WiseTag: An Ensemble Method for Multi-label Topic Classification

Guanqing Liang(✉), Hsiaohsien Kao, Cane Wing-Ki Leung, and Chao He

Wisers AI Lab, Wisers Information Limited, Wan Chai, Hong Kong
{quincyliang,shanekao,caneleung,chaohe}@wisers.com

Abstract. Multi-label topic classification aims to assign one or more relevant topic labels to a text. This paper presents the WiseTag system, which performs multi-label topic classification based on an ensemble of four single models, namely a KNN-based model, an Information Gain-based model, a Keyword Matching-based model and a Deep Learning-based model. These single models are carefully designed so that they are diverse enough to improve the performance of the ensemble model. In the NLPCC 2018 shared task 6 "Automatic Tagging of Zhihu Questions", the proposed WiseTag system achieves an F1 score of 0.4863 on the test set, and ranks no. 4 among all the teams.

Keywords: Topic classification · Tagging · Multi-label

1 Introduction

Multi-label topic classification aims to assign one or more relevant topic labels to a text. It can contribute to many downstream natural language processing applications including recommendation, user profiling and information retrieval. In the NLPCC 2018 shared task 6 "Automatic Tagging of Zhihu Questions task", participants are required to build a multi-label model that assigns relevant tags to a question from a set of predefined topic tags. Specifically, participants are given a training dataset of questions collected from Zhihu, a Chinese community question answering website, where each question in the dataset contains a title, an unique id and an additional description. The predefined tag set contains over 25,000 topic tags, and the task is to assign at most 5 topic tags to each question.

There are two major challenges in this task. First, the high dimensionality and sparsity of the output space increases the difficulty of model training. Second, the quality of the training data is inconsistent, since the questions are tagged collaboratively by users from the Zhihu community. Only the development and testing datasets are relabeled manually for the shared task.

To address the above challenges, we propose an ensemble model combining four single models that are trained based on different features and algorithms. Experimental results on the test set show that the ensemble model is more accurate and robust than the individual models, and ranks no. 4 among all participating teams with a F1 score of 0.4863.

M. Zhang et al. (Eds.): NLPCC 2018, LNAI 11109, pp. 479–489, 2018.
https://doi.org/10.1007/978-3-319-99501-4_47

The remainder of the paper is as follows. Section 2 reviews related work on multi-label topic classification. Section 3 details our WiseTag system including its overall architecture, data pre-processing steps, design of each single model and the ensemble model. Section 4 describes the evaluation setup and experimental results, and finally Sect. 5 concludes the paper.

2 Related Work

Generally, multi-label classification algorithms [12] can be classified into two categories, namely problem transformation methods and algorithm adaptation methods.

Problem Transformation Methods. These methods focus on transforming the multi-label classification problem into other existing well-studied problems. Widely used algorithms include Binary Relevance [13] which transforms the multi-label classification problem into a set of binary classification problems; Calibrated Label Ranking [14] which transforms multi-label classification into label ranking, and Random k-labelsets [15] which transforms the multi-label classification problem into the multi-class classification problem. However, the computation costs of these methods will be very high since there are over 25,000 labels to predict in the task at hand. We therefore do not consider these methods.

Algorithm Adaptation Methods. These methods aim to adapt existing single-label learning algorithms to the multi-label setting. For example, K-Nearest-Neighbors (kNN) has been extended to ML-kNN [16] for multi-label classification, Decision Tree has been extended to ML-DT [17], Support Vector Machine (SVM) has been extended to Rank-SVM [18], etc. After the consideration of the computational cost of the model, we only choose kNN model for further experiments.

More recently, researchers turn to use deep learning-based models for multi-label classification. In the recent Zhihu Machine Learning Challenge [19], all the wining solutions adopt Convolutional Neural Network (CNN) or Recurrent Neural Networks (RNN) models. The evaluation results show that deep learning models achieve the state-of-the-art solution for multi-label classification problem. Therefore, we will consider CNN and RNN model in this paper.

3 System Description

3.1 Overview

Figure 1 depicts the overall architecture of the WiseTag system. The system takes both title and description of a question as input, and generates the top-5 topic tags along with their predicted scores as output. First, the system performs data pre-processing including data cleaning and word segmentation. Second, the pre-processed data are fed into four different topic tagging models, namely the KNN model, the Information Gain (IG) model, the Keyword Matching (KM) model, and the Deep Learning (DL) model respectively. Finally, an ensemble model combines the four prediction results to output the top-5 predicted topic tags and their scores.

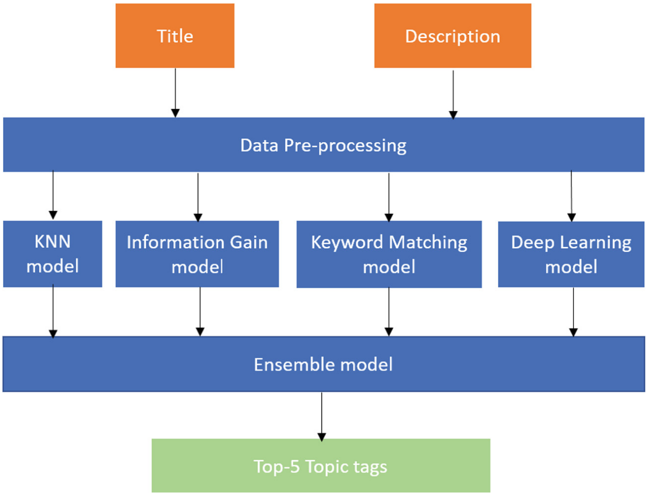

Fig. 1. Overview of the WiseTag system

Table 1. Samples of duplicated data

question_id	question_title	question_detail	tag_names	tag_ids
3926783272	有哪些极短的日剧推荐?		日剧	357
3509771290	有哪些极短的日剧推荐?		日剧	357

Table 2. Samples of question_detail irrelevant to question titles and topic tags

question_detail	Count
如题	3990
本问题已被收录至撰写你的知乎「城市手册」活动，更多活动详情请点击查看。	1031
RT	609

3.2 Data Pre-processing

The following pre-processing steps are performed on each input question:

1. Deduplicate the training data to remove instances having the same question title, descriptions, tag names and tag ids. Table 1 shows a duplicated data example.
2. Remove question descriptions (question_detail) that are redundant or irrelevant to the question titles or topic tags. Table 2 shows some samples of such question_detail which we identified based on data analysis.
3. Convert all characters to half-width and convert all characters to lowercase.
4. Perform word segmentation on question title and description using the Language Technology Platform (LTP) [5] engine.

5. Revise LTP's word segmentation results based on common phrases in the training set identified by the Pointwise Mutual Information (PMI) [8] algorithm.

3.3 KNN Model

KNN [10] is a simple and widely used model for classification, and often proven to be effective for text classification problems. For a given question, we first identify its top 10 ($k = 10$) similar questions and their assigned tags from the training set based on cosine similarity of TF-IDF [11] features. Then, we take the normalized frequencies of these tags as their predicted scores for the given question.

We observe that the KNN model performs well on short questions, but more frequent tags tend to dominate the predictions for new questions.

3.4 Information Gain (IG) Model

Information Gain (IG) is used to measure how much information a word contained in a question provides about the tag of the question. In the training phase, the IG of each word v is first computed for each tag t, denoted as $IG(v, t)$. It is then normalized such that $\sum_{t \in T} IG(v, t) = 1$, where T denotes the set of all predefined tags. In the prediction phase, the normalized IGs of all words in a given question are summed with respect to each tag separately. The summations are then normalized to obtain the predicted scores of the tags. The IG model has built-in feature selection property, therefore, we consider it to be one of our single models.

This model might suffer from over-fitting but has high interpretability because the tags are inferred based on the occurrence of the high-IG words. We observe that it outperforms the other single models except for the Deep Learning model.

3.5 Keyword Matching (KM) Model

We implement a rule-based classifier based on keyword matching. This model is motivated by our observation that some questions are simply labeled using keywords they contained. The KM model counts the predefined tags a given question contains, and takes the normalized tag frequencies as their predicted scores for the question. Table 3 shows some sample questions with their matched keywords (highlighted in red) and predicted tags.

In general, the KM model is able to identify some very specific topics, such as human and school names, thus serves as a good single model candidate for our ensemble model.

Table 3. Samples of keyword matching model prediction

question_title	question_detail	prediction
鹿晗到底做了什么_这么多喷他求科普		p(鹿晗)=0.50, p(科普)=0.50
张飞大战张的宕渠之战，张是否迁民成功?	网上有种，张飞得虚名，张得实惠的说法，是否正确?	p(张飞)=0.67, p(成功)=0.33

3.6 Deep Learning (DL) Model

Deep learning models often report the state-of-the-art performance in text classification tasks, so it is necessary to include one in our ensemble model. During the initial investigation, several DL models including CNN [3], RNN [23] and fastText [2] have been attempted on the given dataset. However, preliminary experimental results showed that RNN and fastText model are difficult to converge, which we attribute to the high dimensionality of the output layer. Thus, we choose CNN as our deep learning model for further experiments.

Figure 2 depicts the architecture of our CNN model, inspired by the 1st Place Solution for Zhihu Machine Learning Challenge [9]. The first layer is an embedding layer with dimension $(150 * 50)$, which allows a maximum of 150 words as input and the embedding size is set to 50. Note that 150 words are able to capture enough information, since the average lengths of question title and description are 13.17 and 70.84 words respectively. On top of the embedding layer, there are five parallel convolution layers with kernel sizes ranging from 1 to 5. The output of the convolution layers are concatenated and fed into a dense layer. A dropout layer with rate 0.5 is added after the dense layer to avoid overfitting. The dense layer is fully connected with the output layer with sigmoid as the activation function. The embedding layer adopts a word2vec embedding pre-trained using the given training set with Gensim [6]. The CNN model is implemented in the Keras framework [7].

3.7 Ensemble Model

To improve classification accuracy, WiseTag uses an ensemble model to combine the four aforementioned models $M = \{KNN, IG, KM, DL\}$ as follows:

$$ensemble\_model = \sum_{m \in M} w_m * m \tag{1}$$

where w_m is the weight assigned to model m.

The final evaluation only accepts up to 5 predicted labels per question. Hence, we output a tag only if it is ranked among the top-5 by our system with a predicted score above a decision threshold. In order to obtain the optimized weights of the different models and the decision threshold, we use the tool Hyperopt [1] for parameter tuning. The Hyperopt supports Tree-structured Parzen Estimators (TPE) algorithm, which is better than random search and grid search [20].

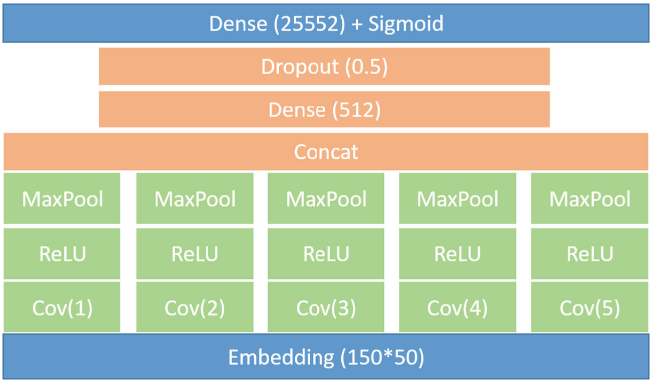

Fig. 2. CNN architecture

4 Experiments

4.1 Dataset

The original training set contains 721,608 instances, which are questions collected from Zhihu. Each question contains a title, an unique id and an additional description; and is tagged collaboratively by users from the Zhihu community. The average lengths of question title and description are 22.23 and 116.29 characters respectively, or 13.17 and 70.84 words respectively after performing word segmentation. There are 3.13 tags per question on average. After deduplication, only 721,531 instances are left for further training. The development and test sets contain 8,946 and 20,596 questions respectively, with their labels manually relabeled for the evaluation task at hand.

4.2 Evaluation Metrics

Each question in the test set will be assigned at most 5 predicted topic tags, sorted by their predicted relevant scores (or probabilities). Performance is evaluated based on the F1 measure with positional weighted precision. Let $correct\_num_{Pi}$ denote the number of correctly predicted tags at position i, and $predict\_num_{Pi}$ denote the number of predicted tags at position i. The precision, recall and F1 measure are computed as follows:

$$F_1 = \frac{2 * P * R}{P + R} \tag{2}$$

$$P = \frac{\sum_{i=1}^{5} correct\_num_{Pi}/log(i+2)}{\sum_{i=1}^{5} predict\_num_{Pi}/log(i+2)} \tag{3}$$

$$R = \frac{\sum_{i=1}^{5} correct\_num_{Pi}}{ground\_truth\_num} \tag{4}$$

4.3 Experimental Results

In order to understand the effectiveness of different modules and parameters, we carry out an extensive series of experiments.

We first study the impact of different pre-processing tasks on classification performance based on the validation set, where the training data is split into two parts, with 90% for training, and the remaining 10% for validation. Specifically, three pre-processing settings are evaluated:

- *raw ltp*: which uses the LTP engine for word segmentation.
- *character conversion + raw ltp*: which performs character conversion (half-width and lowercase normalization) and uses the LTP engine for word segmentation.
- *character conversion + raw ltp + PMI*: which performs character conversion, uses the LTP engine for word segmentation, and then performs the proposed word segmentation revision based on common words identified by the PMI algorithm.

Table 4 presents the results, which show that the proposed revised word segmentation with *character conversion + raw ltp + PMI* settings achieves consistent improvement over the *raw ltp* method under different neural network architectures (Table 5).

Table 4. Impact of pre-processing tasks on classification performance

Pre-processing tasks	Dense layer dimension	Validation score
raw ltp	1024	0.4015
character conversion + raw ltp	1024	0.4031
character conversion + raw ltp + PMI (proposed)	1024	**0.4044**
raw ltp	2048	0.3751
character conversion + raw ltp	2048	0.3758
character conversion + raw ltp + PMI (proposed)	2048	**0.3779**

Next, we empirically evaluate the impact of different parameters on the CNN-based DL model. The results with IDs (2, 3, 4) in Table 4 show the effects of varying the filter number, which produces the best performance when set to 512. The results with IDs (1, 4) show that with the same filter number, a larger kernel size increases classification performance. According to [22], Adam optimizer performs better than Stochastic Gradient Descent (SGD), so we only evaluate Adam with different numbers of the restarts. The results with IDs (4, 5) show that Adam [21] with 2 restarts achieves better validation score than Adam optimizer under the same parameter settings.

Table 5. Impact of different parameters on the DL model

ID	Filter kernel size	Filter number	Optimizer	Validation score
1	2, 3, 4	512	Adam	0.4067
2	1, 2, 3, 4, 5	1024	Adam	0.4066
3	1, 2, 3, 4, 5	256	Adam	0.4045
4	1, 2, 3, 4, 5	512	Adam	0.4083
5	1, 2, 3, 4, 5	512	Adam (with 2 restarts)	**0.4157**

Table 6. Comparison of different single models

Data set	Model	Precision	Recall	F1
Validation	KM	0.1846	0.2005	0.1922
	KNN	0.3949	0.3141	0.3499
	IG	0.3931	0.3784	0.3856
	DL	**0.4373**	**0.3985**	**0.4170**
Dev	KM	0.2395	0.2618	0.2501
	KNN	0.4327	0.3317	0.3755
	IG	0.4141	0.3999	0.4069
	DL	**0.4681**	**0.4218**	**0.4437**

This finding is consistent with that in [4], which reveals that Adam with 2 restarts and learning rate annealing is faster and performs better than SGD with annealing. In particular, we set the learning rate to 0.001 and train the model until convergence. We then halve the learning rate and restart by loading the previous best model.

The optimal parameters of the CNN model we adopted are {'batch size': 128, 'Filter number': 512, 'Kernel size': (1, 2, 3, 4, 5),'Dense layer size': 512, 'Threshold': 0.15 }.

Table 7. Evaluation on dev and test set

Data set	Model	Precision	Recall	F1
Test	Ensemble Model (1)	**0.5048**	0.4692	**0.4863**
	Ensemble Model (2)	0.4878	**0.4839**	0.4858
	Best single model (DL)	0.4715	0.4258	0.4475
Dev	Ensemble Model (1)	**0.5041**	0.4646	**0.4835**
	Ensemble Model (2)	0.4870	**0.4800**	**0.4835**
	Best single model (DL)	0.4681	0.4218	0.4437

Table 8. Model weights in ensemble model

Model/weight	KNN	IG	KM	DL	Threshold
Ensemble (1)	0.28	0.29	0.15	0.28	0.0759
Ensemble (2)	0.3445	0.2803	0.1432	0.2318	0.07812

Table 9. Evaluation results of the task 6 (Top-10 teams)

Rank	Team	Score
1	Tomwindows	0.6271
2	YiWise-QT	0.5840
3	NEUTag	0.5194
4	**WILWAL**	0.4863
5	Team_Wang	0.4840
6	Dream_driver	0.4536
7	iipnku	0.3981
8	HCY_FANS	0.3756
9	scau_AT	0.3404
10	CQUT_301_1	0.3383
...	...	...

In Table 6, we give the comparisons of different single models. The CNN-based DL model outperforms other models on both validation and dev datasets, while KNN and IG achieve comparable results.

Finally, we compare the performance of the ensemble models and the single models as shown in Table 7. Clearly, the ensemble models achieve significantly better results than the best single model on both dev and test datasets. Note that the difference between Ensemble Model (1) and Ensemble Model (2) is that the former use the weights optimized by the Hyperopt tool, while the later uses the weights designed based on rule of thumb that the weight is in proportion to the model's validation score. The weights of ensemble model are shown in Table 8. The final evaluation results of task 6 are presented in Table 9. Our team WILWAL ranks no. 4 with an F1 score of 0.4863 (evaluated on Ensemble Model (1)) on the test set.

5 Conclusion

This paper describes the proposed WiseTag system which performs multi-label topic classification based on an ensemble model. This ensemble model is built upon four diversified models, including: a KNN-based model, an Information Gain-based model, a Keyword Matching-based model and a Deep Learning-based model. Experimental results on the NLPCC-2018 shared task 6 show that the proposed model is effective, and ranks no. 4 with an F1 score of 0.4863. In our future work, we plan to investigate into semi-supervised approaches to multi-label topic classification.

References

1. https://github.com/hyperopt/hyperopt
2. Joulin, A., Grave, E., Bojanowski, P., Mikolov, T.: Bag of tricks for efficient text classification. arXiv preprint arXiv:1607.01759 (2016)
3. Kim, Y.: Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882 (2014)
4. Denkowski, M., Neubig G.: Stronger baselines for trustable results in neural machine translation. In: Proceedings of the First Workshop on Neural Machine Translation, Association for Computational Linguistics, Vancouver, pp. 18–27 (2017)
5. https://github.com/HIT-SCIR/ltp-cws
6. https://radimrehurek.com/gensim/models/word2vec.html
7. Chollet, F.: Keras: deep learning library for theano and tensorflow (2016). https://keras.io
8. https://en.wikipedia.org/wiki/Pointwise_mutual_information
9. https://github.com/chenyuntc/PyTorchText
10. https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm
11. https://en.wikipedia.org/wiki/Tf-idf
12. Zhang, M., Zhou, Z.: A review on multi-label learning algorithms. IEEE Trans. Knowl. Data Eng. **26**(8), 1819–1837 (2014)
13. Boutell, M.R., Luo, J., Shen, X., Brown, C.M.: Learning multi-label scene classification. Pattern Recognit. **37**(9), 1757–1771 (2004)
14. Fürnkranz, J., Hüllermeier, E., Mencía, E.L., Brinker, K.: Multilabel classification via calibrated label ranking. Mach. Learn. **73**(2), 133–153 (2008)
15. Tsoumakas, G., Vlahavas, I.: Random k-labelsets: an ensemble method for multilabel classification. In: Kok, J.N., Koronacki, J., Mantaras, R.L., Matwin, S., Mladenič, D., Skowron, A. (eds.) ECML 2007. LNCS (LNAI), vol. 4701, pp. 406–417. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74958-5_38
16. Zhang, M.L., Zhou, Z.H.: ML-kNN: a lazy learning approach to multi-label learning. Pattern Recognit. **40**(7), 2038–2048 (2007)
17. Clare, A., King, R.D.: Knowledge discovery in multi-label phenotype data. In: De Raedt, L., Siebes, A. (eds.) PKDD 2001. LNCS (LNAI), vol. 2168, pp. 42–53. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44794-6_4
18. Elisseeff, A., Weston, J.: A kernel method for multi-labelled classification. In: Dietterich, T.G., Becker, S., Ghahramani, Z. (eds.) Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic (NIPS 2001), pp. 681–687. MIT Press, Cambridge (2001)
19. https://zhuanlan.zhihu.com/p/28912353
20. Bergstra, J., Bardenet, R., Bengio, Y., Kégl, B.: Algorithms for hyper-parameter optimization. In: Shawe-Taylor, J., Zemel, R.S., Bartlett, P.L., Pereira, F., Weinberger, K.Q. (eds.) Proceedings of the 24th International Conference on Neural Information Processing Systems (NIPS 2011), pp. 2546–2554. Curran Associates Inc., USA (2011)

21. Kingma, D.P., Ba, J.L.: Adam: a method for stochastic optimization. In: International Conference on Learning Representations (2015)
22. http://ruder.io/deep-learning-nlp-best-practices/index.html#fn:21
23. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997). https://doi.org/10.1162/neco.1997.9.8.1735