HW3: Translation

Miro Furtado
furtado@college.harvard.edu

Simon Shen
sshen1@college.harvard.edu

March 8, 2019

1 Introduction

Neural machine translation is now the state of the art technique for computerized translation between languages. It is more effective than past phrase-based techniques and also requires considerably less engineering manpower.

For this assignment, we sought to replicate seq2seq and seq2seq with attention, two of the most important foundational models in Neural Machine Translation (NMT) using NamedTensor.

Ultimately, we found that (as expected) adding attention leads to considerably better translation quality. Both perplexity calculation and visual inspection of the translated results show higher quality results among the encoder-decoder network with attention added.

2 Problem Description

Our goal is to translate German sentences to English. We used a subset of the International Workshop on Spoken Language Translation (IWSLT) 2017 translation dataset (Cettolo et al., 2012). Specifically, we only considered sentence pairs that are at most 20 words long in English and in German. With this filtering criteria, our subset contains about 200,000 sentence pairs.

As our translation task involves language generation, our specific inference goal is the conditional probability of the next English word given the previously predicted English words and the full German sentence:

$$p(x_t \mid x_1, \ldots, x_{t-1}, \boldsymbol{y})$$

where

- $\mathcal{V}_{\text{DE}}$ is the German vocabulary of size 13353 Spacy
- $\mathcal{V}_{\text{EN}}$ is the English vocabulary of size 11560 Spacy
- $\boldsymbol{x} = [x_1, \ldots, x_t, \ldots, x_T]$ is a one-hot encoded German sentence for $x_t \in \{0,1\}^{|\mathcal{V}_{\text{DE}}|}$
- $\boldsymbol{y} = [y_1, \ldots, y_\ell, \ldots, y_L]$ is the corresponding one-hot encoded English sentence for $y_\ell \in \{0,1\}^{|\mathcal{V}_{\text{EN}}|}$

3 Model and Algorithms

3.1 Seq2Seq: Pure Encoder-Decoder Model

The first translational model that we build was a baseline sequence-to-sequence model.

With the pure-encoder-decoder model, we first encode the source sentence $\mathbf{x} = [x_1, ..., x_T]$ using an encoder RNN,

$$h_t = \text{LSTM}(\text{EMB}(x_t), h_{t-1})$$

We then take the last hidden state (or summary vector) h_T produced by the encoder RNN - which should encode the entire context of the source sentence in German.

Using this hidden state, we get the hidden state for the translated target sentence using a decoder LSTM, described by the state change equation:

$$q_i = \text{LSTM}(\text{EMB}(y_{t-1}), q_{i-1}; h_T)$$

Finally, we pass the hidden state for each word through a multilayer perceptron, which is the linear transformation

$$\mathbb{R}^{|H|} \to \mathbb{R}^{|\mathcal{V}_{\text{EN}}|}$$

and then apply a softmax so that we get a categorical distribution over the words.

3.1.1 Training and Tricks

To train our model, we maximized the word-level likelihood,

$$\max_{\theta} \sum_{i} \log p(y_i | y_{<i}, \mathbf{x})$$

We used a technique described in the literature as **teacher-forcing**. Teacher forcing allows us to use the ground truth when predicting the next word.

This has numerous advantages. It allows us to avoid having to backprop through a topk or argmax, which is not easily done/possible. It also lets our model learn more quickly.

We also flip the input sentence before it is fed into the encoder. This allows the summary vector to better represent the context most important to the beginning of the sentence we want to produce and leads to improved results.

3.1.2 Results

We experimented with different optimizers, learning rates, and batch sizes. For all experiments, we had a hidden dimension of 512 and dropout of 0.5.

Because of computational burdens, we could only train 5 epochs for each experiment. We found that for the sequence2sequence model, a large batch size with a learning rate of 0.003 was effective in getting our model to sub-20 perplexity. From there, we lowered both batch size and learning rate by an order of magnitude every 4 epochs.

This allowed us to reach an eventual validation perplexity of 14.58. Because we progressed quickly onto the attention model, we did not get a BLEU score for the pure-seq2seq model.

3.2 Adding Attention

A fundamental problem with pure sequence2sequence networks is that they create a **communication bottleneck**. The decoder is forced to predict the entirety of the translated sentence purely using the context vector produced at the end of the encoding step. This involves reducing the entire context of a sentence to a likely much lower dimensional space than the space of all possible sentence contexts.

To resolve this problem, we add attention. Namely, using the hidden vector produced during a given decoding step, we use this hidden vector to create attention weights over all of the hidden states $\mathbf{h} = (h_1, ..., h_T)$. Using these attention weights, $\vec{\alpha}$, we create a context vector by adding up all of the encoder hidden vectors.

Formally,

$$\vec{\alpha}_i = \text{softmax}(q_i^T \mathbf{h})$$

$$c_i = \sum_t \alpha_{i,t} h_t$$

Using the attention model and the same adaptive learning rate strategy outlines above, we achieved a best perplexity of around 8.70 when not including padding. To train, we used the Adam optimizer with a batch size of 32 and an initial learning rate of 0.001. We reduced our learning rate by an order of magnitude every 5 epochs.

3.3 Beam Search Algorithm

We can treat the problem of generating complete sentences as a graph search problem (e.g. Figure 1 and 2). Each node would correspond to an English word prediction, and each directed edge would represent the log conditional probability of observing the "to" node given the "from" node and all its parents and the whole German sentence. Because our goal with translation is to maximize the joint probability of the whole sentence, we would like to find the path with maximum weight through the graph. Instead of the greedy search approach used in the baseline model, we would ideally like to be able to search the full graph. However, computing the log conditional probability at each edge corresponds to a forward-compute of the Seq2Seq which is expensive.

The idea of beam search is to maintain a constant number of probable hypotheses so that we have some ability to change previously predicted English words based on the future. The number of hypotheses is called the beam width K. We now briefly describe the algorithm.

Each node in the graph should be represented by the English word that would be predicted and the hidden state of the decoder model (described above). The first node is set to the beginning-of-sentence token "$\langle$s$\rangle$". Use the encoder model described above to encode the inputted German sentence, and the final hidden state of the encoder model becomes the hidden state of this first node in beam search.

To perform the search, we iterate over the words to be predicted. During each iteration, we grow the graph by forward-computing the decoder model using the word and hidden state represented by each of the K nodes from the previous iteration. After the forward-compute, we have $K \times |\mathcal{V}_{\text{EN}}|$ new nodes from which we select the top K to keep for the next iteration. When selecting the hypotheses to keep, we consider the log joint probability so far i.e. we sum the log probabilities of the new node and each of its parents.

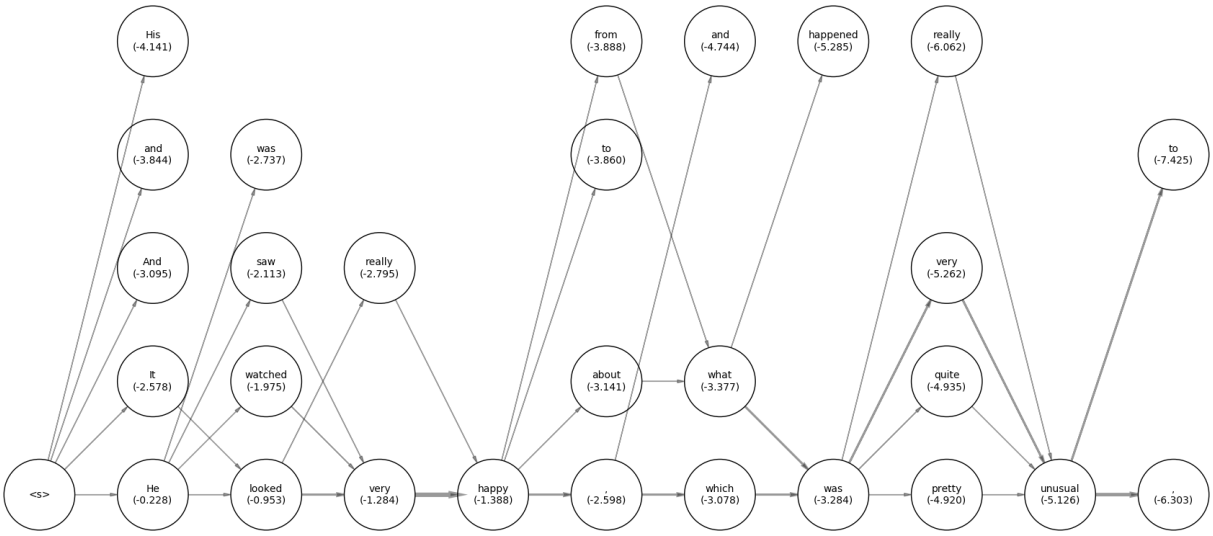

Figure 1: Sample beam search graph generated by our best model. Edge thickness corresponds to multiple hypotheses.

We terminate iterations when the maximum length (determined by the user, we used 20) is reached or if the end-of-sentence (eos) token appears in the most probable hypothesis so far.

If the eos token appears on the beam but not in the most probable hypothesis, we artificially set its log probability to negative infinity so that its corresponding hypothesis is not considered in further iterations. However, after the search is terminated, we look back at the earlier eos hypotheses in addition to the fully grown hypotheses. We selected the top 100 for submitting to Kaggle, or we selected the top 1 hypothesis for calculating BLEU. In order to avoid length bias, we choose the top hypotheses that have the largest log probabilities normalized by the length of the predicted sentence.

4 Experiments

Model	PPL	BLEU
Seq2Seq, no attn	14.51	NA
Seq2Seq, attn, beam = 2	8.71	23.89
Seq2Seq, attn, beam = 3	8.71	23.88

Table 1: Main results. Perplexities (PPL) were calculated with beam = 1.

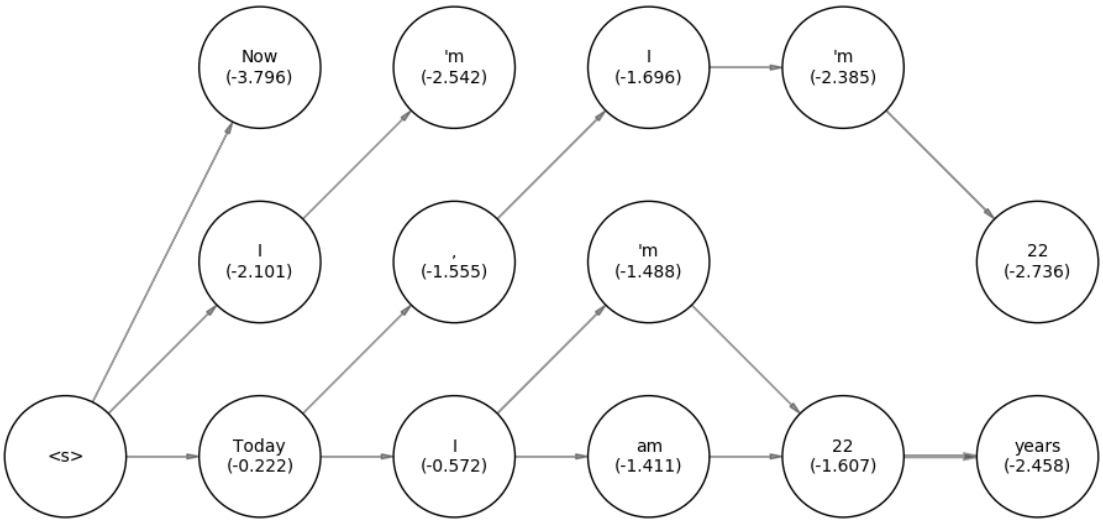

Figure 2: Another sample beam search graph generated by our best model.

4.1 Learning Rate

We found that the learning rate scheduling internally implemented in Adam was insufficient to train our model. Oftentimes the model would level out above 20 perplexity. Instead, we had to manually tune the learning rate as the model trained, decreasing it substantially as the loss decreased so that the model would continue learning.

The training process required much more active management than the language modeling task, suggesting a different loss surface geometry than in other tasks.

4.2 Attention distribution

We observed that our attention distributions do not correspond exactly to intuitive word alignments – it is off by one position (Figure 3).

This phenomena was described by Koehn and Knowles in Six Challenges for Neural Machine Translation, where they observed a similar effect in their attentional German-to-English translation model:

> All the alignment points appear to be off by one position. We are not aware of any intuitive explanation for this divergent behavior the translation quality is high for both systems.

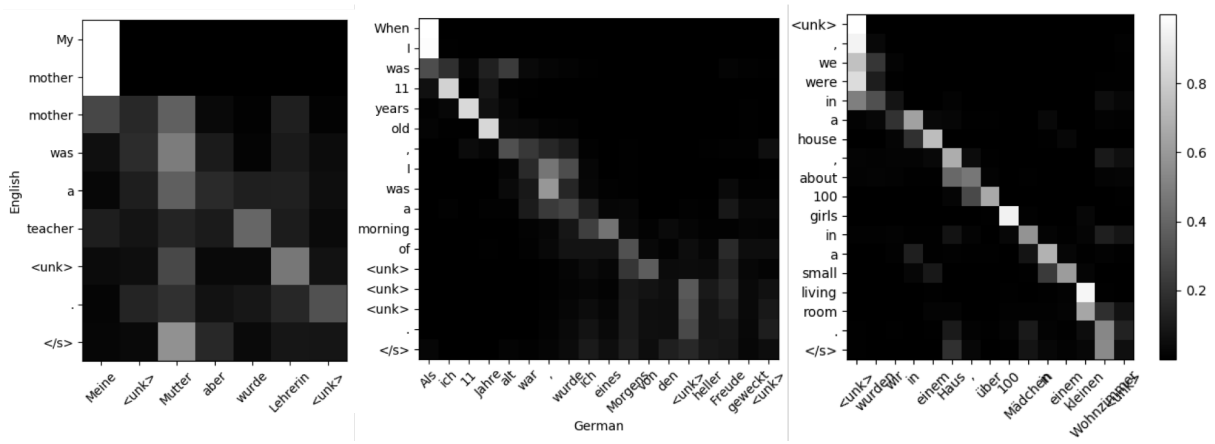

Figure 3: Attention distributions for selected sentence pairs.

This was also the case in our model, where, despite unusual alignment, the translation quality was quite good.

4.3 Tuning beam width

Greedy search (equivalent to beam width = 1) will likely overlook the best overall-scoring prediction because it only considers scores locally i.e. for each word in the sentence on its own. However, larger beam width does not necessarily guarantee better predictions. Beam search is still an approximate algorithm, so it may still overlook a globally high-scoring prediction. We found that a beam width of 3 gives the best results (Table 2).

Beam width	BLEU
1	16.81
3	18.10
5	18.02
10	17.92

Table 2: Tuning beam width using model weights from mid-training.

5 Conclusion

Ultimately, we produced a model with a satisfactory perplexity using attentional sequence to sequence. Despite unintuitive alignment issues in our attention, our model continued to perform well - although certainly far from perfect.

In the future, we think that perhaps applying a self-attention layer to the output of the decoder could allow for more effective and cohesive output sentences. Qualitatively, the model would often get individual words correct, but have difficulty creating a cohesive sentence that made complete sense.

While finding an effective learning rate for our sequence-to-sequence model was difficult, we found that rapidly decreasing the learning rate after an initial training run of 5 epochs was effective at continuing to lower perplexity after training had stalled.

6 Code Accessibility

Our code can be accessed at github.com/MiroFurtado/cs287_ps3.

References

Cettolo, M., Girardi, C., and Federico, M. (2012). WIT3 : Web Inventory of Transcribed and Translated Talks. *Proceedings of the 16th EAMT Conference*, (May):261–268.