NEUTag's Classification System for Zhihu Questions Tagging Task

Yuejia Xiang[1(✉)], HuiZheng Wang[1], Duo Ji[2], Zheyang Zhang[1], and Jingbo Zhu[1]

[1] Natural Language Processing Laboratory, Northeastern University, Shenyang, China
xiangyuejia@qq.com,
{wanghuizhen, zhujingbo}@mail.neu.edu.cn,
hldnpqzzy@sina.com
[2] Criminal Investigation Police University of China, Shenyang, China
l8640037173@l68.com

Abstract. In the multi-label classification task (Automatic Tagging of Zhihu Questions), we present a classification system which includes five processes. Firstly, we use a preprocessing step to solve the problem that there is too much noise in the training dataset. Secondly, we choose several neural network models which proved effective in text classification task. Then we introduce k-max pooling structure to these models to fit this task. Thirdly, in order to obtain a better performance in ensemble process, we use an experiment-designing process to obtain classification results that are not similar to each other and all achieve relatively high scores. Fourthly, we use an ensemble process. Finally, we propose a method to estimate how many labels should be chosen. With these processes, our F1 score achieves 0.5194, which ranked No. 3.

Keywords: Multi-label classification · Question tagging · Ensemble learning

1 Introduction

In the automatic tagging task of Zhihu questions, we need to pick out at most 5 labels out of a set which contains more than 25000 labels. Tagging a label of one question means the question belongs to a class which is corresponding with this label. So, we use term 'label' as a synonym for term 'class' in this paper. And main difficulties of this task are shown as follows.

- Zhihu question texts which contain a large number of non-subject-related terms and other noise are too informal to be analysis.
- There are too many classes and the semantic gaps between some classes are so narrow, which observably increase the difficulty of classification.
- As numbers of labels of different questions are frequently different, it is difficult to estimate how many labels should be chosen for a certain question in order to reach a better performance.

M. Zhang et al. (Eds.): NLPCC 2018, LNAI 11108, pp. 279–288, 2018.
https://doi.org/10.1007/978-3-319-99495-6_24

We use a preprocessing process to reduce the adverse effect of noise. This process includes three parts: word segmentation, data cleaning and long-text truncation. Because the neural networks based classification model have been corroborated superior to some traditional classification models [1], notably superior when facing tasks with lots of categories [1–4], we select several neural network based models: RNNText [5], CNNText [6], RCNNText [7], and fastText [8]. What's more, in order to make these neural network-based models more suitable for our task, we introduce k-max pooling structure to them to build a baseline.

In order to obtain better classification results, we use an ensemble process. As we all know, an outstanding ensemble effect requires many classification results that not similar to each other and all achieve relatively high scores [9]. So we use some methods to design experiments, in order to find some results that meet our needs. Moreover, we propose a method to measure the differences between classification results, which can be used to guide the design of experiments.

Finally, we propose a method to estimate how many labels should be chosen. Our method is a post-processing process, as the method forecast the number via the analysis of relations between statistical characteristics of classification results and the number of labels.

The paper is organized as follows. Section 2 introduces details of our system. Section 3 contains our experimental results and analysis. Section 4 includes summarization and future works. Finally, we express our thanks and appreciations.

2 Our System

2.1 Preprocessing

Our preprocessing process includes word segmentation, data cleansing and long-text truncation. Moreover, we divide data cleansing into two parts: a stopword process that is used to filter useless words and a rule-based matching process which is designed to clean up rubbish information such as web sites, page formats, etc. What's more, because neural network models do not perform well in the classification of text which is very long [10], we introduce long-text truncation to suffer this disadvantage [7], while some texts in our training dataset reach tens of times the length of average (Fig. 1).

2.2 Baseline

In order to build our baseline, we choose four models: RNNText [5], CNNText [6], RCNNText [7] and fastText [8], which have been proved effective in text classification tasks. And then make a small change to their structures. The change is using k-max pooling structures to replace max pooling structures [11], in order to get better adaptation to milt-label classification task.

All models we used could be unified with one process: after using embedding layers, they extract context information by CNN-based, RNN-based or NN-based structures, and then they use fully connected layers, named as classifiers to produce classification results. The general structures of these models are shown in Fig. 2.

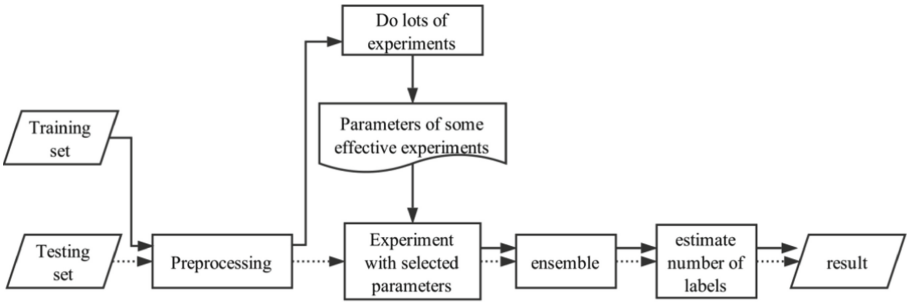

Fig. 1. This is a flow chart of our system, where the solid arrows represent the training process, while the dotted arrows represent the testing process.

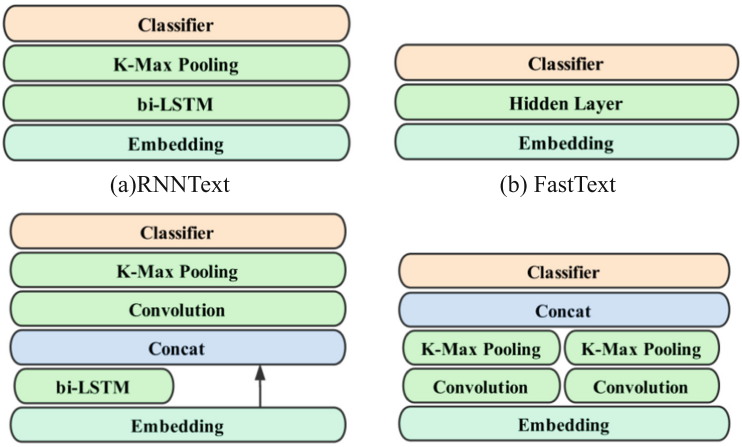

Fig. 2. General structures of models in our baseline

2.3 Design Experiments

In order to find out some classification results that perform well in the following ensemble process, we propose the experiment designing process. In the process, we design lots of experiments by methods listed as follows.

- Use different models in our baseline: RNNText, CNNText, RCNNText or fastText.
- Use different input forms: char-level or word-level. (Different from a char-level method that directly splits original text into character [2], our char-level method splits the text which underwent the word segmentation process and the preprocessing process).
- Use different parameters of model structures: the number of hidden layers, the size of hidden layer units and the batch size.
- Use different training dataset: using additional data or not, a random proportion of data will not be used.

And then we choose classification results that not similar to each other and all achieve relatively high scores, for obtain an outstanding ensemble effect [9]. Moreover, we propose a method to evaluate the differences between classification results, which can be used to guide the designing of experiments. This method evaluates the differences by the difference of normalized accuracy rate distribution on each label (DoNARD), and the algorithm is shown in Algorithm 1.

Algorithm 1

Input: A: experiment result A's accuracy rate on all labels, B: experiment result B's accuracy rate on all labels, T: the number of labels, w: 100000

Output: DoNARD: a number denote the difference between experiment results

begin
for: i = 1, 2, …, T do:

$$x_i = \frac{A_i}{\sum_{t=1}^{T} A_t}, \; y_i = \frac{B_i}{\sum_{t=1}^{T} B_t}$$

$$\text{DoNARD} = w \times \sum_{t=1}^{T} (x_i - y_i)^2$$

end.

We believe that the larger the DoNARD, the larger the difference. By comparing the DoNARD values with scores, we analyze the influence of each method that has been listed. Therefore we use DoNARD to guide the experiments designing process.

2.4 Ensemble

In the ensemble process, the input data is the classification result of our experiment. Firstly, we will assign a weight to each result. Then, we consider the weighted sum of these results as the outcome of our ensemble process. Finally, we use a program to choose these weights' value, in order to get the highest score.

2.5 Estimate Number of Labels

To estimate how many labels should be chosen for each problem is quite difficult in multi-label classification task. Firstly, because probabilities of each label in different questions are tremendous difference, we failed to find out a threshold to judge whether a label should be chosen or not. Secondly, we do not simply estimate the number of labels appeared in the standard answer, but also need to consider the performance of the classification result. For example, top 5 labels of a question are shown in Table 1, and standard labels are 'Healthy', 'Life' and 'Bodybuilding'. The scores for selecting k-top labels are shown in Table 2.

Table 1. Top five labels of the question

Label	Healthy	Bodybuilding	Motion	Travel	Life
Probability	1.841	1.648	1.520	0.9165	0.8426

Table 2. Scores for selecting k-top labels

K-top	1	2	3	4	5
Score	0.6206	0.8000	0.6942	0.6206	0.7843

In this case, when selecting top five or top two labels, we can get higher scores than select top three labels.

In our method, we need to calculate that how many labels are predicted for each question can help us get the highest score in training dataset. After analysis the relationship between statistics of classification results and the optimal number of labels, we find that the sum of probability of top five labels is positive correlation to the optimal number. So we propose a method based on the top five label-probabilities' sum (T5LPS) to estimate the number of labels.

3 Results

3.1 Dataset Sources

Our training dataset includes 721,608 questions from official training dataset and 350,000 additional questions from Zhihu website. Each question includes a title, a description and some labels. There are 25,551 different labels in our dataset.

3.2 Performance Evaluation Indicators

This task uses the positional weighted precision to evaluate performance. Let correct_num_{p_i} denotes the count of predicted tags which are correct at position i, predict_num_{p_i} denotes the count of predicted tags at position i and ground_truth_num denotes the count of correct tags.

$$P = \frac{\sum_{i=1}^{5} \text{correct\_num}_{p_i} / \log(i+2)}{\sum_{i=1}^{5} \text{predict\_num}_{p_i} / \log(i+2)}$$

$$R = \frac{\sum_{i=1}^{5} \text{correct\_num}_{p_i}}{\text{ground\_truth\_num}}$$

$$F_1 = 2 \times \frac{P \times R}{P + R}$$

3.3 Preprocessing

In this part, we compare the effect of each preprocessing method. Firstly, we show the effect of word segmentation which is based on jieba segmentation tool in Table 3. Secondly, we show more experiment results in Table 4.

Table 3. Effect of word segmentation

Model	Score before seg	Score after seg	Score improving
RCNNText	0.3413	0.3455	0.0042
RNNText	0.3598	0.3661	0.0063
fastText	0.2775	0.3583	0.0808
CNNText	0.3213	0.3337	0.0124

Table 4. More experiment results in preprocessing

Experiment	Score change	Score
baseline		0.4070
baseline + word-level	0.0063	0.4133
baseline + word-level + unified expression of all number	–0.0012	0.4121
baseline + word-level + clean all punctuation	0.0054	0.4187
baseline + word-level + stopword	0.0035	0.4168
baseline + word-level + rule-based matching	0.0021	0.4154
baseline + word-level + simplify traditional forms of characters	0.0029	0.4162
baseline + word-level + long-text truncation	0.0011	0.4144
baseline + preprocessing	0.0243	0.4313

We can see that, after segmentation the system score (the highest score of models in our baseline) increases 0.0063. And we find that fastText is more sensitive to segmentation than either RNNText or CNNText, while RCNNText's sensitivity is the least.

The reason is that, as we analysis, single character carries little information which benefits classification, while the fastText's structure do not have Excellent abstract ability provided by deep networks, fastText is more sensitive. [12] Another reason is that the char-level text's length is much bigger than word-level text's. Thus, char-level is more difficult to be learnt [13].

In Table 4, we only show the best result of each method in preprocessing, for example, in stopword method we use a manually selected stopword dictionary with the help of TF/IDF and in the long-text truncation method we use a value which is three times the value of question texts' average length as a truncation threshold. From the experiment results, we get some conclusions that are shown as follows.

- Replacing numbers with uniform expression would lower score, via some labels are sensitive to the value of numbers.
- The method which cleans all punctuations promotes the score, because punctuations do not contribute to classification in this task and there are a lot of useless punctuations which are used as emoticons in the text.
- As long-text truncation brings an improvement, we consider that words in the tail of a long text have little contribution to classification and bring adverse effect to our system because they are so long [13].
- The combination of various preprocessing methods brings an additional score of 0.0030.

3.4 Baseline

We try to apply k-max pooling structure to RNNText, CNNText, RCNNText and fastText models. And experiments are shown in Table 5 that used same preprocessing, based on whole training dataset and additional dataset.

Table 5. Scores of experiments

	RNNText	CNNText	RCNNText	fastText
Max pooling	0.4623	0.4174	0.4557	0.3251
2-max pooling	0.4632	0.4349	0.4425	0.2682
3-max pooling	0.4584	0.4197	0.4441	0.2682

We find that Max Pooling structure performance best in RCNNText and fastText, while 2-max Pooling structure performance best in RNNText and CNNText. But 3-max Pooling structure performance worst. So we introduce 2-max Pooling structure to RNNText and CNNText models. The reason of this phenomenon needs further study whether k-max pooling structure benefit from an better expression ability in multi-label classification task [5].

We can see that 2-max pooling structure is effective in some models,

3.5 Design Experiments

With methods listed in Sect. 2.3, we design hundreds of experiments. Firstly, we check the effectiveness of our char-level method in Table 6. Then with the help of DoNARD method, we analyze the influence on classification results of each method in Table 7. Finally, we show several models in Table 8, with which we obtaining an optimal ensemble effect.

Table 6. Effect of our char-level method

Experiments	Score
RNNText + char-level [2]	33.02
RNNText + our char-level method	35.95

Table 7. The analysis of various changes on RNNText

Change of original model	DoNARD	Ensemble's effect
Using word-level	1.585	0.0107
Double hidden layer size	6.395	0.0082
Adding a hidden layer	0.716	0.0055
Using additional data	2.402	0.0236
30% random data will not be use	1.535	0.0076

Table 8. Several models which are used in our ensemble process

Experiment	Shortened form	Score
fastText + word-level + batch size*0.25 + additional data	FW1	0.4763
RNNText + word-level + hidden size*2 + additional data	LW1	0.4704
RNNText + word-level + additional data	LW2	0.4701
RNNText + char-level + additional data	LW3	0.4561
RNNText + word-level	LW4	0.4352
RCNNText + word-level + randomly not use 30% data	RW1	0.4350
RNNText + char-level + dim*2	LC1	0.4302

Our char-level method is effective, because it removes the noise in original texts, that achieves a better result.

We find that, when the DoNARD is in the range of about (1, 3), the effect of ensemble is better. We consider that if the DoNARD is too large, it means one result is much worse than another, so the effect of ensemble is poor. And if the DoNARD is too small, this indicates that these two results are too similar, so the ensemble not works well. When we use changes such as word-level, size of hidden layer and external dataset, we achieve better performance of ensemble process, so we designed more experiments with these changes.

After our analysis, there are two conclusions which are shown as follows. Firstly, compared 'RNNText + word-level' with 'RNNText + word-level + additional data', we find that using additional data is effective. It can improve about 0.0396 score. Secondly, as fastText achieves the best performance in our experiments, we guess that the hierarchical softmax structure in word2vet benefits most in fastText model and this still needs further works [12].

3.6 Ensemble

Different from translation task, where an ensemble method based on checkpoints of one experiment can yield a boost of performance [14], in this task the ensemble method is useless, as shown in Table 9.

Table 9. Ensemble method based on checkpoints of one experiment

Checkpoint	Checkpoint-1	Checkpoint-2	Checkpoint-3	Checkpoint-4	Ensemble
Score	0.4208	0.4356	0.4267	4229	0.4354

So, we use an ensemble method based on results (best checkpoints) of several experiments. After searching the best weights of classification results in the range of [0.2, 5], we get the highest score which reaches 0.4954. And we show the weights of all classification results in Table 10 with the progressive ensemble performance after ensemble each classification result.

Table 10. Weight of each classification result

Experiment	FW1	LW3	LW1	LW4	LW2	RW1	LC1
Weight	1.12	1.08	1.06	1.06	1.00	1.00	0.96
Ensemble's effect of each step		0.009961	0.005085	0.001124	0.001067	0.001038	0.0008841

We find that the optimal weights that we searched in the range of [0.2, 5] are all close to 1, and the highest score only has 0.0003 higher than using weights that all equal 1. Therefore, it suggests that we should focus on the process of designing experiments instead of focus on searching optimal weights.

3.7 Estimate Number of Labels

With the help of our method, we estimate the number of labels based on T5LPS values, the score improves about 0.0240. Details are shown in Table 11, in which we use T to denote T5LPS value.

Table 11. Details of T5LPS method

Conditions	T > 13	13 > T > 10	10 > T > 6	6 > T > 3	3 > T
Number of Labels	5	4	3	2	1

We expect that the performance can still be improved if we find out better statistics than T5LPS. However, this statistical requires manual screening which is expensive. And we found that the parameters in Table 11 need to be re-tuned manually on training datasets to achieve the best performance for different ensemble results.

4 Conclusions

In our experiments, the effect of all processes evaluated by F1 score is show as follows. 0.0139 from using 2-maxPool structure, 0.0243 from using preprocessing, 0.0450 from designing experiment (including 0.0396 from using additional data), 0.0191 from using ensemble process and 0.0240 from estimating the number of labels. And our conclusions are listed as follows.

- The preprocessing has a significant effect, because reducing noises in texts and shortening the length of texts are beneficial for classification.
- The 2-max pooling structure is effective in multi-label classification tasks.
- The method we proposed to measure the differences between models is useful to guide the designing of experiments.
- The method we proposed to estimate the number of labels is important, as it can promote the performance of the system effectively.
- Using additional training data can improve the performance of classification remarkably.

Acknowledgements. This work was supported in part by the National Project (2016YFB0801306) and the open source project (PyTorchText in GitHub). The authors would like to thank anonymous reviewers, Le Bo, Jiqiang Liu, Qiang Wang, YinQiao Li, YuXuan Rong and Chunliang Zhang for their comments.

References

1. Saha, A.K., Saha, R.K., Schneider., K.A.: A discriminative model approach for suggesting tags automatically for stack overflow questions. In: 10th IEEE Working Conference on Mining Software Repositories, San Francisco, pp. 73–76 (2013)
2. Yang, Z., Yang, D., Dyer, C., He, X., Smola A., Hovy, E.: Hierarchical attention networks for document classification. In: Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, New Orleans, pp. 1480–1489 (2017)
3. Conneau, A., Schwenk, H., Cun, Y.L.: Very Deep Convolutional Networks for Text Classification. arXiv preprint arXiv:1606.01781 (2017)
4. Johnson, R., Zhang, T.: Semi-supervised convolutional neural networks for text categorization via region embedding. Adv. Neural. Inf. Process. Syst. **28**, 919–927 (2015)
5. Zhou, Y., Xu, B., Xu, J., Yang, L., Li, C., Xu, B.: Compositional recurrent neural networks for Chinese short text classification. In: 2016 IEEE, Omaha, pp. 137–144 (2016)
6. Kim, Y.: Convolutional Neural Networks for Sentence Classification. Eprint Arxiv (2014)
7. Lai, S., Xu, L., Liu, K., Zhao, J.: Recurrent convolutional neural networks for text classification. In: AAAI Conference on Artificial Intelligence, Austin, pp. 2267–2273 (2015)
8. Jouling, A., Grave, E., Bojanowshi, P., Mikolov, T.: Bag of Tricks for Efficient Text Classification. arXiv preprint arXiv:1607.01759 (2016)
9. Zhou, Z.H.: Machine Learning. Tsinghua University Press, Beijing (2016)
10. Sundermeyer, M., SchlÜter, R., Ney, H.: LSTM neural networks for language modeling. Interspeech **31**(43), 601–608 (2012)
11. Li, W., Wu, Y.: Multi-level gated recurrent neural network for dialog act classification. In: COLING 2016, Osaka, pp. 1970–1979 (2016)
12. Peng, H., Li, J.X., Song, Y.Q., Liu, Y.P.: Incrementally learning the hierarchical softmax function for neural language models. In: 2016, AAAI, Feinikesi (2016)
13. Kalchbrenner, N., Grefenstette, E., Blunsom, P.: A Convolutional Neural Network for Modelling Sentences. arXiv preprint arXiv:1404.2188 (2014)
14. Sennrich, R., Haddow, B., Birch, A.: Edinburgh neural machine translation systems for WMT 16. WMT16 Shared Task System Description (2016)