Attentional sequence to sequence model
Assignment for 11.731

Paul Michel
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213
pmichel1@cs.cmu.edu

Abstract

We train different variations on the attentional sequence to sequence model introduced in Bahdanau et al. (2014) on the IWSLT German-English dataset.

1 Introduction

We focused our contribution on two main points :

1. Improving speed with minibatching
2. Improving the model with word embeddings

We report our results on the public test set as well as a speed comparison between one-by-one and minibatched training.

2 Architecture

2.1 General considerations

Both our encoders and decoder are standard LSTMs (`dynet.VanillaLSTM`) with one layer.

To improve generalization and prevent overfitting, we applied dropout inside of the LSTM following the method described in Gal and Ghahramani (2016). Since we lacked time to cross-validate an optimal dropout rate, we chose the arbitrary 0.2 probability.

We experimented with both unidirectional and bidirectional encoding.

Our base attentional model followed the usual architecture, with bilinear scoring, ie, if $\hat{h}_i$ is the hidden state of the source sentence at step i, h_j the hidden state of the output sentence at step j, we compute the attention score α_{ij} as follow :

$$\alpha_{ij} = \hat{h}_i^T A h_j$$

Where A is a matrix of learnable parameters with appropriate dimensions. Softmax is then applied on the vector $(\alpha_{ij})_{i=0,\dots,\text{len(source)}}$ to obtain the attention weights for step $j+1$.

As far as decoding was concerned, we implemented beam search, with a maximum beam size of 5.

2.2 Minibatching

Mini-batching is not straightforward to implement efficiently with attentional models. We retained the following approach :

- We grouped sentences randomly (ie. they were not sorted by length). We chose this version to see if the benefits of having purely iid minibatches outweighed the higher amount of padding necessary to equalize the length of sentences within one batch
- We padded the input sentence from the start and the output sentence from the end (eg. `"Ich liebe dich : I love you" -> "<s> <s> Ich liebe dich : I love you </s>"`)
- We masked the source hidden states corresponding to padding symbols so that they be ignored during decoding. Similarly, we masked the log loss corresponding to end of sentence padding symbols (otherwise it becomes easier for the model to predict `</s>`).

2.3 Adding word embeddings

In order to try and add more expressive power to our model, we experiment with adding another embedding layer in our encoder.

More precisely, instead of having $h_t = [\overrightarrow{h}_t, \overleftarrow{h}_t]$ (encoding at step t = concatenation of forward and backward encoder), we add a third encoding vector h_t^w which is simply a word embedding (different that the LSTM embedding layer).

Informally, this is a way to allow a direct flow of information between the input words and the decoder states.

Within the time constraints of this assignment we only managed to test a naive implementation of this method where we just set $h_t = [\overrightarrow{h}_t, \overleftarrow{h}_t, h_t^w]$ and perform attention and decoding as usual.

A more interesting idea though, worthy of future work, would be to separate the attention weights into two different vectors : one for the recurrent encodings and one for the word level embeddings, as well an additional way (maybe another, 2-dimensional, attention vector) to choose between the word or recurrent encoding.

This would have the advantage of allowing the model to backing up to the word-level embedding which could be leveraged for rare words or proper nouns translation.

3 Experiments

3.1 Experimental setting

We experimented with two model size.

Our baseline model is an attentional sequence to sequence model with a unidirectional encoder of embedding and hidden dimension 512 (Uni-512).

In addition to this, we train 2 other models with hidden and embedding dimensions 512, respectively with bidirectional and bidirectional + word embeddings encoding (Bi-512, Bi+Wemb-512).

We set the batch size at 32 to avoid memory overflow and use stochastic gradient descent with a learning rate of 1 and a linear learning rate decay schedule parametrized by 0.01 (ie at epoch n, $\texttt{lr} = \frac{1}{1+0.01n}$). We clip the gradient norm at 1.

We chose SGD by Occam's razor, and also to avoid convergence issues observed with more elaborate optimizers such as Adam (Kingma and Ba (2014)).

All our models were run on a Nvidia®TITAN X GPU with 12GB of available memory. Our implementation[1] used DyNet (Neubig et al. (2017)).

Once our models were trained until convergence of the validation perplexity, we ran decoding with beam-search on the disclosed test set with a beam size varying from 1 to 5 and recorded BLEU score[2].

[1]The code can be found at `https://github.com/pmichel31415/dynet-att`

[2]we used the perl script provided at `https://github.com/neubig/nmt-tips/blob/master/scripts/multi-bleu.pl` to compute the BLEU score

We then ran our highest scoring configuration on the hidden test set.

3.2 Results

We monitored the training of our model by looking at the training perplexity every 100 batches and the validation perplexity every 5000 batches.

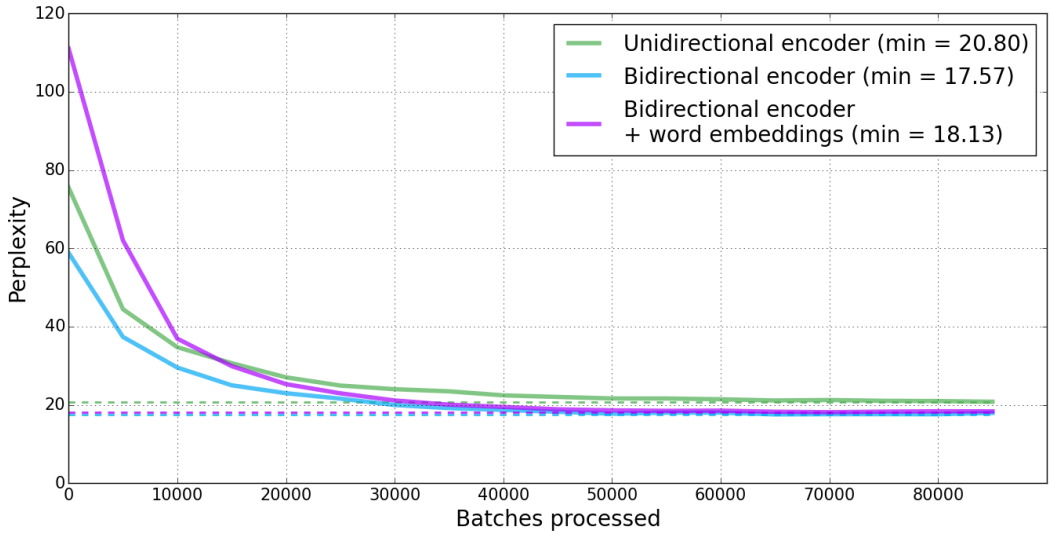

Figure 1: Evolution of the validation perplexity during training

Figure 1 reports the validation perplexity of our different models up to convergence.

As one can see, both bidirectional models perform better than the unidirectional one, but adding word embeddings doesn't seem to make the model better.

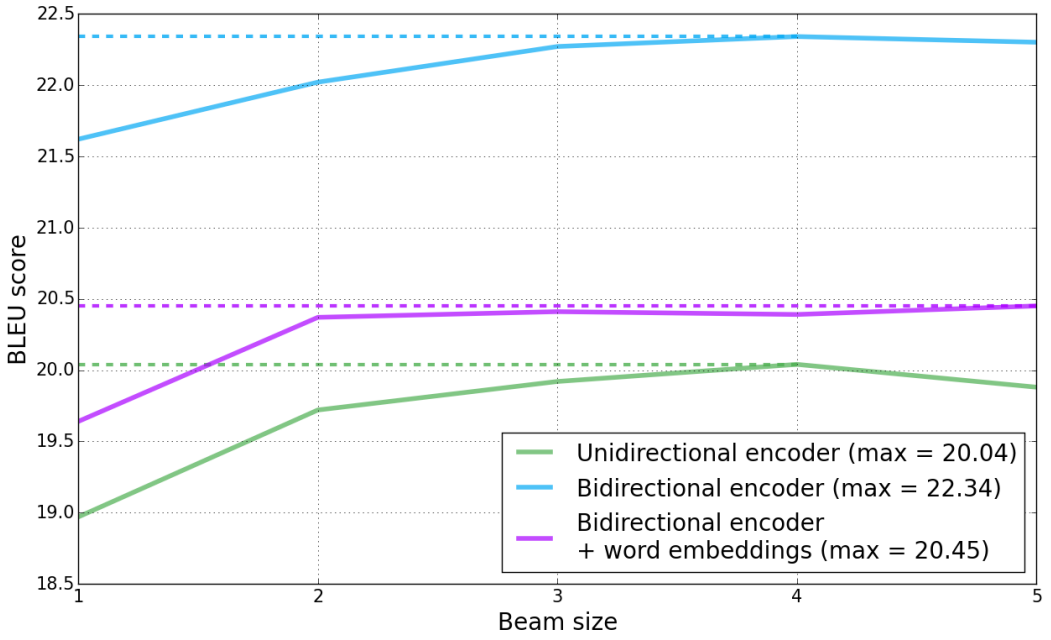

Figure 2: BLEU score given beam size

The BLEU score results are reported in figure 2. Our best BLEU score is 22.34, and again, the bidirectional model without word embeddings still performs better.

In order to get an idea of the speedup gained by using minibatching, we recorded the average token per second (tok.s^{-1}) count for both Bi-512 with and without minibatching. Without minibatching, the model processes $\approx$ 182tok.s^{-1} whereas with a minibatch size of 32 this figure goes up to $\approx$ 1000tok.s^{-1}, for a $\approx 5.5\times$ speedup.

Conclusion

Although our three models attain BLEU scores greater than 20 on the public test set, the bidirectional one seems to outperforms its counterparts. In particular, adding a direct connection between the words and the attention layer through an additional embedding matrix does not seem to increase the performance of the model.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Yarin Gal and Zoubin Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1019–1027, 2016.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, et al. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*, 2017.