CS231n Project: Siamese Recurrent Architectures for Learning Sentence Similarity

Anyi (Casie) Bao
Casie.Bao@mdacorporation.com

1 Introduction

Learning semantic similarity between sentences is one of the essential tasks in Natural Language Processing. This can be extended and applied to many problems including knowledge deduplication, image captioning evaluation metric, etc. In this project, we consider the Siamese Recurrent Architectures proposed by Mueller and Thyagarajan in 2016 [2].

2 Architecture

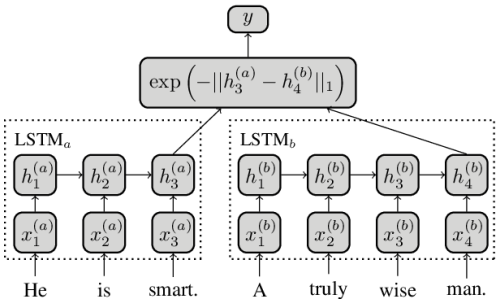

Figure 1: Manhattan LSTM Model

In this project, our goal is to predict the degree of semantic similarity between two sentences using the proposed Siamese Recurrent Architectures as shown in Figure 1. To be specific, given two sentences, our system is required to produce a relatedness score (on a continuous scale) indicating the extent to which the sentences were expressing a related meaning.

3 Experiments and results

3.1 Experimental details

We pre-process datasets using TorchText, and implement the model using Python 3.6.9 and PyTorch 1.5.1. We evaluated the proposed framework on SICK (Sentences Involving Compositional Knowledge) dataset [4] which consists of 4439/495/4906 pairs of training/validation/test sentences. Each pair is annotated with a sentence relatedness score on a 5-point rating scale, i.e., sentence_relatedness $\in [1, 5]$, with 1 indicating that the two sentences are completely unrelated, and 5 indicating that the two sentences are closely related. This score comes from the average relatedness judged by 10 different human annotators.

Data preprocessing We lower case all sentences, and tokenize sentences using spaCy tokenizer. We set a maximum-vocab-size variable called "MAX_VOCAB_SIZE" to keep the top

.

MAX_VOCAB_SIZE common tokens in vocabulary. This may not influence this dataset much as this dataset is pretty small and doesn't have many tokens. After setup, we have 1957 unique tokens in sentence a column in the sentence pair, and 1909 unique tokens in sentence b column. That includes "<UNK>" for unknown tokens, and "<PAD>" for padding tokens. and We print the most common tokens in the vocabulary and their frequencies as below,

[('a', 6220), ('is', 4079), ('the', 2351), ('man', 1271), ('in', 1144), ('and', 1018), ('are', 861), ('on', 857), ('woman', 641), ('of', 593), ('dog', 551), ('playing', 537), ('with', 500), ('two', 354), ('no', 275), ('by', 272), ('an', 255), ('there', 255), ('person', 247), ('boy', 247)].

Normally we would have removed some stop words (such as "the", "a") because they occur in abundance, and provide little information that can be used for classification or clustering. However, we do not do this step here as we have relatively small amount of unique tokens in this dataset, and some of stop words could be useful in overall sentence semantic understanding.

Neural architecture and training We here experiment two models on this architecture.

Model 1: The first model is a basic model, and it focuses mainly on Siamese Network. As stated in the original paper, we share weights between two LSTM models, i.e., $\text{LSTM}_a = \text{LSTM}_b$. Subsequently, the similarity between these representations is calculated and used as a predictor of semantic similarity. Since the similarity function $g(h^a_{T_a}, h^b_{T_b}) = \exp(-\|h^a_{T_a} - h^b_{T_b}\|_1) \in [0, 1]$, we multiply it by a factor of 5 to obtain a predicted score in a range similar to the true relatedness score. That leads to 657500 trainable parameters.

Model 2: We embed tokens in a 300-dimensional space using GloVe vectors pretrained on Wikipedia 2014 and Gigaword 5 [1] as initialization. We also reset the initialization on "<PAD>" and "<UNK>" tokens to all zeros to explicitly tell our model that, initially, they are irrelevant for determining sentiment. This reset initialization is done by manually setting their row in the embedding weights matrix to zeros. We get their row by finding the index of the tokens, which we have already done in Torchtext during preprocessing. Unlike model 1, in this model, Sentence a and Sentence b are fed into two independent LSTM encoder networks. We adopt 2-layer LSTM encoder networks and add dropout on the connections between hidden states in one layer to hidden states in the next layer. Dropout is also implemented by initializing an nn.Dropout layer with dropout rate$= 0.75$ and use it after the token embedding layer. That results in 1341400 trainable parameters.

For both models, we employ mean squared-error (MSE) as loss function, ADAM as the optimizer with learning rate$= 0.01$ and weight decay of 0.01, and choose batch size of 64. For model 2, we also tried gradient clipping to 1.0 to avoid exploding gradients in deep neural networks. It is suggested in the original paper [2], but it doesn't seem to improve our model much as shown from our experimental results. The original paper also suggests us to employ early-stopping based on a validation set containing 30% of the training examples. We experiment it in model 2 notebook.

4 Discussion and Future Direction

For each network models, we train 50 epochs. As shown in Figure 2, we observe that overall training loss continues to drop with more epochs. However, the validation loss fluctuates and doesn't decrease after 10 epochs. That means we probably encounter overfitting. Regarding test loss, we obtain 0.851 on MSE test loss for Model 1, and 1.020 for Model 2, which is reasonable compared to the leaderboard Table 7 in paper [3].

There are a few reasons to explain overfitting. LSTMs typically need large datasets to achieve good generalization due to their vast numbers of parameters. In the original paper, authors resolve this issue by augmenting their dataset with numerous additional training examples, a common practice in SemEval systems (Marelli et al. 2014) as well as high-performing neural networks. Second, we might be able to obtain a better result by fine-tuning some hyperparameters such as learning rate.

References

[1] C. D. M. Jeffrey Pennington, Richard Socher. Glove: Global vectors for word representation, 2014.

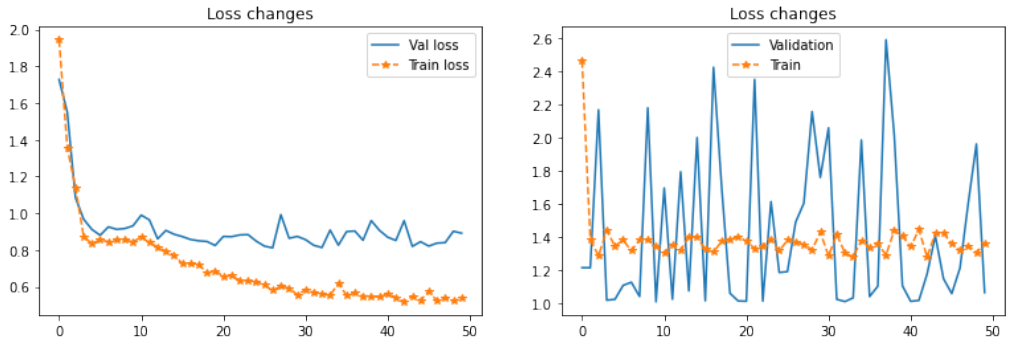

Figure 2: MSE Loss (left: model 1, right: model 2)

[2] A. T. Jonas Mueller. Siamese recurrent architectures for learning sentence similarity. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI-16)*. AAAI, 2016.

[3] M. B. R. B. S. M. R. Z. Marco Marelli, Luisa Bentivogli. Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *Annual Meeting of the Association for Computational Linguistics*. ACL, 2014.

[4] M. B. S. M. R. Z. Marco Marelli, Luisa Bentivogli. Sentences involving compositional knowledge, 2014.