Can Short Answers to Open Response Questions Be Auto-Graded Without a Grading Rubric?

Xi Yang[1], Lishan Zhang[1,2], and Shengquan Yu[1,2](✉)

[1] Beijing Advanced Innovation Center for Future Education, Beijing Normal University, Beijing 100875, China
{xiyang85, lishan, yusq}@bnu.edu.cn
[2] Faculty of Education, Beijing Normal University, Beijing 100875, China

Abstract. Auto-grading short-answers seems to be sufficiently resolved. However, most auto-graders require comprehensive scoring rubrics, which were not always available. This paper used modern machine learning techniques to build auto-graders without expressly defining the rubrics. The result shows that the best auto-grading model is able to achieve a good inter-rater agreement (kappa = 0.625) with expert grading. The agreement can be further improved (kappa = 0.726) if the auto-grading model gave up scoring some of the answers.

Keywords: Auto-grading · SVM · LSTM · Short-answer

1 Introduction

Automated scoring of short-answer to open response questions has been extensively studied for a long time. C-rater [1] is probably the most well-known system. It performed very well even compared with recent auto-grading algorithms. Its accuracy was 84% on average. Other than auto-grading, short-answer evaluation technique has been also used in the intelligent tutoring systems like AutoTutor [2, 3], where adaptive feedback was selected based on a student's specific answer. The existing methods, including recent one [4], first analyzed student answers using statistical technologies, built scoring models, then the human made models were used for auto-scoring unseen answers. These methods required clear grading rubrics to facilitate auto-scoring. But not all the questions had such clear rubrics ready, especially for those complex ones. Making this kind of rubrics demands domain expertise as well as computing technologies.

With the development of machine learning techniques, we are wondering whether we can build auto-scoring models by only taking human graded answers into consideration and without rubrics. We took Chinese reading comprehension as the study domain, built auto-scoring models only with general syntactic features directly extracted from short answers and explored how many graded answers we need for the algorithms to figure out reliable scoring models. All the answers are divided into three levels of grades. So both accuracy and kappa were used to measure the scoring model.

DOI: 10.1007/978-3-319-61425-0_72

The rest of the paper is structured as follow: first of all, we introduce the machine learning algorithms and data we used. Secondly, we describe how the algorithms were applied in auto-scoring. In the last, the results are presented and we conclude with remarks.

2 The Domain and Data

Our study domain is to auto-score 6th grade students' short answers to reading comprehension questions. We currently only conducted a pilot study for one question, but with 534 student answers. Each student answer can be labeled as one of the three different scores: 0, 1, 2, the higher the better. Two human raters were paid to grade all the questions manually. After clarifying grading criteria, they first graded 50 student answers individually and discussed to resolve all the conflicts. The kappa was 0.783. Then they graded the rest 484 students answers individually and discussed to resolve their conflicts. The overall kappa was 0.755.

3 Methodology

In this paper, we consider the auto-graded student answers problem as a text classification problem. Two classification algorithms are employed, i.e. support vector machine (SVM) and long short-term memory (LSTM) [5].

Long short-term memory (LSTM) is a type of artificial neural network architecture and has got a lot of successes recently in the field of natural language processing (NLP), facilitating many NLP tasks such as machine translation, speech recognition, etc. Like other artificial neural network algorithms, LSTM is consisted of many network units. The unit in LSTM can either remember long or short term duration of time. There are gate units in the network to control how long term units and short term units affect the final outputs.

Support vector machine (SVM) is a well-known classification algorithm. Basically, it managed to draw either linear or non-linear boundaries among the different groups of labeled data points. The boundaries then were used to classify unlabeled data points. The algorithm has been proved to be useful in many different applications.

Both of the two algorithms are supervised learning algorithms. It means that the algorithms need labeled data entries, which are essentially the answers with grades, to calibrate the models. Then, the trained model is used to auto-score the ungraded answers. We used slightly different types of features to train the two models based on their properties. We first built auto-grading models with LSTM and SVM separately, and then blended them together.

To preprocess the input for both algorithms, the answers were first tokenized by using a parser called "jieba" [6], which is a Python Chinese word segmentation module. Then based on the properties of the two algorithms, we adopted different feature engineering methods. Specifically, when SVM was used as the training model, tf-idf score of unigram and bigram was calculated. In a result, it made about 2700 features. While LSTM was used, frequency of each token was calculated to build the feature set. 78 features were used to train LSTM. To achieve a better performance, the two models were blended to form the third model. The average probability of LSTM and SVM for each possible grade was used while blending.

5–fold cross validation was applied to test the effectiveness of the three algorithms. All the answers were essentially classified into 3 categories, accuracy and kappa were used to evaluate the classifiers. To better understand how much data we need to achieve a satisfied accuracy, the size of training data was gradually increased, and the corresponding accuracy was depicted.

Furthermore, we also combined the two original grading models in an innovative way. For each answer, if the two models can make agreement on grading, the combined model output the grade. Otherwise, the combined model gives up.

4 Results and Discussion

LSTM performed slightly better than SVM. The accuracy of SVM model was 0.747 (kappa was 0.588) and the accuracy of LSTM model was 0.755 (kappa was 0.612). Running cross-validation for SVM model took less than 1 s, but running cross-validation for LSTM model took more 1000 s. When the two models were blended, the performance was slightly improved (accuracy was 0.766, kappa was 0.625). The accuracies were relatively low because we had three levels of grades. To further improve the grading accuracy, the two original models were combined in the way described earlier. The combined model graded 77.72% short-answers. Out of the graded answers, the accuracy was 0.836 and the kappa was 0.726.

In order to figure out how much data for each model to achieve a stable performance, the correlation between the size of training data and the accuracy of the classifier was illustrated in Fig. 1. The graphs implied that the trained model started to perform stably after half of the data had been used in training, which was about 267 graded short-answers.

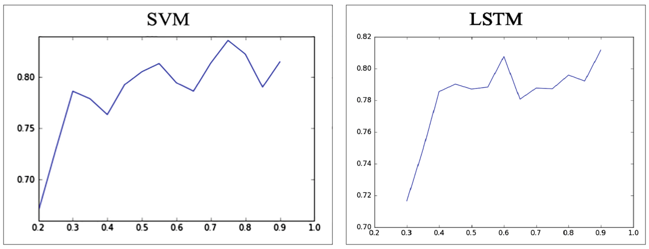

Fig. 1. The correlation between the size of training data and the accuracy

In general, both of SVM and LSTM performed well according to their kappa, and LSTM performed slightly better than SVM in this case. Given that LSTM used much less number of features, and the features were more straightforward, LSTM is probably a better choice in practice. Indeed, training LSTM model took much longer time, but techniques like Hadoop and GPU computing could potentially reduce its training time significantly. Despite of their performance, both of the algorithms need similar size of training data to achieve a stable performance. It means that for a grading task with three different levels, about 300 labeled training entries are need to have the auto-grading mechanism work. It provides an impression of when we should consider establishing auto-grading models. Clearly, it does not make any sense if an instructor only wants to grade answers for a single class. Indeed, more questions are needed to make the estimation of the size of training data more accurate and convincing. But our work perhaps can remind other researchers pay attention on this important aspect while applying machine learning algorithms in education.

When the two models were combined together, although the combined model failed to label 22.28% of the answers, the grading accuracy was improved significantly. In practice, we may have strict criteria on the reliability of grading, but full automation is not required. The innovative combining way then can be considered in this case.

5 Conclusion

With the help of NLP techniques and advanced machine learning algorithms, we managed to auto-grade short-answers without a rubric. By studying the correlations between classifier performance and the size of training data, we made a rough estimation on the size of training data for building stable auto-grading models. We also implemented a method that can significantly improve auto-grading algorithms by sacrificing some automation.

Acknowledgement. We are grateful to Prof. Ningning Zhao for her providing us the dataset.

References

1. Leacock, C., Chodorow, M.: C-rater: automated scoring of short-answer questions. Comput. Humanit. **37**(4), 389–405 (2003)
2. Graesser, A.C., Lu, S., Jackson, G.T., Mitchell, H.H., Ventura, M., Olney, A., Louwerse, M.M.: AutoTutor: a tutor with dialogue in natural language. Behav. Res. Methods **36**(2), 180–192 (2004)
3. Cai, Z., Gong, Y., Qiu, Q., Hu, X., Graesser, A.: Making autotutor agents smarter: autotutor answer clustering and iterative script authoring. In: Traum, D., Swartout, W., Khooshabeh, P., Kopp, S., Scherer, S., Leuski, A. (eds.) IVA 2016. LNCS, vol. 10011, pp. 438–441. Springer, Cham (2016). doi:10.1007/978-3-319-47665-0_50
4. Geigle, C., Zhai, C., Ferguson, D.C.: An exploration of automated grading of complex assignments. In: Proceedings of the Third ACM Conference on Learning@ Scale, pp. 351–360. ACM (2016)
5. Googfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press, Cambridge (2016)
6. Jieba GitHub page. https://github.com/fxsjy/jieba. Accessed 14 Mar 2017