Project 2

SHIVANGI MAHTO
UT AUSTIN, TEXAS
shivangi@cs.utexas.edu
Collaboration: With Nidhi Kadkol till Attention-based model,
Extension part performed individually

Abstract

In this paper, semantic parsers using sequence-to-sequence models have been presented to translate text sentences in to a formal representation based on lambda calculus. Attention-based model achieved a denotation accuracy of 62.50% with 81.30% token accuracy on the Geoquery dataset. To further improve the model's performance, we implemented beam search decoder and scheduled sampling. We could achieve a best denotation accuracy of 65.00% with 82.60% token accuracy.

1 Introduction

In this paper, we are dealing with the task of semantic parsing for a given text sequence. The purpose is to translate sentences into a formal representation such as lambda-DCS. Such representations fully disambiguate the natural language and can effectively be treated like source code: executed to compute a result in the context of an environment such as a knowledge base.

2 Basic sequence-to-sequence model

We trained an encoder-decoder model for translating a text sequence to a target sentence. Each input and target sentence were converted into vectors of indexed words. Each indexed word was transformed into a 100-dimension word-embedding for both of the input and target sentences. The word embedding layers for input and target sequences were constructed over vocabulary of input and target words respectively. For both of the embedding layers, we used a dropout of 0.2. Each of the encoder and decoder component is described briefly as below:

2.1 Encoder

To encode the input sentence into a fixed-length vector, we used an LSTM-based encoder which consists of a BiLSTM layer with 200-dimension hidden size. The encoders transforms input sentence to a fixed-length output which is taken as input in the decoder. We reversed the order of input sentence while encoding. Reversing input sequence helped us to get better token accuracy.

2.2 Decoder

Decoder consists of a single LSTM layer with 200 dimensions hidden size followed by a linear layer with output size of target vocabulary. For each training sentence, decoder takes 'sos' (token for start of the sentence marker) as the first input along with the output hidden state of the encoder and predicts the next word of the sentence, the one with maximum probability (also known as greedy decoder). For the rest of the sentence, target word of the previous time-step is taken as input along with the last output hidden state. For the last word of the target sentence, we trained decoder to predict 'eos'(token for end of the sentence).

For each predicted word, cross entropy loss was calculated w.r.t. to the target word and summed over whole sentence. We trained the model end-to-end i.e. including both embedding layers, encoder and decoder by back-propagating over cross entropy loss for each sentence using Adam optimizer with a learning rate of 1e-3. 20 epochs were found to be optimum for the encoder-decoder model. It takes around 50-60 second for each training epoch.

2.3 Inference

During decoding, input test sentence is passed through encoder to produce hidden state containing context of the sentence. Decoder takes 'sos' as the first input word along with the output hidden state of encoder to predict the first word of the sentence. The predicted word and output hidden state of the decoder is taken as the input for the next time step. We keep doing this until we get 'eos' as predicted token or the predicted output sentence exceeds maximum length. For our experiment, we chose maximum length as three times of the maximum length of the target sentence in the training data. Basic encoder-decoder model performed bad on the Geoquery dataset with a denotation accuracy of just 13.33% with 76.02% token accuracy for 10 training epochs.

3 Attention-based model

To improve the model's performance, we incorporated general attention mechanism proposed by Luong et al. (2015) in the basic decoder. Attention mechanism forces the model to learn to focus on specific parts of the input sequence when decoding, instead of relying only on the hidden state of the decoders LSTM.

We implemented a linear layer which takes hidden state of LSTM as input and produces output which was then multiplied with the encoder's output, called as score or energy for encoder's output. Softmax over scores provided attention weights. Attention weights were then multiplied with encoder's output to get the context vector. Note that our encoder is BiLSTM hence the encoder's outputs consist of both forward and backward hidden states for each time stamp. Context vector was concatenated with LSTM's hidden state to get new vector which is taken as input to a linear layer followed by Tanh activation function. Output of this layer is again passed through another linear layer with Softmax to produce final output which is the predicted probability of next word over the target vocabulary. We used all training parameters same as that for basic model. Training time for each epoch is around 70-80 seconds.

4 Extension

As a part of extension, we implemented beam search decoder, and scheduled sampling. We also tried to fine-tune the sequence-to-sequence models by training on denotation prediction, however, we could not explore it properly to get good accuracy.

4.1 Beam search decoder

Basic or Greedy decoder model uses greedy policy to predict the next word by maximizing prediction probability. However, it might be possible that even if a word has higher probability, the overall predicted sentence with that word has lower chance than others. Also, the basic decoder sometimes cannot predict 'eos' token during inference or predict incorrect words which can make bad decision for the whole sentence. To handle these situations, we implemented a beam-search-based decoder and found beam width of 3 to be optimum for the decoding purpose.

During inference at some instant t, for each of the 3 input candidate words (token stored in the beam) and their corresponding hidden state (also stored in the beam along with token), we search for the top 6 most probable predicted words by the decoder. We add their prediction probabilities to the already stored in the beam (already stored ones are for the whole sentence hypothesis till time $t-1$). Thus, we get overall 18 hypothesis of the target sentence till time 't'. We choose 3 out of them with maximum score (the summed prediction probabilities over tokens til now) and store their token along with hidden state for the next time stamp.

4.2 Scheduled Sampling

To handle noisy outputs from the decoder, we implemented schedule sampling following the method proposed by Bengio et al. (2015). During training of the decoder, instead of always choosing target tokens of last time stamp as current input, we sampled between target tokens and predicted output by the decoder in the last time step. For choosing between target and predicted tokens, we used an exponential decay probability function 0.999^i where i the count over all the training examples for all epochs.

4.3 Fine tuning with denotation learning

We also tried another topic following the work performed by Guu et al. (2017) but the results are not included here. After training attention-based model, we fine tuned the model by minimizing denotation prediction error. For this, we implemented beam search for hypothesis generation. Beam search produces possible predicted sequences by considering top 'k' predictions from the decoder for each word. We experimented only for k = 3. For each candidate hypothesis, we compared predicted denotation and the target. If the predicted denotation is matching with the target, we increase loss by the negative log probability of

Model	Greedy Dec.		BS Dec.	
	Deno.	Token	Deno.	Token
Basic Enc-dec	15.80	76.60	20.80	75.80
+ Attention	62.50	81.60	63.30	81.70
+ Scheduling	65.00	82.60	64.16	82.60

Table 1: Denotation and token accuracy (%) for seq-to-seq models. All the models are trained for 20 epochs.

Training epochs	Greedy Dec.	BS Dec.
10	48.33	55.80
20	62.50	63.30
30	61.67	63.30

Table 2: Comparison of beam search decoder vs. greedy decoder over denotation accuracy (%) in attention-based model for different training epochs.

the hypothesis calculated by beam search. However, we could not achieve good result for this model. Model tuning was difficult as the training time takes a long time.

5 Experimental results

Table 1 shows the summary of our experiments on various kind of variations in the basic encoder-decoder model. Attention-based model performs better than the baseline with a denotation accuracy of 62.50%. Schedule sampling resulted into a small improvement. Also, since attention-based model's result varies largely, so we are not sure if the contribution is purely from the schedule sampling or of the model itself.

Beam search performs similar to greedy decoder. This is not unexpected as the training data and test data is really small. Difference of even one correct denotation changes the accuracy by almost 0.75%. Also as shown in table 2, for smaller epochs, beam search performs significantly well as compared to greedy decoder. However, for larger epochs, both performs similar.

6 Conclusion

In this work, we implemented sequence-to-sequence based semantic parsers on Geoquery dataset. Attention-based model could achieve a denotation accuracy of 62.50%. Beam search based decoding helped to get better performance for the model with a denotation accuracy of 63.30%. Finally, scheduled sampling in the attention model gave best performance of 65.00% denotation accuracy with 82.60% token accuracy.

References

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*. pages 1171–1179.

Kelvin Guu, Panupong Pasupat, Evan Zheran Liu, and Percy Liang. 2017. From language to programs: Bridging reinforcement learning and maximum marginal likelihood. *arXiv preprint arXiv:1704.07926* .

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025* .