DS-GA 1011 Fall 2018
RNN/CNN-based Natural Language Inference

Shizhan Gong, sg5722

October 30, 2018

The goal of this assignment is to train RNN- and CNN-based models to tackle the Stanford Natural Language Inference (SNLI) task. I first built built two baseline model based on RNN and CNN respectively. Then I did ablation study to test the effect of some hyper parameters and compare it with baseline models. Finally, I evaluate two best models on the MultNLI data set and also use the MultiNLI training data to fine-tune the models.

1 Training on SNLI

In this section, I built simple classifications using RNN/CNN based encoders. The words are mapped into pre-trained vectors. The pre-trained vectors I used is *wiki-news-300d-1M*, and then the embedded premise and the hypothesis are encoded by two encoders that share the same parameters. The last hidden states of two encoders are concatenated into one vector and than fed through a network of 2 fully-connected layers. The maximum length of the sentence is set to be 82, which is the length of the longest sentence in the training set. The size of the vocabulary is about 20k. To be more specific, I rank the words according to their frequency in the training set, and select the top 20k words. Some of these words are not included in the pre-trained vocabulary, so the actual vocabulary size is slightly smaller than 20k. We map all other words into a single unknown vector, whose weight will be trained later on. As for the fully connected layer, I set the size of the hidden layer to be 200, and the activation function of the hidden layer is relu, which are determined through some pre-experiments. For training, I used Adam algorithm with learning rate equals to 0.001.

1.1 RNN

For the baseline model of RNN, I built a single-layer, bi-directional GRU.

At the first step, I tried to change the hidden size of GRU and test its effects on the model performance, the results are shown in Table 1. The results show that the validation accuracy first increase as hidden size increases and then decrease because of overfitting. The difference is not very significant based on the given range and when hidden size is about 500, the validation accuracy is the best.

Based on the best hidden size, I further tested the influence of the ways of interacting for two hidden states given by two encoders. I tested three ways of interacting, namely concatenation, max-pooling and element wise multiplication. The results are shown

Table 1: Model performance of different hidden size of RNN

hidden size	number of parameters	train loss	val loss	val acc
100	323,203	0.0417	0.0667	0.7230
300	1,325,603	0.0376	0.0667	0.7270
500	2,808,003	0.0421	0.0679	0.7280
700	4,770,403	0.0418	0.0652	0.7260

in Table 2. The results show that element wise multiplication and max pooling will achieve very similar performance. Therefore, I randomly picked max pooling for further exploration.

Table 2: Model performance of different ways of interacting of RNN

ways of interacting	number of parameters	train loss	val loss	val acc
concatenation	2,808,003	0.0421	0.0679	0.7280
max-pooling	2,608,003	0.0232	0.0669	0.731
element wise multiplication	2,608,003	0.0421	0.0679	0.7310

Finally, I also experiment on the effect of regularization. I tried two different kind of regularization, weight decay with a weight of 1e-8 and dropout in the fully connected layer with a dropout probability of 0.5. The results are also shown in Table 3. From the results we can see that when we include dropout mechanism into the model, the performance improved significantly, while weight decay makes the accuracy lower.

Table 3: Model performance of different regularization of RNN

regularization	number of parameters	train loss	val loss	val acc
no regularization	2,608,003	0.0232	0.0669	0.731
weight decay	2,608,003	0.0744	0.0800	0.636
dropout	2,608,003	0.0206	0.0781	0.7420

1.2 CNN

For the baseline model of CNN, I built a 2-layer 1-D convolutional network wurg ReLU activations. Then I did similar ablation study.

First I tested the size of the hidden dimension of the CNN. The results are shown in Table 4. Still we can see that as hidden size increase, validation accuracy first increase and then decrease. When hidden size is 500, the model performance is the best.

After that, I tested the effects of kernel size on the model performance. And in CNN, padding size also increases as kernel size to make the output length same as input length. The results are shown in Table 5. From the results, we can see the model performance is always decreasing as kernel size increases. This may due to large kernel size fails to detect local patterns in a sentence.

Then I also tested the effects of ways of interacting and regularization on the model performance, the experiment setting is exactly the same as that of RNN. The experiment results are shown in Table 6 and Table 7 respectively. The results show that the best model is the one use concatenation to link outputs of the encoders and without any regularization.

Table 4: Model performance of different hidden size of CNN

hidden size	number of parameters	train loss	val loss	val acc
100	162,203	0.0483	0.0722	0.6890
300	662,603	0.0554	0.0688	0.6970
500	1,403,003	0.0314	0.0788	0.7100
700	2,383,403	0.0468	0.0700	0.6990

Table 5: Model performance of different kernel size of CNN

kernel size	number of parameters	train loss	val loss	val acc
5	1,403,003	0.0314	0.0788	0.7100
9	3,003,003	0.0534	0.0756	0.6750
11	4,603,003	0.0567	0.0787	0.6480
15	6,203,003	0.0659	0.0826	0.6110

1.3 results

In the end, using RNN, we built a model that can achieve an accuracy of 0.742 in the validation set. while with CNN, we can obtain a model with accuracy of 0.710 in the validation set. All the loss curves and accuracy curves are plotted on the jupyter notebook.

Here, we highlight three correct cases and three incorrect cases according to the best model, the one using RNN. In all three cases, the real label are neutral while the predicted label are entailment. The premise and the hypothesis are essentially talking about the same thing, but the hypothesis add some additional information that haven't shown in the premise, such as the gender of the person or the address of the booth. However, since most information of premise and hypothesis is overlapping, the encoder tend to map them into a feature space close to each other, so the model makes mistakes in the end. Therefore, in these three cases, the mistakes are reasonable. Sometimes the same mistake may be made even by a human.

2 Evaluating on MultiNLI

Then we evaluate our best model, one each for RNNs and CNN, using MultiNLI data. The results are shown in Table 9. From the results we can see that although our model can achieve accuracy larger than 0.7 in SNLI data, their accuracy on MultiNLI data are very low. And the accuracy differs among different genres. This shows when we train a neural network, the results highly rely on the distribution of the training data. When the distribution of the validation set is similar to that of training set, the model will have high predictive power and can be generalized to validation set. However, in this case, different genre has different distribution. Therefore, the model does not apply to new data set.

Meanwhile, the CNN and RNN perform better on different genre. CNN performs the best on telephone while RNN performs the best on fiction. This shows different model tend to catch different pattern in the training data.

Table 6: Model performance of different ways of interacting of CNN

ways of interacting	number of parameters	train loss	val loss	val acc
concatenation	1,403,003	0.0314	0.0788	0.7100
max-pooling	1,303,003	0.0457	0.0810	0.6650
element wise multiplication	1,303,003	0.0354	0.0803	0.6800

Table 7: Model performance of different regularization of CNN

kernel size	number of parameters	train loss	val loss	val acc
no regularization	1,403,003	0.0314	0.0788	0.7100
weight decay	1,403,003	0.0641	0.0741	0.6640
dropout	1,403,003	0.0470	0.0739	0.6820

Table 8: Highlighted cases

Premise	Hypothesis	True label	Predicted label
A person is climbing a cliff wall that overlooks water.	A man is climbing.	neutral	entailment
An old dusty car is half way in the brown water.	Someone wrecked their car a long time ago.	neutral	entailment
This is a man in a photo booth.	A man in a photo booth at a carnival.	neutral	entailment
Two men are listening to music through headphones.	Two men listen to music.	entailment	entailment
Two women, one walking her dog the other pushing a stroller.	There is a snowstorm.	contradiction	contradiction
A group of numbered participants walk down the street together.	Participants wait for the beginning of the walkathon.	neutral	neutral

Table 9: Model performance on MultiNLI task

RNN		CNN	
genre	acc	genre	acc
fiction	0.4673	fiction	0.4372
government	0.4281	government	0.4154
slate	0.4261	slate	0.4232
telephone	0.4428	telephone	0.4657
travel	0.4043	travel	0.4389

3 Fine-tuning on MultiNLI

Since the test result show the model trained using SNLI data performs poorly on the MultiNLI, we use MultiNLI train data to fine-tune the model. We set the learning rate to be only 0.0001, and fine tune each model for 10 epochs. After fine tuning using training data of a certain genre, we evaluate the model performance use all other genres. The resulting validation accuracy is given in Table 10 for RNN and Table 11 for CNN.

Table 10: RNN performance after fine tuning

training genre validation acc	pre-train	fiction	telephone	slate	government	travel
fiction	0.4673	0.4884	0.5075	0.4864	0.4915	0.4975
telephone	0.4281	0.4507	0.4886	0.4607	0.4776	0.4796
slate	0.4261	0.4561	0.4561	0.4780	0.4691	0.4451
government	0.4428	0.4528	0.4783	0.4852	0.4980	0.5148
travel	0.4043	0.4532	0.4735	0.4644	0.4919	0.5122

Table 11: CNN performance after fine tuning

training genre validation acc	pre-train	fiction	telephone	slate	government	travel
fiction	0.4372	0.4915	0.4995	0.4824	0.4603	0.4683
telephone	0.4154	0.5025	0.5114	0.4905	0.4965	0.4866
slate	0.4232	0.4551	0.4501	0.4371	0.4531	0.4651
government	0.4657	0.4656	0.4547	0.4754	0.5039	0.4931
travel	0.4389	0.4898	0.4878	0.4868	0.5112	0.5305

From the results, we can see after fine tuning, the accuracy of all genres have improved. Besides, when we fine tune using training data of one genre, the validation accuracy of other genres can also be improved. However, even after fine tuning, the validation is still far from satisfying. It is only about 0.5, much lower than that of SNLI data, which is 0.7. This may due to two reasons, first, the hyper parameter is tuned for SNLI task, so the setting is not suitable for MuliNLI. And second, the size of MultiNLI training data is very small, which is not enough to train a good model.

A Appendix: github repo

https://github.com/670973787/RNN-CNN-based-natural-language-inference