Duplicate Question Detection based on Neural Networks and Multi-head Attention

Heng Zhang
Shanghai Key Laboratory of Trustworthy Computing
East China Normal University
Shanghai, China
51174500154@stu.ecnu.edu.cn

Liangyu Chen
Shanghai Key Laboratory of Trustworthy Computing
East China Normal University
Shanghai, China
lychen@sei.ecnu.edu.cn

***Abstract*—It is well known that using only one neural network can not get a satisfied accuracy for the problem of Duplicate Question Detection. In order to break through this dilemma, different neural networks are ensembled serially to strive for better accuracy. However, many problems, such as vanishing gradient or exploding gradient, will be encountered if the depth of neural network is blindly increased. Worse, the serial integration may be poor in computational performance since it is less parallelizable and needs more time to train. To solve these problems, we use ensemble learning with treating different neural networks as individual learners, calculating in parallel, and proposing a new voting mechanism to get better detection accuracy. In addition to the classical models based on recurrent or convolutional neural network, Multi-Head Attention is also integrated to reduce the correlation and the performance gap between different models. The experimental results in Quora question pairs dataset show that the accuracy of our method can reach 89.3%.**

***Keywords*-deep learning; multi-head attention; ensemble learning;**

I. Introduction

Duplicate Question Detection(DQD) is an important task of Semantic Textual Similarity(STS) in Natural Language Processing(NLP). It is mainly used to judge whether two problems are similar, so duplicate question detection is a binary classification problem. It is also the basis of many NLP tasks. For example, in automatic Question Answering (QA), by retrieving questions that are semantically equivalent to a question presented by a user, an automatic QA system can answer the user's question with answers of the retrieved questions.

About duplicate question detection, researchers proposed a variety of topic modeling[1], syntactic structure[2] and designed various similarity features based on word embeddings[3][4]. Another approach is to use neural networks such as Feed-Forward Neural Networks(FNN)[5], Convolutional Neural Networks(CNN)[6], Long Short-Term Memory(LSTM)[7], and more complex models[8]. Although the above methods have achieved good results on DQD, each single model has one limitation, that is, it does not match and perform well for all types of texts. Take RNN as an exmaple, RNN is suitable for sequence prediction tasks for texts with arbitrary length, since it uses cyclic joins. However, as the sequence length increases, the more information of the header is forgotten. For example, "Why is argon used in welding?" and "How is argon used in welding?" are two problems with different words in the header of sentence. Although many words are same, the meanings of two sentenses are not equal for the reason that the meanings of "Why" and "How" are different. Therefore, when the RNN is applied, it uses fewer header information and more tail information, which may lead to erroneous prediction.

In this paper, we use ensemble learning with treating different neural networks as individual learners and proposing an integrated mechanism based on a new voting algorithm: Credible Voting. Our voting method can escape from the unstable classification in neural network, that is, ensure classification results same after several rounds of training. In addition to the classical models based on recurrent or convolutional neural network, Multi-Head Attention is also integrated to reduce the correlation and the performance gap between different models. For Quora question pairs dataset, the experimental results show that the classification accuracy of our method can reach 89.3%.

The rest of the paper is structured as follows. In Mixed Multi-head Attention section, we introduce the network structure based on Multi-head Attention. Then in the third section, we describe the common neural network structure in the semantic textual similarity task, which is introduced by Bi-LSTM example. Later we present our new voting algorithm in the fourth section. In the fifth section, we show the experimental process, and the results were analyzed. Finally, we summarize in the sixth section.

II. Mixed Multi-Head Attention

In this section, we describe our mixed multi-head attention model in detail. The framework of the model is illustrared in Figure 1. The model mainly contains three components:encoder layer based on the word embedding, hidden layer based on self-attention, mutual-attention and ResNet[9], and decoder layer based on GlobalMaxPooling[10] and Multi-Layer Perceptron(MLP)[11].

A. Encoding Layer

Like to other textual similarity tasks, we use learned embeddings to convert the input words to vectors. Then we use linear module to convert the matrix into a matrix of dimension d_{model} as the input to the entire neural networks.

978-1-7281-5014-7/19/$31.00

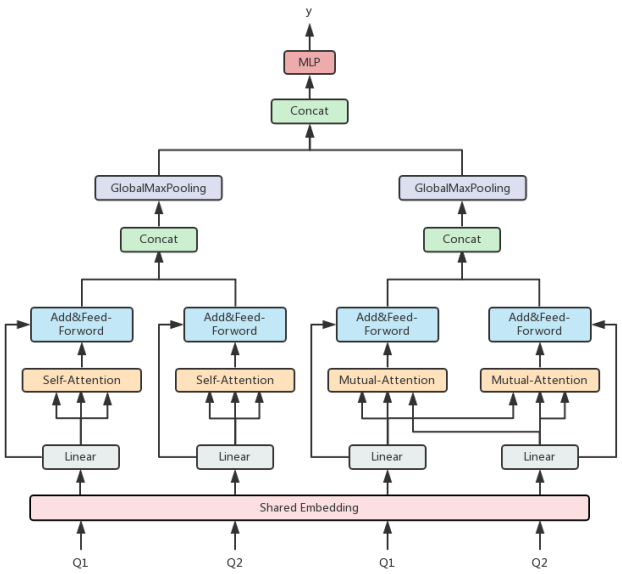

Figure 1: The structure of Mixed Multi-Head Attention

B. Hidden Layer

Hidden layer has two sub-layers. The first is self-attention and mutual-attention, and the second is a simple, ResNet and feed-forward network.

Self-attention and mutual-attention are two variants of multi-head attention, where the difference between them is input vector. Figure 2 depicts the computation graph of multi-head attention mechanism. The attention function can be described as follows:

$$\begin{aligned} &Att(\mathbf{Q},\mathbf{K},\mathbf{V}) = softmax(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}})\mathbf{V}, \\ &head_i = Att(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V), \\ &\mathbf{Q} \in \mathbb{R}^{n\times d_k}, \mathbf{K} \in \mathbb{R}^{n\times d_k}, \mathbf{V} \in \mathbb{R}^{n\times d_k}, i \in h \end{aligned} \tag{1}$$

where $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$ represent the attention query matrix, key matrix and value matrix respectively, n and d_k represent the number of matrices and hidden units of our network respectively. For each head of attention, we denote the learned linear maps by $\mathbf{W}_i^Q \in \mathbb{R}^{d_{model}\times d_k/h}$, $\mathbf{W}_i^K \in \mathbb{R}^{d_{model}\times d_k/h}$ and $\mathbf{W}_i^V \in \mathbb{R}^{d_{model}\times d_k/h}$, which denote to queries, keys and values respectively.

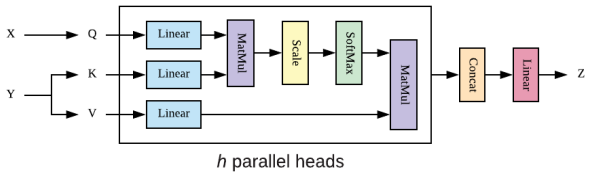

Figure 2: The graph of Multi-Head Attention computation mechanism.

Finally, all vectors produced by parallel heads are concatenated to form a single vector. Again, a linear map is used to combine different channels from different heads:

$$\mathbf{Z} = Concate(head_1, \ldots, head_h)\mathbf{W}^{\mathbf{O}}, \tag{2}$$

where $\mathbf{W}^{\mathbf{O}} \in \mathbb{R}^{d_{model}\times d_k}$.

In this work, we employ $h = 8$ parallel attention layers, or heads. For each of these, we use $d_k = d_{model}/h = 32$. We use two separate self-attentions on Q1 and Q2, respectively, and two separate mutual-attentions between Q1 and Q2.

We attach a ResNet and a feed-forword neural network behind each multi-head attention. In the ResNet, we add the multi-head attention input and its output as an composite input to the feed-forword neural network. There are two linear variations in a full connected neural network, one of which contains a ReLU activation function. The expression is listed as follows.

$$ReLU(x) = max(x, 0), \tag{3}$$

$$FNN(x) = ReLU(xW_1 + b_1)W_2 + b_2, \tag{4}$$

where $W_1 \in \mathbb{R}^{d_{model}\times d_k\times 2}$ and $W_2 \in \mathbb{R}^{d_{model}\times d_k}$.

C. Decoder Layer

For better results, we used GlobalMaxPooling and MLP in the decoder layer. GlobalMaxPooling can reduce the number of parameters to avoid overfitting. MLP is a forward-structured artificial neural network that maps a set of input vectors to a set of output vectors, so MLP can learn in high-dimensional data.The input layer uses ReLU layer as activation function, while the output layer uses sigmoid function. The sigmoid function is listed as follows:

$$sigmod(x) = \frac{1}{1+e^{-x}}. \tag{5}$$

III. Neural Networks Model

Because our focus is on how to integrate different neural networks rather than proposing a novel NNM, we directly use some classic models that perform well on semantic textual similarity tasks. In experiments, many different kinds of neural networks are tried. Here we only introduce a basic one as an example, that is a Siamese network[12] consists of two bidirectional long short-term memory (Bi-LSTM) networks. The structure is shown in Figure 3.

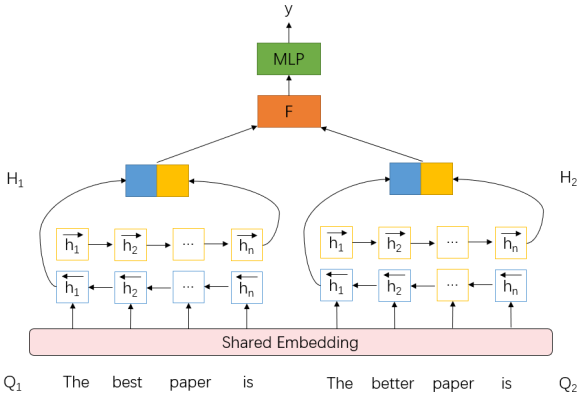

Figure 3: The structure of Bi-LSTM neural network.

At first, as a question Q=$[w_1, w_2, \cdots, w_n]$ can be represented as a sequence of word embeddings which construct a matrix $E = [e_{w_1}, e_{w_2}, \cdots, e_{w_n}]$, where e_{w_n}

is the word embedding of w_n. Existing works use this question matrix as the input of Bi-LSTM. For the t-th word in a question, the LSTM takes the word embedding x_t as input. The detailed computation process is listed as follows:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i), \tag{6}$$
$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f), \tag{7}$$
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o), \tag{8}$$
$$\widehat{c_t} = tanh(W_c[h_{t-1}, x_t] + b_c), \tag{9}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \widehat{c_t}, \tag{10}$$
$$h_t = o_t \odot tanh(c_t), \tag{11}$$

where i_t, f_t, o_t, c_t are input gate, forget gate, output gate and memory cell respectively, σ denotes the sigmoid function, W_i, W_f, W_o, W_c denote weight matrices, b_i, b_f, b_o, b_c denote biases, $\odot$ denotes the element-wise vector product, and h_{t-1} denotes the output vector at the time step $t-1$.

A Bi-LSTM consists of a forward and a backward LSTM. The hidden states of them are denoted as $(\overrightarrow{h_1}, \ldots, \overrightarrow{h_n})$, $(\overleftarrow{h_1}, \ldots, \overleftarrow{h_n})$ respectively. By connecting the last forward and backward hidden state as an output, we obtain the representation.

$$H = [\overrightarrow{h_n}; \overleftarrow{h_1}]. \tag{12}$$

For duplicate question detection we need to build a connection between the two questions, so we use the function F to reflect the connection between the two questions.

$$F = [|H_1 - H_2|; |H_1 \odot H_2|]. \tag{13}$$

Finally, we use the MLP output prediction value P. The output layer of the MLP uses the sigmoid function.

IV. Ensemble Approach

In this section, we describe how to integrate multiple neural networks. It is well known that CNN generates separate hyperplanes by extracting adjacent words from multiple convolution kernels. RNN generates separated hyperplanes by recording the last state, and Attention is to generate separated hyperplanes by global weighted summation. Obviously, the way MHA generates the separated hyperplane is different from the above three.

Since the optimizer in the neural network mostly uses random sampling, the predicted values of the same data set on different training batches are different. That is, the separated hyperplanes used in different batches of training are fluctuated and not same. It is obviously observed that when the sample distance is very close to the separation hyperplane, the classification result obtained by the separation hyperplane is not credible, because it is likely to be classified into other categories in the next training. In order to solve this problem, this paper proposes a voting algorithm based on credibility: Credible Voting.

In this algorithm, if the predicted value of the model M is closer to 0.5, the predicted value is considered to be incredible. Based on this, we set the incredible offset $\Delta \in [0, 0.5]$ for the array of n model M predicted values consisting of $P = [p_1, p_2, \ldots, p_n]$. If the predicted value $p_i \in [0.5 - \Delta, 0.5 + \Delta]$, the result of the model M_i is considered incredible and is considered as a discard, otherwise can join voting. Finally, if the voting number of positive or negative classification is more than half, the category with the highest vote is selected, otherwise the category of all prediction values farthest from 0.5 is outputed. The description of the algorithm 1 is presented as follows.

Algorithm 1: The description of the Credible Voting algorithm

Input: incredible offset Δ, model number n, model predictors set $P = [p_1, p_2, \ldots, p_n]$
Output: classification y
$zore \leftarrow 0$;
$one \leftarrow 0$;
$half \leftarrow \lfloor n/2 \rfloor + 1$;
for $i = 1, \ldots, n$ **do**
  if $P[i] < 0.5 - \Delta$ **then**
    $zore \leftarrow zore + 1$
  else if $P[i] > 0.5 + \Delta$ **then**
    $one \leftarrow one + 1$
$y \leftarrow argmax([zero, one, half])$
if $y == 2$ **then**
  $closer0 \leftarrow min(P)$
  $closer1 \leftarrow 1 - max(P)$
  $y \leftarrow argmin([closer0, closer1])$
return y

We found that when $\Delta = 0$, there would be no discards, so CV algorithm is equivalent to absolute voting. When $\Delta = 0.5$, all predictions are considered as discards, so the algorithm selects the category farthest from 0.5. We call this with Maximum Credible Voting (MCV).

For example, assume there are five models, who present a set of predicted values of $P = [0.58, 0.1, 0.87, 0.89, 0.22]$ respectively and $\Delta = 0.2$. Then for each $p_i \in P$, if $p_i \in [0.3, 0.7]$, we deem it as a discard, so the final voting result is $negative = 2, positive = 2$, which shows that the number of votes on both sides is not more than half. Therefore, according to the algorithm, we choose the prediction value farthest from 0.5. Finally we output category negative.

CV is based on Voting to increase the discarding mechanism. When there is a rejection prediction, the sample farthest from the separation hyperplane will be selected. The MCV considers all voting as a discard on the basis of CV, and the sample farthest from the separation hyperplane will be directly selected.

V. Experiments

A. Datasets

Quora question pairs (QQP) dataset[1] is adopted in our experiments. It consists of over 400, 000 lines of potential

[1]https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs

question duplicate pairs. Each line contains two IDs and corresponding questions and a label. Then we shuffle the dataset randomly and split into train/dev/test set with a proportion of 8:1:1. The statistics of the question pairs are listed in Table I.

Table I: Statistics of QQP dataset

	Positive	Negative	Total
Train(80%)	119282	204150	323432
Dev(10%)	15010	25419	40429
Test(10%)	14971	25458	40429
Total	149263	255027	404290

B. Experimental Models

In this experiment, we selected some classical neural networks in text similarity tasks. These models have similar overall structures and all use QQP data sets. The main difference between these neural networks is the structure of the neurons. The neurons used in the experiment are as follows.

- **LSTM** has been specifically introduced in the "Neural Networks Model" section.
- **GRU**[13] is similar to LSTM neurons, mainly combining the forgotten gate and the input gate into an update gate. The GRU neurons are described below.

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t]), \tag{14}$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t]), \tag{15}$$

$$\widetilde{h}_t = tanh(W_{\widetilde{h}} \cdot [r_t \odot h_{t-1}, x_t]), \tag{16}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \widetilde{h}_t, \tag{17}$$

  where z_t, r_t are respectively represented as update gates and reset gates.
- **CNN** [14] consists of some fixed-size convolution kernels. The specific formula is described below.

$$h_t = f(W_h \odot x_{t:t+w-1} + b_h), \tag{18}$$

  where f represents a nonlinear activation function, and in the experiment we use the ReLU function. w represents the convolution kernel window size and $x_{t:t+w-1}$ represents a feature matrix consisting of t to $(t + w - 1)$ input eigenvectors.
- **Attention** [15] can give weight values to the hidden state sequences $(h_1, h_2, \ldots, h_n)$ of the above three neurons. The specific expression is listed as follows:

$$e_t = tanh(W_c \cdot h_t + b_c), \tag{19}$$

$$a_t = \frac{exp(e_t)}{\sum_{k=1}^{T} exp(e_t)}, \tag{20}$$

$$c = \sum_{t=1}^{T} a_t \cdot h_t, \tag{21}$$

  where e_t is the hidden state of h_t after the tanh activation function, a_t is the weight value of the t time hidden state normalized by the softmax function and c is the new eigenvector obtained after weighting.

We use the Attention mechanism on Bi-LSTM, Bi-GRU and CNN to generate three new models, namely: AT-BLSTM, AT-BGRU and AT-CNN. In the experiments, we use spaCy[2] as the word segmentation tool, and word embeddings[3] are pre-trained using GloVe[16] with a dimension of 300. All models are trained using Adam[17] optimization method with the learning rate set to 0.001 and the batch size is set to 512.

C. Experimental Results

We set up two groups of experiments. The first group used Accuracy (Acc), Precision (P), Recall (R) and F1 Score (F1) to measure each model. The evaluation indices are defined as follows:

$$Acc = \frac{TP + TN}{TP + FP + FN + TN}, \tag{22}$$

$$P = \frac{TP}{TP + FP}, \tag{23}$$

$$R = \frac{TP}{TP + FN}, \tag{24}$$

$$F1 = \frac{2 \times P \times R}{P + R}, \tag{25}$$

where TP is short for true positive, FP represents false positive, FN stands for false negative and TN is true negative. Table II shows the results of each model under four evaluation indices.

Table II: The indicators for all models

#	Neural Network Model	Acc	P	R	F1
1	Bi-LSTM	85.69	79.87	81.18	80.52
2	AT-BLSTM	86.54	83.15	81.00	82.06
3	Bi-GRU	85.98	81.97	80.52	81.24
4	AT-BGRU	86.65	82.08	**81.91**	82.00
5	CNN	85.21	78.23	81.14	79.66
6	AT-CNN	85.18	81.84	78.92	80.35
7	Multi-Head Attention	**86.83**	**84.07**	81.06	**82.54**

Through Table II, we found that all accuracies of these seven models are around 86%. At the same time, we found that Multi-Head Attention performed best on Acc, P, and R. Additionally, we found that AT-BGRU performs best on R.

The second group of experiments combines Mixed Multi-Head Attention with six neural network models, and compares the difference in accuracy among the three methods of voting, Maximum Credible Voting (MCV) and Credible Voting (CV). In the case of the same number of votes in each voting method, this experiments use negative as default setting. Table III shows the accuracy between the different combinations under the three integration strategies, where each of combinations has Multi-Head Attention.

In the first six rows of Table III, the MCV and CV have higher increasing rate. Mainly because the number of neural networks in the first six rows is even, the same number of votes will occur. In this case, the experiments use negative as default setting.

[2]https://spacy.io/

[3]http://nlp.stanford.edu/data/glove.840B.300d.zip

Table III: Results of the ensemble method. The second column is the accuracy of Voting. The third and fourth columns are the accuracy of our proposed ensemble methods, i.e., Maximum Credible Voting (MCV) and Credible Voting (CV). The improvements are listed in the parentheses.

#	Combined models	Voting	MCV	CV
1	Bi-LSTM	86.6	87.6(+1.0)	87.6(+1.0)
2	AT-BLSTM	87.1	88.2(+1.1)	88.2(+1.1)
3	Bi-GRU	86.8	88.1(+1.3)	88.1(+1.3)
4	AT-BGRU	87.0	88.2(+1.2)	**88.2(+1.2)**
5	CNN	86.1	87.3(+1.2)	87.3(+1.2)
6	AT-CNN	86.6	87.9(+1.3)	87.9(+1.3)
7	AT-BLSTM&Bi-LSTM	88.4	88.4(+0.0)	88.6(+0.2)
8	AT-BGRU&Bi-GRU	88.3	88.4(+0.1)	88.6(+0.3)
9	AT-CNN&CNN	88.1	87.6(-0.4)	88.2(+0.2)
10	Bi-LSTM&CNN	88.3	88.0(-0.3)	88.5(+0.2)
11	Bi-LSTM&Bi-GRU	88.1	87.8(-0.3)	88.2(+0.1)
12	Bi-GRU&CNN	88.2	87.6(-0.6)	88.4(+0.1)
13	AT-BLSTM&AT-BGRU	88.5	88.8(+0.3)	**88.9(+0.5)**
14	AT-BLSTM&AT-CNN	88.2	88.4(+0.2)	88.6(+0.4)
15	AT-BGRU&AT-CNN	88.3	88.4(+0.2)	88.6(+0.3)
16	All models except: AT-CNN&CNN	88.9	88.7(-0.2)	**89.0(+0.1)**
17	All models except: AT-BGRU&Bi-GRU	88.7	88.5(-0.2)	89.0(+0.2)
18	All models except: AT-BLSTM&Bi-LSTM	88.8	88.4(-0.4)	89.0(+0.2)
19	All models except:MHA	88.7	88.6(-0.1)	89.1(+0.4)
20	All models	89.1	88.7(-0.4)	**89.3(+0.2)**

As can be seen from the table III, after the 6 lines, the MCV has a lower accuracy than the voting. The reason for this is:As the number of models increases, the greater the probability of false predictions, the lower the accuracy than Voting.At the same time, the accuracy of CV is always higher than voting, indicating that the CV algorithm is effective.

It can be seen from the results of the last two lines that after adding the MHA model, the three voting algorithms have improved to different degrees, indicating that the MHA model can improve the accuracy of the voting method.

As can be seen from Table III, the accuracy of CV is higher than Voting and MCV. The main reason is that when deep learning models cannot resolve the text, the classification result will be very close to the separation hyperplane.

D. Further Analysis of the Ensemble Approach

In this subsection, we provide some further analysis of the ensemble approach, mainly about the detailed statistics on the 19th line of the Table III.

Figure 4 shows the effect of different Δ on accuracy, where the step size of Δ is 0.05. We found that when $\Delta = 0$, its result is the same as Voting, when $\Delta = 0.5$, the result is the same as MCV and peaks at $\Delta = 0.2$. We perform a detailed analysis of $\Delta = 0$ and $\Delta = 0.2$, and count how many neural networks can correctly predict each problem.

From (a) of Figure 5 we can see that the proportion of four and above models successfully predicted is 89.1%, which is the same as the Voting result. In (b) of Figure 5, we found that due to the discarded votes, the number that

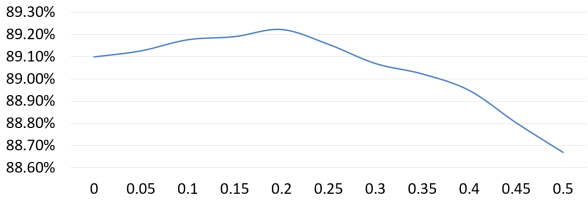

Figure 4: The influence of Δ in accuracy

can be successfully predicted by seven neural networks has experienced a large decline(from 63.51% to 58.48%), and the rest also occurs to varying degrees. According to the description of the algorithm 1, we find that our improvement in accuracy mainly comes from the part of the model whose number is less than or equal to three. From (a) we can see that the proportion of this part is small (only 10.75%), so the overall rate of increase is also relatively small (only 0.2%).

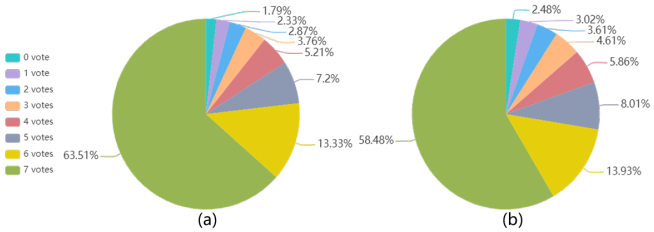

Figure 5: (a) is the number of correct prediction of Voting and (b) is the number of correct prediction of CV

VI. Conclusion

Duplicate question detection is a classic and important binary classification problem. In this paper, we use multi-head attention to process duplicate question detection task and propose the Credible Voting algorithm. The algorithm eliminates the separation hyperplane which is too close to the sample, obtains stable classification results in several rounds of training, and improves the accuracy. Finally, the algorithm can achieve an accuracy of 89.3% on this task.

References

[1] K. Zhang, W. Wu, H. Wu, Z. Li, and M. Zhou, "Question retrieval with high quality answers in community question answering," in *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. ACM, 2014, pp. 371–380.

[2] S. Filice, G. Da San Martino, and A. Moschitti, "Kelp at semeval-2017 task 3: Learning pairwise patterns in community question answering," in *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, 2017, pp. 326–333.

[3] M. Salvador, S. Kar, T. Solorio, and P. Rosso, "Combining lexical and semantic-based features for community question answering," *Proceedings of SemEval*, pp. 814–821, 2016.

[4] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, p. 533, 1986.

[5] P. Nakov, L. Màrquez, and F. Guzmán, "It takes three to tango: Triangulation approach to answer ranking in community question answering," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016, pp. 1586–1597.

[6] P. Wang, L. Ji, J. Yan, D. Dou, N. D. Silva, Y. Zhang, and L. Jin, "Concept and attention-based cnn for question retrieval in multi-view learning," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 9, no. 4, p. 41, 2018.

[7] S. Romeo, G. Da San Martino, A. Barrón-Cedeno, A. Moschitti, Y. Belinkov, W.-N. Hsu, Y. Zhang, M. Mohtarami, and J. Glass, "Neural attention for learning to rank questions in community question answering," in *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, 2016, pp. 1734–1745.

[8] M. Zhang and Y. Wu, "An unsupervised model with attention autoencoders for question retrieval," *arXiv preprint arXiv:1803.03476*, 2018.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[10] M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.

[11] C. M. Bishop *et al.*, *Neural networks for pattern recognition*. Oxford university press, 1995.

[12] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a" siamese" time delay neural network," in *Advances in neural information processing systems*, 1994, pp. 737–744.

[13] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[14] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[15] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *Computer Science*, 2014.

[16] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[17] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.