, we implemented dot-product attention as described earlier in class. In dot-product attention, the query vector is the current hidden state of the decoder, s_t, and the key vector for each word is the hidden state (at that word) of the encoder, h_i. Finally, the value vector for each word is the same as that of the key vector; the encoder hidden state, h_i. Then, we can compute the attention score function as

$$f_{att}(s_t, h_i) = s_t^T h_i$$

which is soft-maxed to provide an attention distribution over all the input words. Averaging over this distribution, the final context vector passed into the decoder is computed as

$$c_t = \sum_{i=1}^{n} h_i \cdot \frac{\exp(f_{att}(s_t, h_i))}{\sum_{i=1}^{n} \exp(f_{att}(s_t, h_i))}$$

For our second attention model, we implemented additive attention, as presented in (Bahdanau et al., 2014). Here, the attention score function is instead computed as

$$f_{att}(s_t, h_i) = v_a^T \tanh W_a s_t + U_a h_j$$

where v_a, W_a and U_a are all learned attention parameters. In our implementation, we chose to bundle s_t and h_j together by setting $W_a = U_a$.

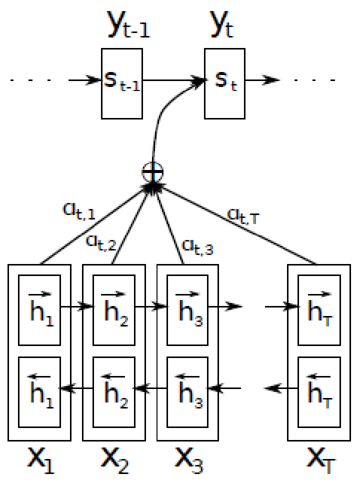
y_{t-1}
y_t
s_{t-1}
s_t
$a_{t,1}$
$a_{t,2}$
$a_{t,3}$
$a_{t,T}$
$\overrightarrow{h}_1$ $\overrightarrow{h}_2$ $\overrightarrow{h}_3$ $\overrightarrow{h}_T$
$\overleftarrow{h}_1$ $\overleftarrow{h}_2$ $\overleftarrow{h}_3$ $\overleftarrow{h}_T$
x_1 x_2 x_3 x_T

Figure 2: Additive Attention Source: Dzmitry Bahdanau

3.3 Beam Search

Instead of applying an argmax at each word individually at test time, we chose to build a beam search function that would keep track of the top k candidates and their current decoder states. We initialized the beam with a score of 0 and one candidate: the beginning of sentence token. Then, at each time step and for each candidate without an end of sentence token, our beam would run the decoder and search through the vocab space, and compute a score for each potential new candidate. We would then keep the best k new candidates from the beam, and repeat this process until all our candidates had an end of sentence token. Then from our k final candidates, we would output the best overall sentence, by score, as our final prediction.

For our beam search score function, we decided to use log likelihood, which had a slight tendency to favour shorter sentences (since shorter sentences would generally have higher log likelihood). Nevertheless, we did not notice a significant impact in sentence quality or length at test time.

4 Experiments

4.1 Training

Before describing the experiments we conducted on the models above, we should discuss our data preprocessing and training procedures. We use a pre-processed tokenized version of the ISWLT dataset along with the standard train-val split provided by the PyTorch torchtext library. With torchtext, we pre-process sentences into minibatches of sentences of approximately the same length. Sentences shorter than the longest sentence in the batch are padded with $< pad >$ tokens (zeros). When calculating attention as described above, we mask these tokens such that our model cannot attend to padding.

For all experiments below, we use batch size = 64, as this size enables us to train relatively quickly, gives stable results, and was used in a number of other papers, notably (Britz et al., 2017). Training a single model on the entire dataset using a K80 GPU takes approximately 90 minutes. We train using teacher-forcing, feeding in the ground truth target at each time step. For optimization, we use the Adam optimizer with a learning rate of $1e-3$, and decrease the learning rate by a factor of 2 after the model performace plateaus for a number of consecutive epochs.

While building the model and conducting initial experiments, the training process (especially initially loading the data) took too long for quick iterations, so we partitioned 10% of the dataset into a smaller dataset for training. For comparison, our best model trained on 10% of the training data, which has a vocabulary of only 1337 words, gives 19.50 BLEU (more details on evaluation are provided below), while our best model trained on the entire dataset gives 32.310 BLEU.

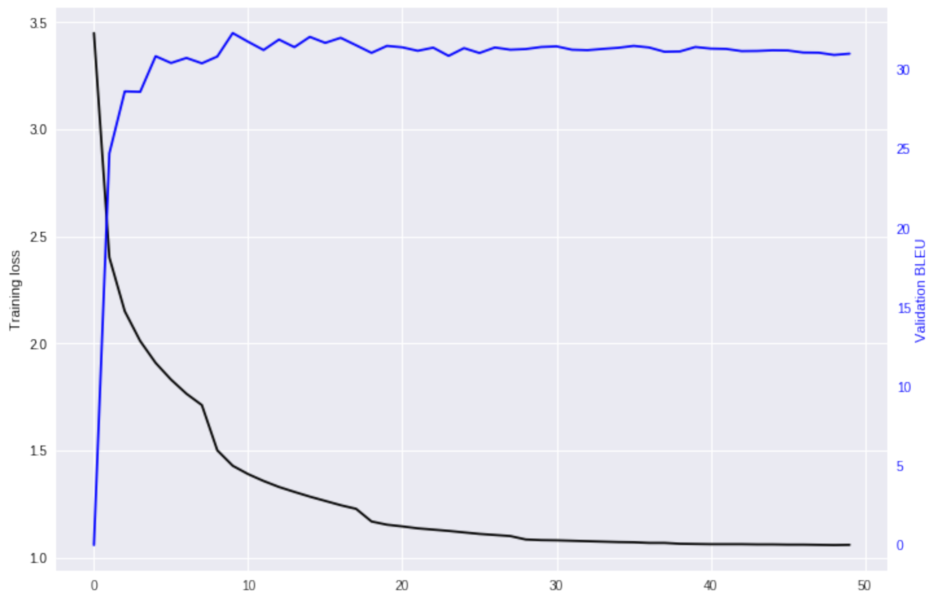

Figure 3: Training Loss vs. Validation BLEU

4.2 Validation and Prediction

We validate our model by translating each sentence in the validation set (without teacher forcing) and calculating our average BLEU score. To calculate BLEU, we use the standard Moses *multi-bleu.perl* script. BLEU score correlated well but not exactly with perplexity (using teacher forcing) on the validation set. We decided to use BLEU score rather than perplexity for model selection, as we are interested in translation quality rather than language modeling accuracy.

We validate at the end of each epoch of training using greedy search rather than beam search for purposes of computational efficiency. After tuning our hyperparameters on the validation set and selecting our best models, we use beam-search for optimal prediction performance.

For the Kaggle competition, we were asked to predict 100 trigrams for the beginning of each sentence of a test set. To do so, we used beam search with a beam size 100 and truncated the search after 3 words. We suspect that there are more optimal ways of predicting the first 3 words (in particular, predicting the entire sentence may help with prediction on the first three words), but we did not explore these other methods of prediction.

4.3 Experiments

4.3.1 Attention

We implemented and tested three types of attention: no attention, dot-product attention, and additive attention (Bahdanau et al., 2014). We found that dot-product attention performed best by a small margin, and that the non-attention model was much weaker than both attention-based models. These result surprised us, as (Britz et al., 2017) suggested dot-product attention was slightly better than dot-product attention.

Attention Type	BLEU
none	28.040
dot-product	**32.310**
additive	31.960

4.3.2 Dropout

We added dropout after the initial word embeddings, between the stacked layers of the LSTM network, and before the final linear layer. We found that adding some dropout gave a significant performance increase, but that there was no significant difference between dropout values of 0.2 to 0.5.

Dropout Probability	BLEU
0.0	28.940
0.25	**32.310**
0.5	31.310

4.3.3 Layer Depth and Bidirectionality

We found that using a bidirectional encoder greatly improved translation quality, both in terms of BLEU score and qualitatively.

Encoder Bidirectionality	
Uni/Bi	BLEU
Unidirectional	30.020
Bidirectional	**32.310**

We found that changing the number of layers of the encoder and decoder did not make such a large difference, although the deeper model . Note that the encoder of the 4-layer model is a 2-layer bidirectional model, and the encoder of the 2-layer model is a 1-layer bidirectional model.

Encoder and Decoder Depth	
Num. Layers	BLEU
2 layers	30.930
4 layers	**32.310**

4.3.4 Reversed Input

We tried reversing the input, as was done in (Sutskever et al., 2014). Although this technique made a large difference for a non-attention based model, it did not make a large difference with our attention-bsed model.

Encoder Sentence Direction	
Standard or Reversed	BLEU
Standard	32.310
Reversed	**32.640**

4.3.5 Beam Search Size

We varied the size of our beam search from 1 (greedy search) to 10. We achieved marginally better results with beam size 10, but these came with significantly longer inference times. As a result, for our final translations, we most often used beam size 5.

Beam Size	
Size	BLEU
1	32.310
5	33.400
10	**33.450**

5 Translations and Attention Visualizations

After building our model, we visualized our attention distributions. These visualizations, presented in (Bahdanau et al., 2014), give us a sense of what the network focuses on each timestep.

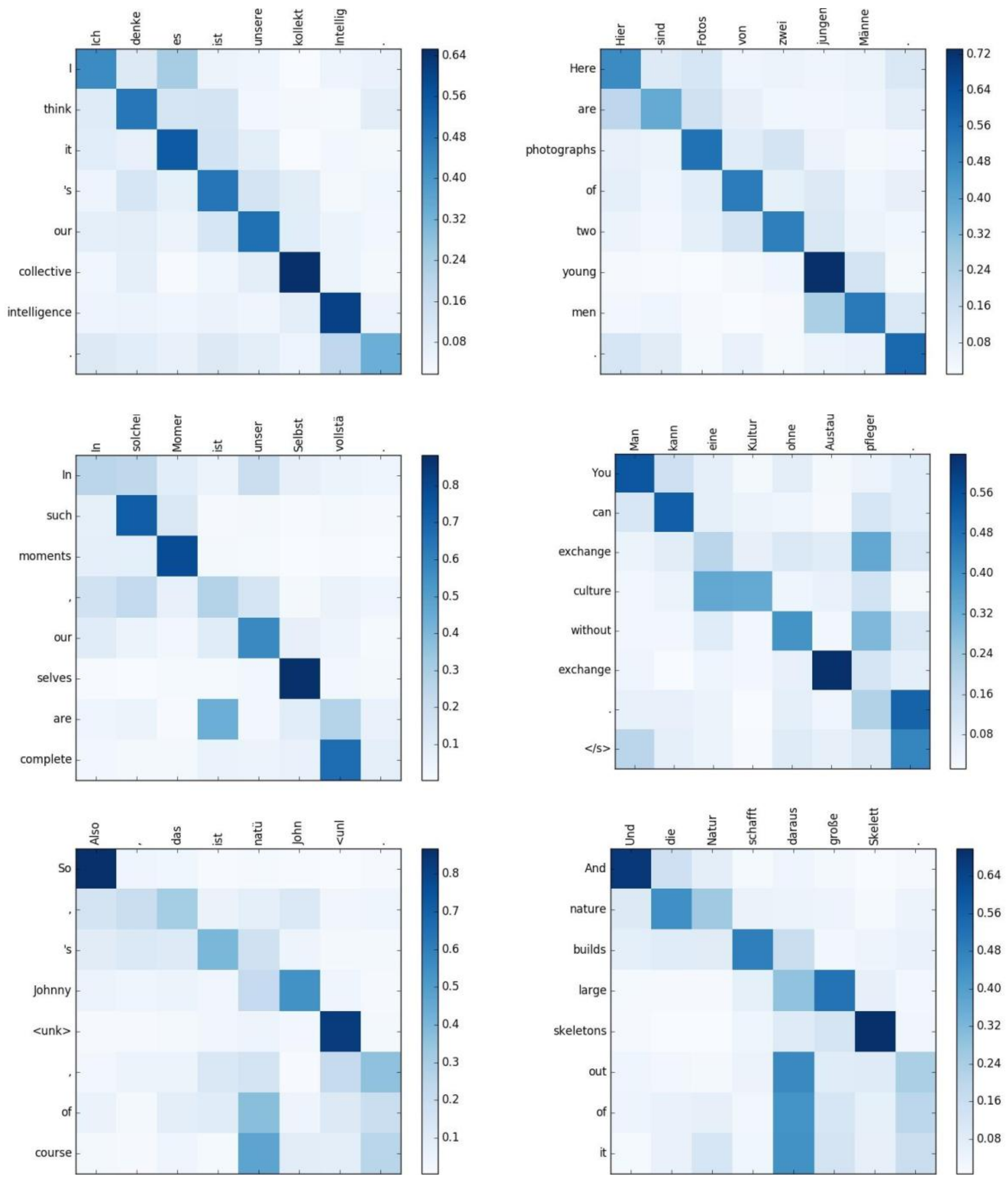

Figure 4: Attention Distributions Visualized

For fun, we also printed out a number of our translated sentences and compared them to the ground truth.

Example Sentences

Method	Translation
Source	Man kann eine Kultur ohne Austausch pflegen .
Target	You can have culture without exchange .
Training Prediction	You can exchange culture without exchange .
Greedy Search	You can exchange a culture without exchange .
Beam Search	You can exchange a culture without exchange .
Source	Dieser Tag hat unsere Sicht nachhaltig verndert .
Target	And that day really changed our perspective .
Training Prediction	This this day , changed our perspective .
Greedy Search	This day , our perspective has changed sustainable .
Beam Search	This day , our view has changed .
Source	So lernt man als Kind eine Sprache .
Target	And this is what you learn when you learn a language as a child .
Training Prediction	This that is how you learn as you learn a language . a child .
Greedy Search	This is how you learn a language as a child .
Beam Search	This is how you learn a language as a child .

6 Conclusion

In this assignment, implemented a sequence-to-sequence LSTM model with attention and applied the model to the task of machine translation on the IWSLT German-to-English dataset. Through significant hyperparameter search and experimentation, we found that a deep LSTM network with a bidirectional encoder, dot-product attention mechanism, and dropout produces strong translations as measured by BLUE score, validation perplexity, and subjective sentence quality. We found that a beam search with beam size 5 provides a significant performance increase over greedy search and implement beam search during prediction.

For future model extensions, we would like to further pre-process our data, particularly with Base Pair Encoding (BPE) (Sennrich et al., 2015). Additionally, we are interested in trying other model architectures, such as the Transformer architecture from (Vaswani et al., 2017). If deploying our model in production, we would also experiment with how to best deal with $< unk >$ tokens, either by replacing them with translated words from the source sentence or by using BPE. Finally, if given greater computing resources, we would love to train our model on a larger dataset such as the

In summary, we translated German sentences into English with an attention-based sequence-to-sequence LSTM model. We very much enjoyed this assignment and learning about machine translation!

References

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Britz, D., Goldie, A., Luong, M., and Le, Q. V. (2017). Massive exploration of neural machine translation architectures. *CoRR*, abs/1703.03906.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Sennrich, R., Haddow, B., and Birch, A. (2015). Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

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.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010.