A Memory-Augmented Neural Model for Automated Grading

Siyuan Zhao
Worcester Polytechnic Institute
Worcester, MA 01609, USA
szhao@wpi.edu

Yaqiong Zhang
Worcester Polytechnic Institute
Worcester, MA 01609, USA
yzhang19@wpi.edu

Xiaolu Xiong
Worcester Polytechnic Institute
Worcester, MA 01609, USA
xxiong@wpi.edu

Anthony Botelho
Worcester Polytechnic Institute
Worcester, MA 01609, USA
abotelho@wpi.edu

Neil Heffernan
Worcester Polytechnic Institute
Worcester, MA 01609, USA
nth@wpi.edu

ABSTRACT
The need for automated grading tools for essay writing and open-ended assignments has received increasing attention due to the unprecedented scale of Massive Online Courses (MOOCs) and the fact that more and more students are relying on computers to complete and submit their school work. In this paper, we propose an efficient memory networks-powered automated grading model . The idea of our model stems from the philosophy that with enough graded samples for each score in the rubric, such samples can be used to grade future work that is found to be similar. For each possible score in the rubric, a student response graded with the same score is collected. These selected responses represent the grading criteria specified in the rubric and are stored in the memory component. Our model learns to predict a score for an ungraded response by computing the relevance between the ungraded response and each selected response in memory. The evaluation was conducted on the Kaggle Automated Student Assessment Prize (ASAP) dataset. The results show that our model achieves state-of-the-art performance in 7 out of 8 essay sets and can be trained efficiently due to the simplicity of model structure.

ACM Classification Keywords
I.2.7. ARTIFICIAL INTELLIGENCE: Natural Language Processing

Author Keywords
Automated grading; neural networks; memory networks; word embeddings; natural language processing

L@S'17, April 20–21, 2017, Boston, MA, USA

ISBN 123-4567-24-567/08/06. . . $15.00
DOI: http://dx.doi.org/10.475/123_4

INTRODUCTION
Automated grading is a critical part of Massive Open Online Courses (MOOCs) system and any intelligent tutoring systems (ITS) at scale. Many studies have been conducted to improve automated grading for assignments with simple fixed-form answers, short-answers [3, 15, 19, 26, 21], or long-form answers [26, 2, 7, 14]. Some standard tests, such as Test of English as a Foreign Language (TOEFL) and Graduate Record Examination (GRE), assess student writing skills. Manually grading these essay will be time-consuming. Thus automated essay scoring (AES) systems has been used in these tests to reduce the time and cost of grading essays. Moreover, as massive open online courses (MOOCs) become widespread and the number of students enrolled in one course increases, the need for grading and providing feedback on written assignments are ever critical.

As part of the automated grading system, AES has employed numerous efforts to improving its performance. AES uses statistical and Natural Language Processing (NLP) techniques to automatically predict a score for an essay based on the essay prompt and rubric. Essay writing is usually a common student assessment process in schools and universities. In this task, students are required to write essays of various length, given a prompt or essay topic.

Most existing AES systems are built on the basis of predefined features, e.g. number of words, average word length, and number of spelling errors, and a machine learning algorithm [4]. It is normally a heavy burden to find out effective features for AES. Moreover, the performance of the AES systems is constrained by the effectiveness of the predefined features. Recently another kind of approach has emerged, employing neural network models to learn the features automatically in an end-to-end manner [29]. By this means, a direct prediction of essay scores can be achieved without performing any feature extraction. The model based on long short-term memory (LSTM) networks in [29] has demonstrated promise in accomplishing multiple types of automated grading tasks.

Neural Networks have achieved promising results on various NLP tasks, including machine translation [1, 5], sentiment analysis [6], and question answering [13, 31, 18, 28]. Neural Network models, in terms of NLP tasks, use word vectors to learn distributed representations from text. The advantages are that these models do not require hand-engineered features and can be trained to solve tasks in an end-to-end fashion.

Recent work [29] has exploited several Recurrent Neural Network (RNN) models to solve AES tasks. The results show that neural-based models outperform even strong baselines. Memory Networks (MN) [31, 18, 28] have been recently introduced to deal with complex reasoning and inferencing NLP tasks and have been shown to outperform RNNs on some complex reasoning tasks [28]. MN is a class of models which contains an external scalable memory and a controller to read from and write to that memory. The notion of neural networks with memory was introduced to solve complex reasoning and inferring AI-tasks which require remembering external contexts. Some work [18, 28] has shown the success of MN on different kinds of tasks, e.g. bAbI tasks [30], MovieQA, and WikiQA [18].

To our knowledge, no study has been conducted to investigate the feasibility and effectiveness of MN applied in automated grading tasks. In this study, we develop a generic model for such tasks using Memory Networks inspired by their capability to store rich representations of data and reason over that data in memory. For each essay score, we select one essay exhibiting the same score from student responses as a sample for that grade. All collected sample responses are loaded into the memory of the model. The model is trained with the rest of student responses in a supervised learning manner on these data to compute the relevance between the representation of an ungraded response and that of each sample. The intuition is that as a part of a scoring rubric, a number of sample responses of variable quality are usually provided to students and graders to help them better understand the rubric. These collected responses are characterized with expectations of quality described in the rubric. The model is expected to learn the grading criteria from these responses. We evaluate our model on a publicly available essay grading data set from the Kaggle Automated Student Assessment Prize (ASAP) competition (https://www.kaggle.com/c/asap-aes). Our experiments show that our model achieves state-of-the-art results on this dataset and training of the model is found to be efficient and cost-effective.

The rest of the paper is organized as follows. Section 2 gives an overview of related work in this research area. Section 3 provides detailed information of our model. Section 4 describes the ASAP dataset and evaluation metrics used to test our framework. Furthermore, it contains the details of our implementation and experimental setup to help other researchers replicate our work. In section 5, we present the results of our framework and compare them with other models. Finally, we discuss the results and conclude the paper.

RELATED WORK

Automated Grading

MOOCs were introduced in 2008 and become more popular recently. Most MOOCs systems provide automated grading as their important features to prove the efficiency of their interaction with massive number of online users. Some specific assignment types have been adopted for automated grading since the correct answers of these kinds of assignments have some simple fixed-forms, such as multi-choice questions. Programming assignments are the represents of these kinds of assignments with simple form answer such as "yes" or "no" [8, 11]. Not satisfied with providing answers for one specific assignment, more efforts have been devoted to providing feedback on many different assignments according to the shared features of the programming codes [22, 25].

However, many assignment types cannot be responded well only with simple feedback. Some studies have been conducted with the attempt to fixing this problem by using semi-automatic grading approach. This kind of approach aims to optimize the collaboration between humans and machines and provide short-answers [20, 3]. Another approach is to provide prediction directly. One research direction of this approach is to apply information extraction techniques to constructing specific answer patterns manually or to training from large training dataset with strong supervision support [2,3,24]. Another direction is to compare the students' answers with a established standard answer with an unsupervised text-similarity approach [21].

Most studies mentioned above are dealing with simple fixed-form answers or short-answers assignments. Some complex assignments have long form answer instead of short, simple one. Essay writing with a given topic is a typical assignment with long form answers and AES has become one important research branch of automated grading system.

AES is generally treated as a machine learning problem. We can group the existing AES solutions from different points of view. Most developed AES system is based on a number of predefined features. These features include essay length, number of words, lexicon and grammar, syntactic features, readability, text coherence, essay organization, and so on [4]. Recently, there emerges another trial to treat the whole essays as inputs and learn the features automatically in an end-to-end manner [29]. Without pres-working on features extraction, work burden was lightened. Moreover, the predicting accurate is improved by removing the dependency of effectiveness of predefined features.

Based on learning techniques utilized in existing solutions, we divide them into three categories: regression based approach, classification based approach and preference ranking based approach. PEG-system and E-rater are two examples that belong to regression based approach. Specifically, when the scores range of the essays is wide, the regression based approach is normally adopted since it treats the essay score as a continuous value.

Besides essay writing, some complex assignments such as medicinal assignments utilized regression model as well [9].

Some work such as [27, 14] treated the AES task as a classification problem. Each possible score is converted into a class label. By using classic classification algorithms, AES system predicts which class an essay should belongs to. Since it treats each score as a class label, this kind of approach is not suitable for a very large range of scores. Recently preference ranking based approach was also proposed by [33].

According to the prompts or essay topics the AES system deals with, the existing solutions can be divided into two groups: prompt-specific and generic. The prompt-specific approach train the AES system with essays from one specific topic. This kind of AES system normally has excellent performance on the specific topic it was trained. Most of existing works belongs to this prompt-specific approach. Generic approach train the AES system with essays from different prompts. As an example, the work of [4] proposed a domain adaptation technique which is based on Bayesian linear ridge regression, to achieve a generic prompt adaption AES system. This kind of approach normally neglects the prompt related features but focus on writing quality.

Memory Networks

Memory Networks (MN) [31] and Neural Turing Machines (NTM) [10] are two classes of neural networks models with external memory. MN store all information (e.g. knowledge base, background context) into external memory, assign a relevance probability to each memory slot using content-based addressing schemes, and read contents from each memory slot by taking the their weighted sum with relevance probabilities. End-to-End Memory Networks (MemN2N) [28] can be trained end-to-end compared to MN, and hence require less supervision. Key-value Memory Networks [18] have a key-value paired memory and is built upon MemN2N. Key-value paired structure in memory is a generalized way of storing content in memory. The contents in key memory are used to calculate the relevance probabilities. The contents in value memory are read into the model to help make the final prediction.

NTM form another family of neural networks models with external memory. The NTM controller uses both content and location-based mechanism to access the memory. On the other hand, MN only uses content-based mechanism. The fundamental difference between these two models is the MN do not have a mechanism to change the content in memory, while the NTM can modify the content of the memory in each episode. This leads to the fact that MN is easier to be trained in practice.

MODEL

An illustration of our model is given in Figure 1, which is inspired by the work of memory networks applied in question answering [18, 28]. Our model is comprised of four layers: input representation layer, memory addressing layer, memory reading layer, and output layer. Input representation layer is responsible for generating a vector representation for a student response. Memory addressing layer loads selected samples of student responses to memory, and assigns a weight to each memory piece. Afterward memory reading layer gathers the content from memory by taking weighted sum of each memory piece based on the weights calculated from previous layer, and produces a resulting state. Finally the output layer makes the prediction on the basis of the resulting state. Neural networks models are usually featured with multiple computational layers to learn a more abstract representation of the input. Our model is extended to have the structure of multiple layers (hops) by stacking memory addressing layer and memory reading layer repeatedly.

Input Representation

Each student response is represented as a vector in our model. Given a student response $x = \{x_1, x_2, x_3, ..., x_n\}$, where n is the length of the response, we map each word into a word vector $w_i = Wx_i$. All word vectors come from a word embedding matrix $W \in R^{d \times V}$, where d is the dimension of word vector and V is the vocabulary size. To represent an essay in a vector, we selected position encoding (PE) described in [28]. By the scheme of PE, the vector representation of a response is calculated by $m = \sum_j l_j \cdot Wx_{ij}$, where $\cdot$ is an element-wise multiplication. l_j is a column vector with the structure $l_{kj} = (1 - j/J) - (k/d)(1 - 2j/J)$ (assuming 1-based indexing), where J is the total number of words in the response, d is the dimension of word vector. PE is a simple and efficient way to represent a response, and does not need to learn extra parameters. It has been used in other tasks [32].

Alternative way to represent a response is to feed each word vector from a response into Recurrent Neural Networks (RNN) [29]. Compared to traditional forward neural networks, hidden states of RNN are able to retain the sequential information. By feeding a response into RNN, all the information which are useful for the grading ideally should be stored in the hidden states. Instead of taking the last hidden state as the essay representation, it is recommended to calculate the mean of all hidden states to retrieve the representation for a long response.

Memory Addressing

After generating the representation of the responses, we select a sample from student response for every possible score, which is graded with the same score. The selected samples work as a representation of the criteria in the rubric for all possible scores. Expert knowledge can be used here to choose most representative sample for each score or even generate a number of ideal samples. The motivation is that the model is highly likely to distinguish the difference within the criteria for each score with these representative samples. For our experiment, we randomly pick a sample from student responses for each score, which is graded with that score.

All sampled responses are loaded into the memory as an array of vectors $m_1, m_2, , m_h$, where h is the total number of sampled essays. An ungraded response is denoted as x. The basic idea of memory addressing is that it assigns a weight/importance to each sampled response m_i by calculating a dot product between x and m_i followed by a softmax.

$$p_i = Softmax(xA^T \cdot m_i B^T) \tag{1}$$

where $Softmax(y_i) = e^{y_i} / \sum_j e^{y_j}$, A is a $k \times d$ matrix and so is B. Defined in this way p is a weight vector over all sampled responses. A and B are learned matrices used to transfer the

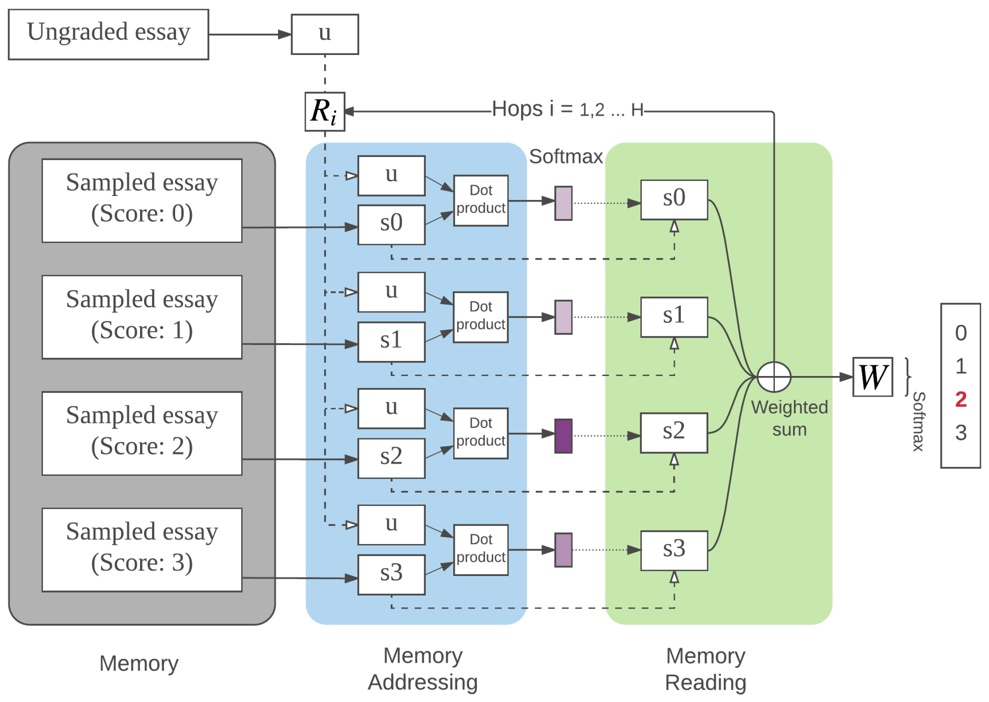

Figure 1. An illustration of memory networks for AES. The score range is 0 - 3. For each score, only one sample with the same score is selected from student responses. There are 4 samples in total in memory. Input representation layer is not included.

response representation to a d-dimensional features space. The intuition is that the responses with the same grade are highly likely to have the similar representation in the feature space.

Memory Reading

After weight vector p is calculated, the output of the memory is computed as a weighted sum of each piece of memory in m:

$$o = \sum_i p_i m_i C^T \tag{2}$$

where C is a $k \times d$ matrix used to transfer the response representation to the feature space. The $k \times d$ matrix C may be identical to A, but from our experiment, we found that training a separate C leads to a better performance. From the equation, we can see that weight vector p controls the amount of content that is read from each memory piece.

Multiple Hops

The success of neural networks is due to its ability of learning multiple layers of neurons and each layer can transform the representation at previous level into a higher level of abstract representation. Inspired by this idea, we stack multiple memory addressing step and memory reading step together to handle multiple hops operations.

After receiving the output o from equation 2, the ungraded response u is updated with:

$$u_2 = Relu(R_1(u+o)) \tag{3}$$

where R_1 is a $k \times k$ matrix, $u = xA^T$ and $Relu(y) = max(0,y)$. Then memory addressing step and reading memory step are repeated, using a different matrix R_j on each hop j. The memory addressing step is modified accordingly to use the updated representation of the ungraded response.

$$p_i = Softmax(u_j \cdot m_i B) \tag{4}$$

Output Layer

After a fixed number H hops, the resulting state u_H is used to predict a final score over the possible scores:

$$\hat{s} = Softmax(u_H W + b) \tag{5}$$

where W is $k \times r$ matrix, r is the number of possible scores and b is the bias value. Note that the number of output nodes equals to the length of score range. We calculate a distribution over all possible scores and select most probable score as the prediction.

The whole network is trained in end-to-end fashion without any hand-engineered features, and the matrices A, B, C, W and $R_1, ..., R_H$ are learned through backpropagation and stochastic gradient descent by minimizing a standard cross entropy loss between the predicted score $\hat{s}$ and the actual score s.

EXPERIMENTAL SETUP

Dataset

Dataset used in this study comes from Kaggle Automated Student Assessment Prize (ASAP) competition sponsored by

William and Flora Hewlett Foundation (Hewlett). There are 8 sets of essays and each set is generated from a single prompt. All responses collected in the dataset were written by students ranging from grade 7 to grade 10. Score range varies on essay sets. All essays were graded by at least 2 human graders. The average length of the essays differs for each essay set, ranging from 150 words to 650 words. Selected details for each essay set is shown in Table 1.

Evaluation Metric

Quadratic weighted Kappa (QWK) is used to measure the agreement between the human grader and the model. We choose to use this metric because it is the official evaluation metric of the ASAP competition. Other work such as [4, 29, 24] that uses the ASAP dataset also uses this evaluation metric. QWK is calculated using

$$k = 1 - \frac{\sum_{i,j} w_{i,j} O_{i,j}}{\sum_{i,j} w_{i,j} E_{i,j}} \quad (6)$$

where matrices O, w and E are the matrices of observed scores, weights, and expected scores respectively. Matrix $O_{i,j}$ corresponds to the number of student responses that receive a score i by the first grader and a score j by the second grader (the model in our experiment). The weight matrix are $w_{i,j} = (i-j)^2/(N-1)^2$, where N is the number of possible scores. Matrix E is calculated by taking the outer product between the score vectors of the two graders, which are then normalized to have the same sum as O.

Implementation Details

The model was implemented using Tensorflow framework [16]. We used Adam stochastic gradient descent [12] for optimizing the learned parameters. The learning rate was set to 0.002 and batch size for each iteration to 32 for all models. As final prediction layer, we used a fully connected layer on top of output from memory reading layer with a softmax activation function. The model learned the parameters by minimizing a standard cross-entropy loss between predicted score and the correct score.

For regularization we used L2 loss on all learned parameters with lambda set to 0.3 and limited the norm of the gradients to be below 10. Moreover, we added gradient noise sampled from a Gaussian distribution with mean 0 and variance 0.001 when training the memory networks.

We used the publicly available pre-trained Glove word embeddings [23], which was trained on 42 billion tokens of web data, from Common Crawl (http://commoncrawl.org/). The dimension of each word vector is 300. Word2vec [17] is another popular word embedding algorithm and pre-trained word embeddings are also publicly available from this algorithm. As results shown in [23], Glove outperforms word2vec on word analogy, word similarity, and named entity recognition tasks.

5-fold cross validation was used to evaluate our model. For each fold, the data was split into two parts: 80% of the data as the training data and 20% as the testing data. The sampled response for each score is selected from the training data. A model was trained on each essay set due to the fact that score range varies among 8 essay sets. We trained each model for 200 epochs using batch gradient descent.

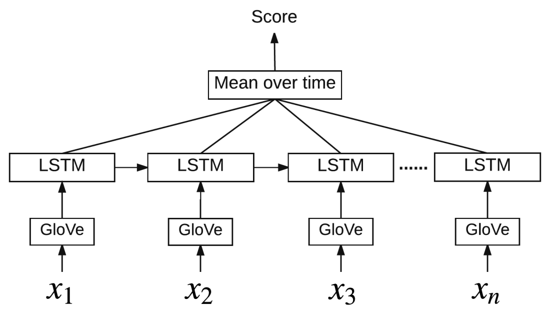

Figure 2. An illustration of baseline LSTM model for AES

Baselines

In [29], their system are compared with Enhanced AI Scoring Engine (EASE), an open-source AES system, to demonstrate the improvements on performance. EASE, like traditional NLP techniques, requires fine-grained hand-engineered features and builds a regression model on top of these features. The reason we use this system as baseline is that it achieved best QWK scores among all open-source systems participated in ASAP competition. [31] described a set of reliable features and reported the results of two models using these features: support vector regression (SVR) and Bayesian linear ridge regression (BLRR).

[29] examined several neural networks models, e.g. RNN and Convolutional Neural Networks (CNN), on ASAP dataset. In their experiments, Long Short Term Memory networks (LSTM) [36], a variant of RNN, achieved the best performance. LSTM is designed to have three gates in each hidden node: input gate, forget gate, and output gate. By controlling these three gates, LSMT has the capability of attaining long-term dependencies. The structure of the LSTM model described in [10] is presented in Figure 2.

To verify the efficacy of GloVe word embeddings and external memory, we developed a simple multi-layer forward neural networks (FNN) model, which is similar to our model with respect to the model structure, but without an external memory. We refer this baseline model as FNN for the rest of paper for convenience. As shown in Figure 3, each word of a student response is first converted to a continuous vector using GloVe word embeddings. The vector representation for the response is obtained by applying PE on all word vectors from the response. Afterward the representation is fed into 4 hidden layers, each of which has 100 hidden nodes. Apply a softmax operation on the resulting states of last hidden layer at output layer to predict the final score. The model is also trained using Adam Optimizer by minimising the standard cross entropy between $\hat{s}$ and truth score s. FNN is properly defined by the

Set	Grade level	# Essays	Avg len	Max len	Min score	Max score	Mean score
1	8	1,783	350	911	2	12	8
2	10	1,800	350	118	1	6	3
3	10	1,726	150	395	0	3	1
4	10	1,772	150	383	0	3	1
5	8	1,805	150	452	0	4	2
6	10	1,800	150	489	0	4	2
7	7	1,569	250	659	0	30	16
8	10	723	650	983	0	60	36

Table 1. Selected Details of ASAP dataset

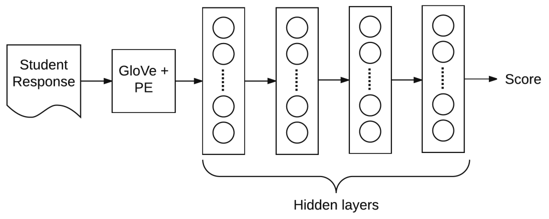

Figure 3. An illustration of baseline FNN. Use GloVe with PE to represent a student response. The representation is fed into 4-layer networks and each layer has 100 hidden nodes.

equations below:

$$h_0 = Relu(A^T x) \tag{7}$$

$$h_i = Relu(R_i h_{i-1}),\ for\ i \geq 1 \tag{8}$$

$$\hat{s} = Softmax(h_H W) \tag{9}$$

where x is the representation generated by GloVe with PE for a student response. h_i is the output of hidden layer i. H is the total number of hidden layers. A, R_i ,and W are weight matrices. The bias vectors are omitted in the equations.

RESULTS

In this section, we describe the results of our experiments on ASAP dataset and compare these results with baselines mentioned above. Column MN of Table 2 presents the QWK scores of our model. Column EASE (SVR) and column EASE(BLRR) contain the results from EASE with two different regression methods. We also compare our model to other neural models in [29] and the best results from [29] is listed in column LSTM+CNN of Table 2. Note that their best results reported in the paper are obtained by ensembling results from 10 runs of LSTM and 10 runs of CNN. However, in our experiment, the results are recorded from a single run of a single model after optimizing the hyperparameters. This is not fair to compare their best results with ours directly. Therefore we also pick the best performance achieved by a single model from their paper and list in Column LSTM of Table 2. In their setup, the number of hidden nodes in LSTM is 300 and pre-trained word embeddings released by [34] is used.

As indicated in Table 2, our model outperforms in 7 out of 8 sets (except for set 7) and improves the average QWK score by 4.0% compared to the baseline LSTM. Even compared to their best ensembled model (LSTM+CNN), our model still achieved better performance in 7 essay sets (except for essay 7). As expected, our model surpasses EASE in all 8 sets and improves average QWK score by 10%.

The results from the FNN model mentioned above is presented in column FNN of Table 2. In our experiments, FNN has 4 hidden layers and each layer has 100 hidden nodes, whose structure is similar to that of our model except that the external memory is removed. When comparing these results to the best results from EASE, we find that this basic model outperforms EASE in 7 out of 8 sets of essays (except for essay set 1) and is even comparable with the complex model (LSTM). This proves that using Glove word embeddings with PE to represent a student response is able to capture important features useful for grading the response. The effectiveness of the external memory is proved by the fact that MN accomplishes better performance on 7 sets (set 4 is equal) than FNN does. The comparison between FNN and other models indicates that representing a student response using GloVe with PE and adding external memory are two key factors which may lead to the good performance on ASAP dataset.

In ASAP dataset, two human graders are assigned to each student response and each grader gives a score separately. The final gold-standard score for each response is calculated based on these two scores. In Column Human of Table 2, we calculated the QWK scores between these two graders to measure the agreement between two graders.

Time Cost

To study the time efficiency of our model, we recorded the average training time for each epoch on three models: FNN, MN, and LSTM. The number of hidden nodes in LSTM is 100, which is the the same number of hidden nodes in FNN and MN. However, in practice the number is usually set to a larger number, e.g. 300 or 500. All these models are implemented in Tensorflow, trained with same hyperparameters and run on the same GPU (GTX 1080) server. When calculating the average time cost, we do not include the time for loading GloVe word embeddings and ASAP dataset, and the time for building model before the data is fed into it.

Table 3 presents the average training time of each epoch on 8 essay sets. It is clear that the LSTM model is computationally expensive and requires more computational resource. The complex calculation in each hidden node and long sequence of the input slows down the training process. On the other

Set	MN	FNN	EASE(SVR)	EASE(BLRR)	LSTM	LSTM+CNN	Human
1	0.83	0.75	0.78	0.76	0.78	0.82	0.72
2	0.72	0.7	0.62	0.61	0.69	0.69	0.81
3	0.72	0.7	0.63	0.62	0.68	0.69	0.77
4	0.82	0.8	0.75	0.74	0.8	0.81	0.85
5	0.83	0.8	0.78	0.78	0.82	0.81	0.75
6	0.83	0.79	0.77	0.78	0.81	0.82	0.78
7	0.79	0.73	0.73	0.73	0.81	0.81	0.72
8	0.68	0.63	0.53	0.62	0.59	0.64	0.63
Avg	0.78	0.74	0.7	0.71	0.75	0.76	0.75

Table 2. QWK scores on ASAP dataset.

Set	FNN	MN	LSTM
1	0.2	1.1	15.5
2	0.2	1	19.5
3	0.2	1	7
4	0.1	1	7
5	0.2	1	8
6	0.2	1	8.1
7	0.2	1.5	10
8	0.1	1.4	6.5
Avg	0.2	1.1	10.2

Table 3. Average runtime (seconds) of each training epoch

hand, MN is 9 times faster than LSTM since the computation of GloVe with PE is a simple element-wise sum and MN is insensitive to the length of a response. FNN is the fastest since the structure of FNN is the simplest. Unlike MN, FNN does not need to loop through each memory piece to measure the relevance of two student responses at training time.

DISCUSSION AND CONCLUSION

In this study, we develop a generic model for automated grading tasks using memory networks and word embeddings. To our best knowledge this is the first study that memory networks are applied for this kind of task. Our model is tested on ASAP dataset and achieves state-of-the-art performance in 7 out of 8 essay sets. Similar to other neural networks models for AES, our model can be trained in an end-to-end fashion and does not require any hand-engineered features. Compared to RNN, CNN, using GloVe word embeddings with PE to represent a student response makes our model simple and cost-effective. Adding external memory improves the performance over FNN model, which means our model is able to take advantage of sampled responses stored in the external memory.

Our model can be generalized to automatically grade assignments from other subjects. As shown above, there are two key factors to the performance: reliable representation and memory component. In order to apply our model to other kinds of assignment, learning a good vector representation for the assignment is the first step. It is analogous to how the regression model is built for supervised NLP tasks: first extract numerical hand-engineered features from text and then apply a regression model on these generated features to predict true labels. In the context of neural networks, a vector is required to represent the student response. Learning the vector can be a part of the predictive model. For example, the word embeddings in [10] are learned from their predictive model. These vectors can also come from pre-trained models, like GloVe and word2vec. The next step is to select characterized samples and store these samples to memory. The purpose of this step is to teach the model to understand the grading strategy and eventually associate a vector representation to a score.

However, we only test our model on one dataset. There is a need to explore our model with more datasets that contain various formats of assignments to verify our model. Furthermore, the representation of the assignment and the mechanism for measuring relevance among assignments is still elementary. Future work should therefore focus on these two areas to improve the generalizability of the model. A lot of effort is still needed to better interpret memory networks and explain the key factors behind our performance improvement.

ACKNOWLEDGMENTS

We acknowledge funding from multiple NSF grants (ACI-1440753, DRL-1252297, DRL-1109483, DRL-1316736 & DRL-1031398), the U.S. Department of Education (IES R305A120125 & R305C100024 and GAANN), the ONR, and the Gates Foundation.

REFERENCES

1. Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. (1 Sept. 2014).

2. S P Balfour. 2013. Assessing writing in MOOCs: Automated essay scoring and calibrated peer review. In *Research and Practice in Assessment 8*, Vol. 1. 40–48.

3. Michael Brooks, Sumit Basu, Charles Jacobs, and Lucy Vanderwende. 2014. Divide and correct: using clusters to grade short answers at scale. In *L@S*.

4. Hongbo Chen and Ben He. 2013. Automated Essay Scoring by Maximizing Human-Machine Agreement. In *EMNLP*.

5. Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. (3 June 2014).

6. Cícero Nogueira dos Santos and Maira Gatti. 2014. Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts. In *COLING*. 69–78.

7. Rehab Duwairi. 2006. A framework for the computerized assessment of university student essays. *Comput. Human Behav.* 22 (2006), 381–388.

8. G E Forsythe and N Wirth. 1965. Automatic Grading Programs. Commun. *Commun. ACM 8* 5 (1965), 275–278.

9. Chase Geigle, Chengxiang Zhai, and Duncan C Ferguson. 2016. An Exploration of Automated Grading of Complex Assignments. In *L@S*.

10. Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural Turing Machines. (20 Oct. 2014).

11. Michael T Helmick. 2007. Interface-based programming assignments and automatic grading of java programs. In *ITiCSE*.

12. Diederik P Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2014).

13. Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask Me Anything: Dynamic Memory Networks for Natural Language Processing. In *ICML*.

14. Leah S Larkey. 1998. Automatic Essay Grading Using Text Categorization Techniques. In *SIGIR*.

15. Claudia Leacock and Martin Chodorow. 2003. C-rater: Automated Scoring of Short-Answer Questions. *Comput. Hum.* 37 (2003), 389–405.

16. Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. (2015).

17. Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26*, C J C Burges, L Bottou, M Welling, Z Ghahramani, and K Q Weinberger (Eds.). Curran Associates, Inc., 3111–3119.

18. Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-Value Memory Networks for Directly Reading Documents. *CoRR* abs/1606.03126 (2016).

19. Tom Mitchell, Terry Russell, Peter Broomhead, and Nicola Aldridge. 2002. Towards Robust Computerised Marking of Free-text Responses towards Robust Computerised Marking of Free-text Responses.

20. P Mitros, V Paruchuri, J Rogosic, and others. 2013. An integrated framework for the grading of freeform responses. *The Sixth Conference of* (2013).

21. Michael Mohler and Rada Mihalcea. 2009. Text-to-Text Semantic Similarity for Automatic Short Answer Grading. In *EACL*.

22. Andy Nguyen, Chris Piech, Jonathan Huang, and Leonidas J Guibas. 2014. Codewebs: scalable homework search for massive open online programming courses. In *WWW*.

23. Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global Vectors for Word Representation. In *EMNLP*, Vol. 14. 1532–1543.

24. Peter Phandi, Kian Ming Adam Chai, and Hwee Tou Ng. 2015. Flexible Domain Adaptation for Automated Essay Scoring Using Correlated Linear Regression. In *EMNLP*.

25. Chris Piech, Jonathan Huang, Andy Nguyen, Mike Phulsuksombati, Mehran Sahami, and Leonidas J Guibas. 2015. Learning Program Embeddings to Propagate Feedback on Student Code. *CoRR* abs/1505.05969 (2015).

26. Carolyn Penstein Rosé, Antonio Roque, Dumisizwe Bhembe, and Kurt VanLehn. 2003. A Hybrid Text Classification Approach For Analysis Of Student Essays.

27. Lawrence M Rudner and Tahung Liang. 2002. Automated Essay Scoring Using Bayes' Theorem. *The Journal of Technology, Learning and Assessment* 1, 2 (1 June 2002).

28. Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-To-End Memory Networks. In *Advances in Neural Information Processing Systems 28*, C Cortes, N D Lawrence, D D Lee, M Sugiyama, and R Garnett (Eds.). Curran Associates, Inc., 2440–2448.

29. Kaveh Taghipour and Hwee Tou Ng. 2016. A Neural Approach to Automated Essay Scoring. In *EMNLP*.

30. Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2015. Towards AI-Complete Question Answering: A Set of Prerequisite Toy Tasks. (19 Feb. 2015).

31. Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory Networks. *CoRR* abs/1410.3916 (2014).

32. Caiming Xiong, Stephen Merity, and Richard Socher. 2016. Dynamic Memory Networks for Visual and Textual Question Answering. (4 March 2016).

33. Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A New Dataset and Method for Automatically Grading ESOL Texts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1 (HLT '11)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 180–189.

34. Will Y Zou, Richard Socher, Daniel M Cer, and Christopher D Manning. 2013. Bilingual Word Embeddings for Phrase-Based Machine Translation. In *EMNLP*.