Question Classifier

COMP61332 Text Mining Coursework 1

Xinyi Ouyang, Chuhan Qiu, Zhangli Wang, Mingchen Wan, Mochuan Zhan

Abstract

Question classification has been a popular NLP research field since it was regarded as a significant part of question answering. In this project, we built the question classifier by BoW (bag-of-words) and BiLSTM combined with a feed-forward neural network, where BoW or BiLSTM is used to get the sentence vector and the feed-forward neural network is used to do the classification task. The aim is to find how the different embedding settings will influence the two question classifiers.

1 Introduction

Question answering has been a more significant research field since users struggle to handle the abundant information online(Hirschman and Gaizauskas, 2001). In order to answer a question correctly, it is important to classify the question to understand what the question asks for (Zhang and Lee, 2003). According to Zhang and Lee (2003), question classification task can put constraints to the plausible answers of the question to reduce the search space, which would help the question answering system perform better. Therefore, almost all question answering systems have a question classification part.

In this project, the basic problem is to develop a question classifier and explore how the different settings influence the classification results. Sentence representation is a core task in question classification because classification relies on understanding the sentences (Liu et al., 2020), which is based on word embedding. This research concentrates on implementing the question classifier using BoW (bag-of-words) and BiLSTM (Bi-directional Long Short-Term Memory) combined with a feed-forward neural network, and aims to compare the results on different word embedding methods and settings of the two models.

2 Methodology

To implement the classifier, we should convert sentences into machine-readable vectors. With the randomly initialised word embedding or the pre-trained word embedding, BoW and BiLSTM can get the sentence representation based on those word vectors.

2.1 Bag-of-words

Bag-of-words is a popular model which is widely used in Natural Language Processing and Information Retrieval (Zhang et al., 2010). BoW model will ignore the order of the words and assess the sentences as a bag of words. In this case, the sentence representation from the BoW model will be the sum or average of the words vectors in the sentence. The vector calculation method for sentences is as follows:

$$S = w_1 + w_2 + w_3 + ... + w_i$$

or

$$S = \frac{w_1 + w_2 + w_3 + ... + w_i}{i}$$

where S is the sentence vector, w_i is the last word vector in the sentence.

In our experiments, we calculate the mean value of the word vectors in the sentence to get the sentence vector. And after getting the sentence representation, there is a feed-forward neural network to implement the classification task, where the output of BoW is the input of the NN. Therefore, in this project, the whole BoW model combines the BoW part and a feed-forward NN together to build a classifier.

2.2 Bi-directional Long Short-Term Memory

In general, the order of words will also have a huge influence on the meaning of sentences, and BiLSTM model can make the context of the word into consider. BiLSTM is a combination of a forward LSTM and a backward LSTM. LSTM network is a recurrent neural network that some outputs can be a part of the next inputs (PyTorch, 2021). With this feature, for each element in the input sequence, there is a hidden state h_t that can contain the information from the previous elements in the sequence. In this case, the sentence can be represented as the final hidden state of the sequence.

To decide preserving or removing the existing information, LSTM has a forget gate which can judge the information according to weights that assigned to the information during the training process. Moreover, LSTM also have a input gate i and a output gate o, which determine the information need to be updated and the output value of LSTM unit respectively. Their calculation methods are as follows:

$$\begin{cases} i_t & = \sigma(W_i x_t + U_i h_{t-1} + V_i c_{t-1} + b_i) \\ c\_in_t & = \tanh(W_c x_t + U_c h_{t-1} + V_c c_{t-1} + b_c) \\ c_t & = f_t \cdot c_{t-1} + i_t \cdot c_i n_t \end{cases}$$

and

$$\begin{cases} o_t = \sigma(W_o x_t + U_o h_{t-1} + V_o c_{t-1} + b_o) \\ h_t = o_t \cdot \tanh(c_t) \end{cases}$$

where $tanh(.)$ is tanh activation function, W, U and V are connecting weights, f and i are the values to decide update or not.

In our experiments, we jointed a part of the output of the forward and the backward LSTM to get the correct-dimension sentence vector. Some hidden states of forward and backward LSTM are supposed to contain the information of context, so jointing them together can be the representation of the whole sentence. Additionally, the BiLSTM model in this project combines the BiLSTM and a feed-forward neural network together. With the sentence vectors that are generated from BiLSTM, the subsequent NN can implement the classification task.

3 Experiments

3.1 Experiment set-up

In this paper we utilize the training set and test set from the "experimental data for question classification" (Li and Roth, 2002) in our experiments, which contain 5500 and 500 labeled questions respectively. Because there is no development set provided, we randomly split the training set into 10 portions. 9 portions of which are for training and the left is for development. As for the pre-trained weight required by the word embedding, we utilize a small dataset originating from Glove which contains 9549-word vectors (Pennington et al., 2014).

The steps for pre-processing all datasets are as follows: a) lowercase all letters, separate labels and questions b) remove stopwords from the question texts and create files for storing separated labels and questions c) create a vocabulary for all words/symbols that appear in the question texts. d) when reading pre-trained data (token, vector) from the Glove, replace those whose token is not included in the vocabulary with token #unk# and its vector. Similarly, the words in the test set will be replaced by #unk# if they are not in the vocabulary.

To determining the stopwords list, we reference the stopwords list from the NLTK (Loper and Bird, 2002) as our base list. However, the interrogatives in the list should not be regarded as the stopwords in this project since our task is to classify the questions. So, we update the base list by deleting the interrogative words in the list(e.g., what, where, when, how...).

As for performance evaluation, we take advantage of several scikit-learn metric functions (Kramer, 2016) to evaluate the model's prediction, which are known as Accuracy and Weight-F1-score.

3.2 Results & Evaluation

The hyperparameter tuning takes the learning rate of the ADAM optimizer and training batch size into account. According to studies, a larger batch size could speed up the computation by utilizing parallelism but may lead to poor generalization. For this project, the batch size for training is specified as 60 for all models that guarantee both training efficiency and performance. Learning rates of the ADAM optimizer are tuned and specified as the same for the same type of model for analyzing the impact of different configurations to the BoW model or the BiLSTM model.

3.2.1 BOW Model

For the BOW model, the learning rate of the ADAM optimizer is tuned as 0.001 and the loss curves during training of three different configurations are plotted in Figure 1. It can be identified that with the uniform learning rate, the model with random initial weights converges the earliest while the model with frozen pre-trained weights meets the greatest resistance during training. The early convergence of model with random weights demon-

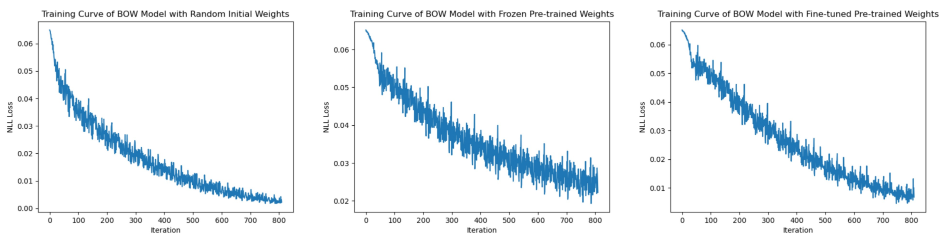

Figure 1: NLL Loss of BOW Models During Training

Model	Model Weights		Accuracy	Weighted F1 Score
BoW	Pre-trained	Frozen	0.582	0.501
		Fine-tuned	0.704	0.669
	Random	Fine-tuned	0.714	0.689
BiLSTM	Pre-trained	Frozen	0.718	0.682
		Fine-tuned	0.730	0.710
	Random	Fine-tuned	0.700	0.682

Table 1: Evaluations of the BoW Model and BiLSTM Model with the Test Set

strates that the BOW model is capable to be trained to tackle with this classification task. The reason that the model with fine-tuned pre-trained weights converges relatively slower might be that improving pre-trained weights could be more challenging than improving random weights. The reason why the model with frozen weights meets the highest resistance might be that training the whole neural network with the embedding layer frozen is more challenging, in other words, updating weights of the embedding layer could assist convergence.

The quantitative evaluation of the BOW model is calculated on the test set shown in Table 1, including Accuracy and Weighted F1 Score. We can identify that the performance of the model with frozen pre-trained weights is the worst while the model with random weights performs the best, which conforms to the former analysis. This result is contrast to our assumption that the model with fine-tuned pre-trained should perform better than the model with random initial weights, one possible explanation is that as the pre-trained weights are more difficult to be updated, the convergence of this model may take longer epochs.

A confusion matrix is generated with the BOW model of random initial weights and the test set. Primarily, it can be identified that the distribution of classes in the test set is imbalanced, and the majority of the model are predicted correctly in general. For the classes with an abundant number of instances such as $desc : def$ and $hum : ind$, this model has F1 scores of more than 0.85. But there exist challenges for the model to predict the classes with insufficient instances. For example, the model predicts more than half of $entry : color$ classes to other classes. This problem may be the account that in the training set this kind of imbalanced distribution of classes may exist as well that the model can be trained well for predicting the majority classes but not the minority classes.

3.2.2 BiLSTM Model

The experimental application parameters of Bilstm model are (embedding dim = 300,batch size = 60,hidden dim bilstm = 150,n input = 300,n hidden = 128,n output = 50,learning rate = 0.0004).

The results of the three experiments are shown in the Table 1 according to whether Pre-train or freeze. When the Randomly initialised word embedding was applied to the test set Trec 10, the accuracy was 0.718 and the F1 score was 0.682. When the Pre-trained Word embedding method is used (glove.txt is used as the Pre-trained data set) and the weight is not frozen, the accuracy is 0.730 when the training model is applied to test set Trec 10. The F1 score is 0.710. When the Pre-trained Word embedding method is used (glove.txt is used as the Pre-trained dataset) and the weight freeze is applied, the accuracy of the training model applied to Trec 10 is 0.718. The F1 score is 0.682.

According to the experimental results, the accuracy and the F1 score were higher with the Pretrain data set than with random initialization. This

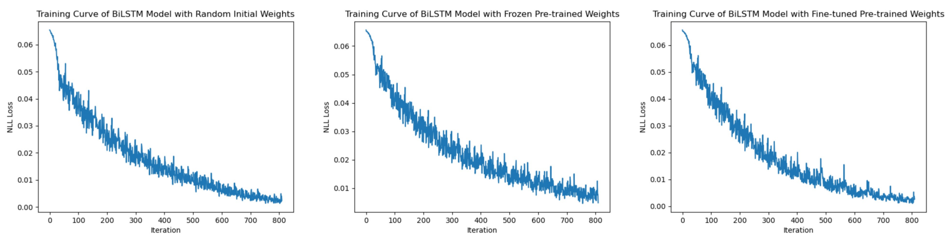

Figure 2: NLL Loss of BiLSTM Models During Training

phenomenon is intuitively clear to explain: For relatively small data sets such as the training set in this experiment, the machine can theoretically only learn the meaning of words from the data set, but when confronted with the test set data, the words in the test set may have more meanings and uses. However, if Pre-trained Word embedding is used, the machine learns more word meanings from a larger and broader data set, thus achieving higher accuracy in the face of test sets.

For bilSTM model, the learning rate is adjusted to 0.0004 using ADAM optimizer, and the loss curves of training with three different configurations are shown in FIG. 2. It can be found that under the condition of the same learning rate, the model with random initial weight converges the fastest, while the model with frozen pre-training weight encounters the greatest resistance in the training process. The early convergence of the stochastic weight model indicates that bilSTM can be trained to handle this classification task. The reason why the convergence speed of the model with fine-tuned pre-training weights is relatively slow is that word weights need to be updated during bilSTM and classification tasks, resulting in more computation. The reason why the weight frozen model has the greatest resistance may be that the frozen embedding layer is more challenging for the training of the whole neural network. In other words, updating the weight of the embedding layer is conducive to convergence.

As for whether the freeze is used for fine-tuning of Pre-trained word embedding, according to the test results, higher accuracy can be achieved when the freeze is not used. Using freeze means that the word vector will not be updated during subsequent network training. This indicates that the meaning of each word will be determined after word embedding. Therefore, in Bilstm and classifier networks, the meaning of the word will not change. From the perspective of mathematics, the fitting effect of learning sentence meaning through Bilstm from word-formation to problem classification will be worse, while a classification effect can learn new word meaning along with the whole network and change the weight of word vector will be better.

3.2.3 In-depth analyses

After completing the test of two basic models, several in-depth analyses were conducted. For the first stage, we test the accuracy of the model trained by varying sizes (1000, 3000, 5000) of the training set. The result shows that the accuracy of both the test set and validation set is reduced to varying degrees (78% 71% 64%) and this trend is roughly proportional to the size of the training set.

For the second stage, we took a detailed look at the confusion matrices, finding out that the number of questions of different classes in the test set is quite in-balanced. The largest class has 136 questions while some $enty$: classes do not have any. Subsequently, we only cared about the classes have more than 10 questions to see their classification accuracy, and the result shows that the classes with the lowest accuracy were $enty : other$ 7.7% (1/13) $desc : desc$ 10% (1/10) $enty : veh$ 18% (2/11), and the class with the highest accuracy is $loc : city$ 87% (13/15).

As for the third stage, we trained three models with different stopwords lists. Except for the first model, the other two models used a modified stopwords list, in which the first one contained interrogative words (e.g., what where, when) and another contained several common punctuations (e.g.“, ”). By comparing their accuracy, we figured out that the removal of interrogative words greatly affected the accuracy of the model, the model without the removal of interrogative words is 11% more accurate than the model that did the removal. Conversely, the removal of the punctuation does not seems to have a great impact on the accuracy of the model.

References

Lynette Hirschman and Robert Gaizauskas. 2001. Natural language question answering: the view from here. *natural language engineering*, 7(4):275–300.

Oliver Kramer. 2016. Scikit-learn. In *Machine learning for evolution strategies*, pages 45–53. Springer.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics*.

Zhiyuan Liu, Yankai Lin, and Maosong Sun. 2020. Sentence representation. In *Representation Learning for Natural Language Processing*, pages 59–89. Springer.

Edward Loper and Steven Bird. 2002. Nltk: The natural language toolkit. *arXiv preprint cs/0205028*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

PyTorch. 2021. Sequence models and long short-term memory networks. [OL]. https://pytorch.org/tutorials/beginner/nlp/sequence_models_tutorial.html Accessed March 11, 2022.

Dell Zhang and Wee Sun Lee. 2003. Question classification using support vector machines. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 26–32.

Yin Zhang, Rong Jin, and Zhihua Zhou. 2010. Understanding bag-of-words model: a statistical framework. In *International Journal of Machine Learning and Cybernetics*, pages 43–52.