CSI 5386 Natural Language Processing
Assignment 2
Wei Li 0300113733(implemented LSTM and CNN) libo long 300151908(implemented BERT)
report date: 3/9/2020

1 Question

Assignment 2 requires to train machine learning models for the Text Entailment Task and the Semantic Relatedness Task on the SICK data.

SICK data

SICK dataset consists of 10,000 pairs of sentences annotated for semantic relatedness and entailment. In the experiment, 4500 data points are used as train set, 500 data points are used as validation set and 5000 data points are used as test set.

Task 1: Text entailment

Text entailment is a 3-classes-classification problem. Given 2 sentences, their relationship are either ENTAILMENT, CONTRADICTION, or NEUTRAL, which is encoded as 0, 1 and 2 in the experiment. We train classifiers on the training set, and use the classifiers to given prediction on the test set. Based on the prediction and true data label, classification accuracy as well as the Precision, Recall, and F-measure for each classes is calculated. The baseline achieve accuracy of 56.15%.

Task 2: Semantic relatedness

Text entailment is a supervised regression problem, given 2 sentences, a 1-5 continuous scale score of relatedness is calculated by the regression model. The main evaluation measure is Pearson correlation between outputs and the labels, additional evaluation MSE and Spearman correlation is also calculated. The baseline achieve pearson correlation of 0.62.

2 Methods

We approach both of the tasks by three model architectures, they are 3-layer-Bi-LSTM, CNN and BERT.

3-layer-Bi-LSTM

The 3-layer-Bi-LSTM (Figure 1) is basically the same structure as the Theano starter code, but implemented in Pytorch. We use 300 dimensional pre-trained GloVe word embedding to embedding tokenized words in both Premise sentences and Hypothesis sentences, which is then passed to 3 layers of stacked bi-LSTM respectively. We set the size of hidden state to 64. The hidden state of the top layer is then passed to a mean pool, which produces an vector encoding of both premise and hypothesis, i.e. $\mathcal{P}$ and $\mathcal{H}$. We calculate the absolute difference of $\mathcal{P}$ and $\mathcal{H}$ and their element-wise product, and then concatenate the 4 vector to form a single vector. The vector is then pass to a single layer full connect layer without activation function. For the task 1 (classification), the last layer produce 3×1 vector which is then passed to Softmax layer. We

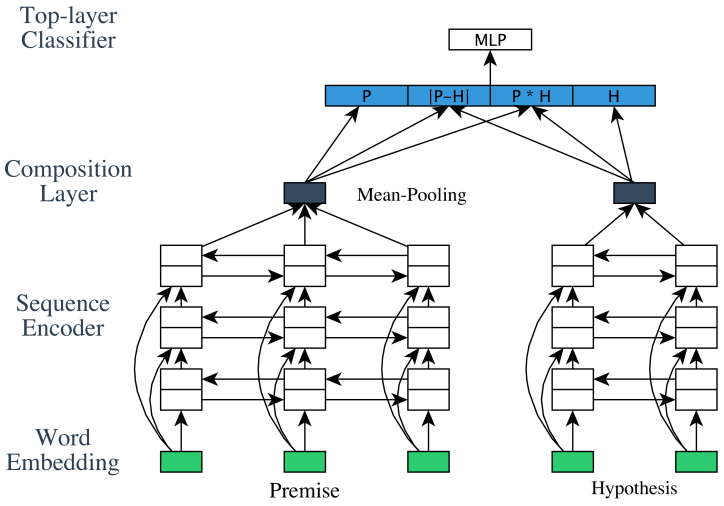

Figure 1: Bi-LSTM structure

use cross entropy loss for Task1. For the task 2 (regression), last layer produce a scalar, and we use MSE loss.

CNN

Like Bi-LSTM, we embed each sentence into a matrix using GloVe pre-trained word embedding. We padded all the sentences into max_length 30. In the CNN structure (Figure 2), for convolution layers, we use filters of kernal size 2, 3 and 4, and then ReLU function and max pooling layers. The output of max pooling layer is concatenated to form a encoding of sentence. We train encoder for premise and hypothesis independently. The premise and hypothesis vectors are then concatenated into a single vector, being randomly dropout and then passed to a full connection layer for classification or regression. We use the same full connection layer structure and loss function as Bi-LSTM.

BERT

Bidirectional Encoder Representation from Transformers (BERT)[1] is a bidirectional Encoder for pre-training language representation. In the paper, Google illustrated two structure Masked LM and Next Sentence Prediction to capture high-level features of word and sentence.

The pre-training model embedding is the sum of three embeddings.

- Token Embeddings is a words vector, first word is always CLS
- Segment Embeddings is used to distinguish the difference of sentence.
- Position Embeddings is the feature about position of the words. Word can have different meaning in different position of a sentence

When training the pre-training model, the model randomly masks 15% tokens. The loss function only calculates the masked token loss.

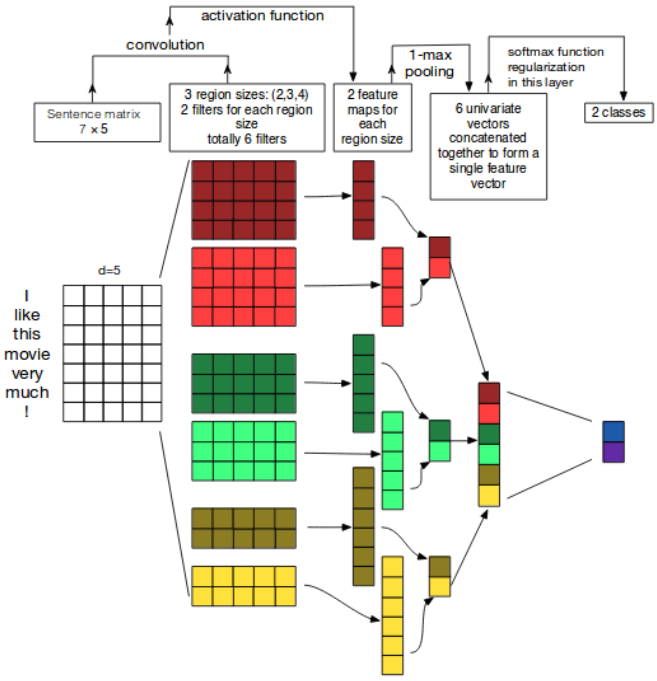

Figure 2: CNN structuree (Zhang and Wallace, 2015)

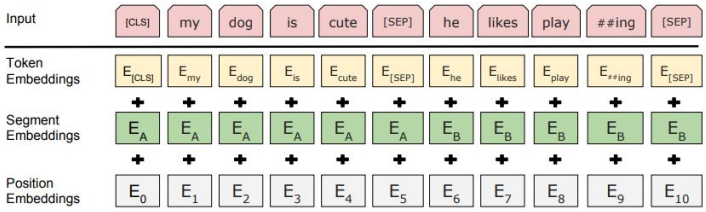

Figure 3: BERT pre-training embedding

3 Experiment Setup

3-layer-Bi-LSTM and CNN

We use similar training scheme for Bi-LSTM and CNN. During training, we do not use cross-validation, only sampling batches from test set to monitor the training accuracy and test accuracy(or loss). The learning rate is 0.001, batch size is 250 and the model is trained for 250 epochs. We also use adjusting learning rate, which divide learning rate by 10 after the 128th epoch and the 180th epoch.

For accuracy evaluation, we evaluate the whole test set in each training step in the last 5 epochs, and average the accuracy, Precision, Recall, and F-measure for each classes (in total, the denominator is 90). We also calculate pearson correlation, MSE and Spearman correlation is the same manner.

BERT

Task 1 is a classification problem. We use BERT to embed all sentences and put the embedded sentences in a bi-directional LSTM model, adding a softmax layer at the end. The output value is a list of probability of labels [ENTAILMENT, CONTRADICTION, NEUTRAL]. We use cross entropy as loss function.

Task 2 is a regression problem. We build the model based on the structure like task 1, but remove the softmax layer. The output is a number between 1-5. We use mean square error as loss function.

The learning rate is 0.00002, batch size is 32 and the model is trained for 3 epochs.

4 Result

4.1 Task 1

	accuracy	class	ENTAILMENT	CONTRADICTION	NEUTRAL
bi-LSTM	0.5712	precision	0.4769	0.4697	0.5886
		recall	0.3837	0.0799	0.7890
		f score	0.4186	0.1325	0.6728
CNN	0.5748	precision	0.4644	0.5522	0.6160
		recall	0.4354	0.4389	0.6675
		f score	0.4460	0.4792	0.6384
BERT	0.8646	precision	0.8899	0.7970	0.8960
		recall	0.8528	0.8550	0.8725
		f score	0.8709	0.8250	0.8841

Table 1: Task 1 Results

BERT confusion matrix: $\begin{bmatrix} 1209 & 198 & 7 \\ 287 & 2437 & 69 \\ 21 & 85 & 614 \end{bmatrix}$

	MSE	Spearman	Pearson
bi-LSTM	0.9045	0.0016	0.0023
CNN	0.9017	0.0482	0.0473
BERT	0.2561	0.8301	0.8854

Table 2: Task 2 Results

5 Instruction to Run the Program

5.1 Bi-LSTM and CNN

To train bi-LSTM and CNN, put data set and **util.py**, **Models.py**, **training.py** on the same directory. You need to have GloVe word embedding file in the same directory. Run the **training.py** file can begin the training and evaluation automatically. To see training options, call **$python training.py –help**.

5.2 BERT

requirement:

- Linux
- python >= 3.6.0
- tensorflow = 1.12
- numpy
- scikit-learn

training:

- download and unzip Bert pre-training package from https://github.com/google-research/bert. (BERT-Base, Uncased: 12-layer, 768-hidden, 12-heads, 110M parameters)
- put the package in the experiment root.
- execute: bash classification.sh in experiment root for training the model
- execute: bash classification_test.sh in experiment root for validation.

6 Discussion

According to the experiment, the best classifier is BERT pre-trained model. The key improvement to the accuracy is from the sentence embedding (feature extractor). Fig 3 shows that the pre-trained embedding is a combination of sentence embedding, word embedding and index embedding, which extracts more high-level feature than GloVe.

References

1. Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Well-read students learn better: On the importance of pre-training compact models. *arXiv preprint arXiv:1908.08962v2*, 2019.