Identifying Current Issues in Short Answer Grading

Tianqi Wang[†,‡] Tomoya Mizumoto[‡] Naoya Inoue[†] Kentaro Inui[†,‡]

[†]Tohoku University [‡]RIKEN Center for Advanced Intelligence Project (AIP)
{outenki, naoya-i, inui}@ecei.tohoku.ac.jp
tomoya.mizumoto@riken.jp

1 Introduction

Short answer grading (SAG) is the task of automatically assessing the short answers to questions, particularly in an educational context. Suppose the following examination question Q from a computer science domain and the reference answer R:

(1) *Q: What is the scope of global variables?*
R: File scope.

Given a *query answer* (e.g. *"The entire program."*, *"main() function."*), the task is to evaluate the correctness of the answer with respect to the reference answer (e.g. 5, 0). SAG is expected to be useful in many real-world applications such as automated assessment of student answers in examinations.

In recent years, a number of datasets have been released such as SciEntsBank [3] and X-CSD [5], which leads to creating a number of computational models for SAG [6, 1]. However, the performance of SAG is still limited, which hampers applying SAG to real-world applications. For example, a state-of-the-art system for SciEntsBank achieved 0.643 of weighted F1 score in 5-ways scoring [7]. Furthermore, it has not been explored what issues remain for creating a better SAG system yet in the literature.

This paper aims at making these issues clear. For this aim, we create a simple SAG system which is easily analyzable but comparable to the state-of-the-art systems. We employ a simple k-Nearest Neighbors (kNN)-based system, where the instances, namely answers, are simply represented by additive word vectors. Our experiments show that the kNN-based system achieves reasonable performance compared to the state-of-the-art approaches. In addition, our detailed analysis of the system's behavior highlights some remaining issues of SAG.

2 kNN-based SAG

2.1 Overall framework

To reveal current challenges in SAG, we would like an SAG system to satisfy at least three requirements: (i) the framework should be transparent enough to analyze; (ii) the confidence value of prediction should be able to be estimated for real world applications; and (iii) the model should not be too simple, but ideally comparable to state-of-the-art approaches.

Given these requirements, we chose to employ k-Nearest Neighbors (kNN) classification algorithm as an overall framework. More specifically, we create a kNN classifier for a single question. Given a question q, set D of score-labeled answers, query answer a, and a distance function d, the task of the kNN classifier is to return the score of a based on k-nearest score-labeled answers in D. To obtain a sophisticated vector representation of score-labeled answers, we train a vector representation of answers in a supervised manner.

2.2 Answer representation

Because the importance of words in answers is different, we represent a vector $\vec{S}(a)$ of an answer a as the weighted sum of word vectors:

$$\vec{S}(a) = \frac{1}{|W(a)|} \sum_{i \in W(a)} w_i \vec{v_i}, \tag{1}$$

where $W(a)$ is a set of words in a, w_i is the weight of word i, and $\vec{v_i}$ is a word vector of i.

We train the word weights w in a supervised manner, so that (i) the distance between answers with different scores are maximized, and (ii) the distance between answers with the same scores is minimized.

Formally, given a set D of score-answer pairs (s_i, a_i) for a single question, we minimize the following loss function:

$$L = \frac{1}{|D|^2} \sum_{(s_i,a_i),(s_j,a_j) \in D \times D} l(a_i, a_j), \tag{2}$$

where $l(a_i, a_j)$ is the loss between a pair of answers a_i, a_j:

$$l(a_i, a_j) = \begin{cases} 1 - sim(a_i, a_j) \text{ if } s_i = s_j \\ sim(a_i, a_j) \text{ otherwise,} \end{cases} \tag{3}$$

where $sim(a_i, a_j)$ is cosine similarity between $\vec{S}(a_i)$ and $\vec{S}(a_j)$.

2.3 Experiments

Settings. To ensure that the present kNN-based approach achieves reasonable performance, we eval-

Dataset	X-CSD [5]	SciEntsBank [3]
Domain	Computer Science	Nature Science
#Questions	87	135
#Reference answers	87	135
#Answers	2,442	4,969
Scores	0-5	2 or 5-ways

Table 1: Statistics of the datasets used in our experiments.

	kNN	State-of-the-art
X-CSD	1.133	0.887 [9]
SciEntsBank (2 ways)	0.757	0.773 [7]
SciEntsBank (5 ways)	0.606	0.643 [7]

Table 2: Comparison of root-mean-square error on X-CSD and weighted average F1 scores on sciEntsBank: results of kNN in our experiments compared to state-of-the-art results.

uate our approach on two popular datasets in SAG: X-CSD [5], and SciEntsBank [3]. The details of these datasets are shown in Table 1. For SciEntsBank, we use the 5-way classification setting. We used Euclidean distance as the distance function of kNN and employed distance-based voting for final decision. We conducted leave-one-out evaluation for each question, where all answers except a query answer are used for training word weights w. We initialize v_i with 300 dimension of GloVe word embeddings[1] and fix them during training. Word weights w are initialized with 0.

Results. The results are listed in Table 2. Considering that our approach is fairly simple, the result indicates reasonably high performance compared to that of the state-of-the-art models [9, 7]. By using the simple approach and model, we can analyze the results to determine the reason behind the performance.

3 Discussion

3.1 Statistics analysis

We show the distribution of prediction in Figure 1. This indicates that the prediction is biased towards five and it is hard for the model to correctly predict the scores of answers with low scores.

This leads to a hypothesis that answers with lower scores are more diverse than those with higher scores. To verify this, for each question, we calculated the average distance between answers for each group of answers with the same scores in X-CSD. Here we represent the answers as:

$$\vec{S}(a) = \frac{1}{|W(a)|} \sum_{i \in W(a)} \vec{v}_i \quad (4)$$

where $W(a)$ is the set of words in a, and $\vec{v}_i$ is the vector of word $i \in W(a)$.

[1] https://nlp.stanford.edu/projects/glove/

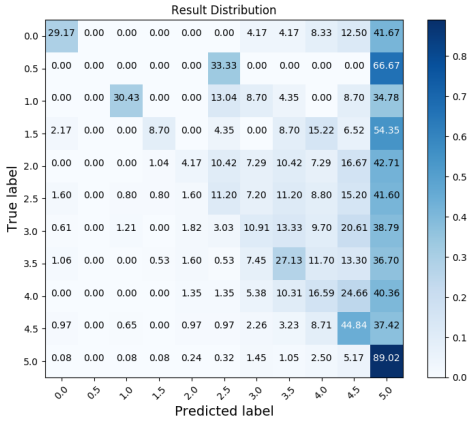


Figure 1: Distribution of multi-label classification results of kNN on X-CSD. The number in each square is proportion in percentage normalized by row.

We calculate the average and standard deviation of distance between each pair of answers with same score for each question, and calculate the mean of average distance and standard deviation for each score. Figure 2 shows the result, which verifies our hypothesis.The mean distance decrease with increase of score, meaning answers with higher score are more similar to each other.

3.2 Error analysis

Figure 3 shows the proportion of errors of results with different confidence on X-CSD. The proportion of answers with errors of 0 increases with confidence, and up to 90% after the confidence gets to 0.99.

We manually analyzed the top 20-most confident errors because the prediction errors with higher confidence are more critical. As the confidence c of prediction, we use the distance between a query answer and its nearest neighbors as follows:

$$c = 1 - \frac{1}{k} \sum_{i=1}^{k} D_i / D_{max}, \quad (5)$$

where k is the number of nearest neighbors, D_i is the distance between a query answer and its i-th nearest neighbor, and D_{max} is the maximum of the distance.

Two major sources of errors are revealed from our analysis, and we show two instances of them in Figure 4.

3.2.1 Miss of key words

We consider highly weighted words as *key words* for classification. Answers consisting of a few words are unlikely to have key words, especially when the score of answer is low. The reasons could be summed to:

- Answers with lower scores are more diverse, as we have discussed with Figure 2. The diversity makes it harder to learn word weights.

Figure 2: Average distance between answers graded to same score. There is very few answers with score 0, hence we did not plot them in the figure.

- The more words a query answer has, the more possible for it to contain words in training answers with learned weights of them. When a query answer has few words, it is less possible to contain a key word learned from the training answers.

For example, in question 4.7 shown in the top of Figure 4, the answer Query#1 is scored to 1 by our model, but the gold-standard score is 2.5. Notice that there are no key words found in this answer, because words such as 'both' and 'sections' do not appear in any answers in training data, hence the weights of them are 0 (i.e. the initial value). Words such as 'stored' are contained in answers with different scores, so they are not highly weighted.

When there is no key word in a query answer, the distance between the query answer and answers in training data will be decided as following:

- Answers with few key words contained are easy to be far away from the query answer in feature space, because the distance between the key word and the words in the query answer are emphasized by the weights of key words. For example, the score of Train#28 is the same as the gold-standard score of the query answer, but it is not a nearest neighbor because of its key words.

- Distance between query answer and answers containing no key word is decided by their common words. Because the weight of none word is greater than others, the more common words a pair of answers have, the more similar they are. Take Train#30 as an example, it is the nearest neighbor of query answer because of it has the common words 'they are stored in' with the query answer.

- For answers with many key words, the distance to the query answer depends to the key words. The representation of answers are effected by all the key words, making the distance to the query answer difficult to predict, like Train#1 and Train#4.

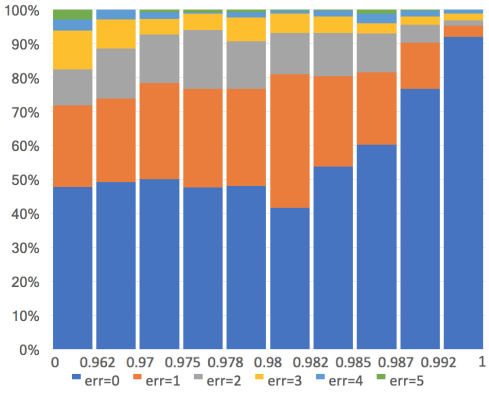


Figure 3: Proportion of errors with various confidence of results on X-CSD.

3.2.2 Incorrect key words

In the instance of question 9.6 shown in the bottom of Figure 4, the word 'enqueue' and 'dequeue' are found as key words, because they only appear in answers with some certain scores. On the other hand, weights of words like 'push' tend to be 0 because they appear in answers with different scores, which means that they are not important for classification.

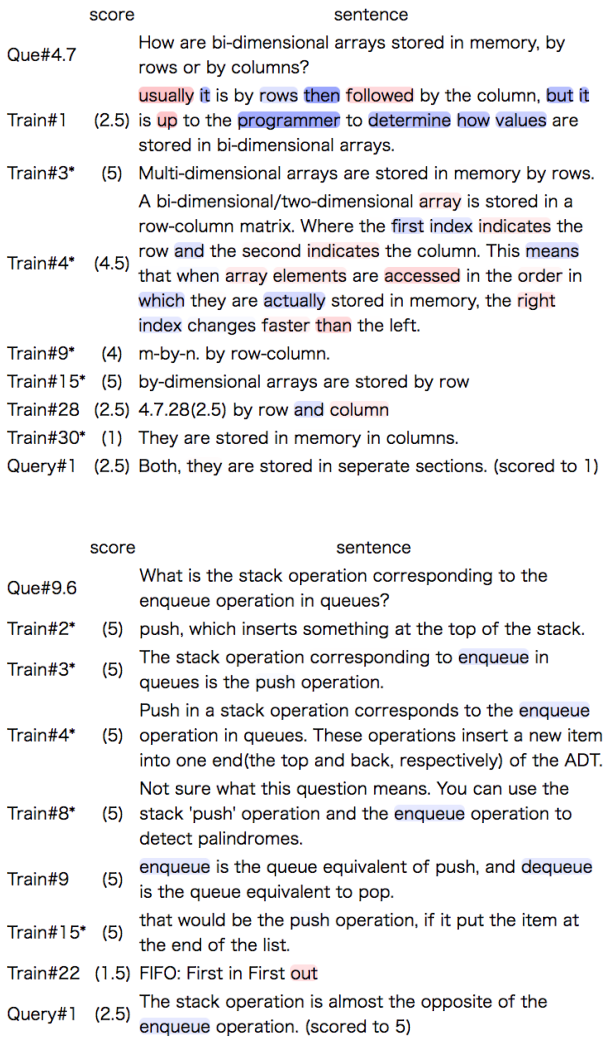
The meaning of sentences containing one same key word may be different depending on the context of it, as what happened to the query answer in this instance. The true score of answer Query#1 is 2.5, but it is scored to 5. It contains one single key word 'enqueue', and the other words are weighted near to 0. The vector generated for query answer is close to the word vector of 'enqueue', hence it is near in feature space to almost answers containing 'enqueue', such like Train#8, Train#4 and Train#3. But the important words to represent the query answer is not 'enqueue' but 'opposite of the enqueue', which caused the error. Moreover, errors caused by incorrect key words are easier to happen to answers whose length is very short, because of the lack of key word.

3.2.3 Possible solution

For the first reason of errors, we could use weight of words in training data that are similar to words in query answers as the learned weight, to avoid the lack of key word. For the second one, weights of phrases could be computed. For example, we could learn the weight of 'opposite of the enqueue' to avoid the problem happened to instance of question 9.6. The information of sentence structure such as dependency graph may also help solve the problem.

4 Related Work

Roy et al.[8] proposed an approach for mining common patterns between student answers and assumed that such commonalities are characteristics of correct answers. Mohler et al.[5] introduced a set of features based on dependency graphs and word similarities



	score	sentence
Que#4.7		How are bi-dimensional arrays stored in memory, by rows or by columns?
Train#1	(2.5)	usually it is by rows then followed by the column, but it is up to the programmer to determine how values are stored in bi-dimensional arrays.
Train#3*	(5)	Multi-dimensional arrays are stored in memory by rows.
Train#4*	(4.5)	A bi-dimensional/two-dimensional array is stored in a row-column matrix. Where the first index indicates the row and the second indicates the column. This means that when array elements are accessed in the order in which they are actually stored in memory, the right index changes faster than the left.
Train#9*	(4)	m-by-n. by row-column.
Train#15*	(5)	by-dimensional arrays are stored by row
Train#28	(2.5)	4.7.28(2.5) by row and column
Train#30*	(1)	They are stored in memory in columns.
Query#1	(2.5)	Both, they are stored in seperate sections. (scored to 1)

	score	sentence
Que#9.6		What is the stack operation corresponding to the enqueue operation in queues?
Train#2*	(5)	push, which inserts something at the top of the stack.
Train#3*	(5)	The stack operation corresponding to enqueue in queues is the push operation.
Train#4*	(5)	Push in a stack operation corresponds to the enqueue operation in queues. These operations insert a new item into one end(the top and back, respectively) of the ADT.
Train#8*	(5)	Not sure what this question means. You can use the stack 'push' operation and the enqueue operation to detect palindromes.
Train#9	(5)	enqueue is the queue equivalent of push, and dequeue is the queue equivalent to pop.
Train#15*	(5)	that would be the push operation, if it put the item at the end of the list.
Train#22	(1.5)	FIFO: First in First out
Query#1	(2.5)	The stack operation is almost the opposite of the enqueue operation. (scored to 5)

Figure 4: Error instances of question 4.7 (top) and question 9.5 (bottom). Words in colors are highly weighted, where the blue color stands for positive weights and the red color stands for negative. Deeper color means greater value weight. Nearest neighbors around the query answer is marked with '*'.

for calculating the distance between student answers and model answers. Adhya and Setua [1] computed the similarity based on the friendship graph. In [2], the authors proposed generic text similarity features including alignment, semantic vector similarity, and length ratio to calculate similarities between student answers and model answers. As mentioned previously, many works generally score the student answers based on the distance to the model answer. Magooda et al.[4] use similarity features of word-to-word similarity and text-to-text similarity for generating a vector of sentences. They then score the answers based on the cosine similarity between student answers and model answers.

Considering the datasets we used in our analysis work are popular in the related researches, the findings on properties of short answers such as diversity and the influence from the short length may also help understand the performance of other approaches on SAG problem.

5 Conclusion

In this work, we have analyzed the results of a simple SAG approach in order to observe the issues for the task of SAG. We used kNN to score query answers, where vector representations of answers are generated from weighted, pre-trained word embeddings. While our approach is simple and transparent enough for analyzing its behavior, our approach is comparable to other state-of-the-art SAG approaches, especially for 2-way classification. By analyzing the errors of our approach, we showed how the diversity and short length of answers caused problems to SAG. Statistics analysis also showed some properties of short answer scoring such as diversity of answers.

Acknowledgement This work was partially supported by JSPS KAKENHI Grant Number 16H06614.

References

[1] Soumajit Adhya and SK Setua. Automated short answer grader using friendship graphs. In *Proceedings of ACITY 2016*, volume 6, pages 13–22, 2016.

[2] Eric Brill, Susan Dumais, and Michele Banko. An analysis of the askmsr question-answering system. In *Proceedings of the ACL-EMNLP 2002*, pages 257–264. Association for Computational Linguistics, 2002.

[3] Myroslava O Dzikovska, Rodney D Nielsen, Chris Brew, Claudia Leacock, Danilo Giampiccolo, Luisa Bentivogli, Peter Clark, Ido Dagan, and Hoa T Dang. Semeval-2013 task 7: The joint student response analysis and 8th recognizing textual entailment challenge. Technical report, NORTH TEXAS STATE UNIV DENTON, 2013.

[4] Ahmed Ezzat Magooda, Mohamed A Zahran, Mohsen Rashwan, Hazem M Raafat, and Magda B Fayek. Vector based techniques for short answer grading. In *Proceedings of FLAIRS Conference*, pages 238–243, 2016.

[5] Michael Mohler, Razvan Bunescu, and Rada Mihalcea. Learning to grade short answer questions using semantic similarity measures and dependency graph alignments. In *Proceedings of ACL 2011*, pages 752–762. Association for Computational Linguistics, 2011.

[6] Michael Mohler and Rada Mihalcea. Text-to-text semantic similarity for automatic short answer grading. In *Proceedings of the EACL 2009*, pages 567–575. Association for Computational Linguistics, 2009.

[7] Brian Riordan, Andrea Horbach, Aoife Cahill, Torsten Zesch, and Chong Min Lee. Investigating neural architectures for short answer scoring. In *Proceedings of the BEA 2017*, pages 159–168, 2017.

[8] Shourya Roy, Sandipan Dandapat, Ajay Nagesh, and Y Narahari. Wisdom of students: A consistent automatic short answer grading technique. In *Proceedings of 13th ICON-2016*, page 178, 2016.

[9] Md Arafat Sultan, Cristobal Salazar, and Tamara Sumner. Fast and easy short answer grading with high accuracy. In *Proceedings of HLT-NAACL 2016*, pages 1070–1075, 2016.