MANTECH Publications

Journal of Computational Intelligence Theory and Its Applications
Volume 2 Issue 1

Automated Essay Scoring using Neural Network

Steve Thomas[1], Steephen Joseph[2], Basil Sunny[3], Surekha Mariam Varghese[4], Aby Abahai T[5]

Department of Computer Science and Engineering

Mar Athanasius College of Engineering, Kothamangalam

Corresponding Authors: *stevethomask@gmail.com[1], steephenj6@gmail.com[2], basilsunny4@gmail.com[3]*

surekha@mace.ac.in[4], abytom@gmail.com[5]

Abstract

Traditional automated essay scoring systems rely on carefully designed features to evaluate and score essays. The performance of such systems is tightly bound to the quality of the underlying features. However, it is difficult to manually design the most informative features for such a system. In this paper, we develop an approach based on recurrent neural networks to learn the relation between an essay and its assigned score, without any feature engineering. We use Long Short-Term Memory neural network model for the task of automated essay scoring and perform some analysis to get some insights of the model. The results show that our system, which is based on long short-term memory networks, outperforms a strong baseline by 5.6% in terms of quadratic weighted Kappa, without requiring any feature engineering.

Keywords: *Neural Network, Automated essay scoring systems, Long Short-Term Memory neural network.*

I. INTRODUCTION

There is a recent surge of interest in neural networks, which are based on continuous-space representation of the input and non-linear functions. Hence, neural networks are capable of modeling complex patterns in data. Moreover, since these methods do not depend on manual engineering of features, they can be applied to solve problems in an end-to-end fashion. SENNA and neural machine translation are two notable examples in natural language processing that operate without any external task-specific knowledge. In this paper, we report

a system based on neural networks to take advantage of their modeling capacity and generalization power for the automated essay scoring (AES) task. Essay writing is usually a part of the student assessment process. Several organizations, such as Educational Testing Service (ETS), evaluate the writing skills of students in their examinations. Because of the large number of students participating in these exams, grading all essays is very time consuming. Thus some organizations have been using AES systems to reduce the time and cost of scoring essays.

Automated essay scoring refers to the process of grading student essays without human interference. An AES system takes as input an essay written for a given prompt, and then assigns a numeric score to the essay reflecting its quality, based on its content, grammar, and organization. Such AES systems are usually based on regression methods applied to a set of carefully designed features. The process of feature engineering is the most difficult part of building AES systems. Moreover, it is challenging for humans to consider all the factors that are involved in assigning a score to an essay. Our AES system, on the other hand, learns the features and relation between an essay and its score automatically. Since the system is based on recurrent neural networks, it can effectively encode the information required for essay evaluation and learn the complex patterns in the data through non-linear neural layers.

The rest of this paper is organized as follows. Section 2 gives an overview of related work in the literature. Section 3 describes the automated essay scoring task and the evaluation metric used in this paper. We provide the details of our approach in Section 4, and present and discuss the results of our experimental evaluation in Section 5. Finally, we conclude the paper in Section 6.

II. RELATED WORK

There exist many automated essay scoring systems (Shermis and Burstein, 2013) and some of them are being used in high-stakes assessments. e-rater (Attali and Burstein, 2004) and Intelligent Essay Assessor (Foltz et al., 1999) are two notable examples of AES systems. In 2012, a competition on automated essay scoring called ‘Automated Student Assessment Prize’ (ASAP) was organized by Kaggle and sponsored by the Hewlett Foundation. A comprehensive comparison of AES systems was made in

the ASAP competition. Although many AES systems have been developed to date, they have been built with hand-crafted features and supervised machine learning algorithms.

Researchers have devoted a substantial amount of effort to design effective features for automated essay scoring. These features can be as simple as essay length (Chen and He, 2013) or more complicated such as lexical complexity, grammaticality of a text (Attali and Burstein, 2004), or syntactic features (Chen and He, 2013). Readability features (Zesch et al., 2015) have also been proposed in the literature as another source of information.

Moreover, text coherence has also been exploited to assess the flow of information and argumentation of an essay (Chen and He, 2013). A detailed overview of the features used in AES systems can be found in (Zesch et al., 2015). Moreover, some attempts have been made to address different aspects of essay writing independently. For example, argument strength and organization of essays have been tackled by some researchers through designing taskspecific features for each aspect (Persing et al., 2010; Persing and Ng, 2015).

Our system, however, accepts an essay text as input directly and learns the features automatically from the data. To do so, we have developed a method based on recurrent neural networks to score the essays in an end-to-end manner. Our model is a long short-term memory neural network (Hochreiter and Schmidhuber, 1997) and is trained as a regression method. Similar recurrent neural network approaches have recently been used successfully in a number of other NLP tasks. For example, Bahdanau et al. (2015) have proposed an attentive neural approach to machine translation based on gated recurrent units (Cho et al., 2014).

Neural approaches have also been used for syntactic parsing. In (Vinyals et al., 2015), long short-term memory networks have been used to obtain parse trees by using a sequence-to-sequence model and formulating the parsing task as a sequence generation problem. Apart from these examples, recurrent neural networks have also been used for opinion mining (Irsoy and Cardie, 2014), sequence labeling (Ma

and Hovy, 2016), language modeling (Kim et al., 2016; Sundermeyer et al., 2015), etc.

III. AUTOMATED ESSAY SCORING

In this section, we define the automated essay scoring task and the evaluation metric used for assessing the quality of AES systems.

A. Task Description

Automated essay scoring systems are used in evaluating and scoring student essays written based on a given prompt. The performance of these systems is assessed by comparing their scores assigned to a set of essays to human-assigned gold-standard scores. Since the output of AES systems is usually a real-valued number, the task is often addressed as a supervised machine learning task (mostly by regression or preference ranking). Machine learning algorithms are used to learn the relationship between the essays and reference scores.

B. Evaluation Metric

The output of an AES system can be compared to the ratings assigned by human annotators using various measures of correlation or agreement (Yan-nakoudakis and Cummins, 2015). These measures include Pearson's correlation, Spearman's correlation, Kendall's Tau, and quadratic weighted Kappa (QWK). The ASAP competition adopted QWK as the official evaluation metric. Since we use the ASAP data set for evaluation in this paper, we also use QWK as the evaluation metric in our experiments.

In our experiments, we compare the QWK score of our system to well-established baselines. We also perform a one tailed paired t-test to determine whether the obtained improvement is statistically significant.

IV. A RECURRENT NEURAL NETWORK APPROACH

Recurrent neural networks are one of the most successful machine learning models and have attracted the attention of researchers from various fields. Compared to feed-forward neural networks, recurrent neural networks are theoretically more powerful and are capable of learning more complex patterns from data. Therefore, we have mainly focused on recurrent networks in this paper. This section gives a description of the recurrent neural network architecture that we have used for the essay scoring task and the training process.

A. Model Architecture

Lookup Table Layer: The first layer of our neural network projects each word into a dLT dimensional space. Given a sequence of words W represented by their one-hot representations (w_1, w_2, · · , w_M), the output of the lookup table layer is calculated by Equation 3:

$$LT(W) = (E.w_1, E.w_2, \cdots, E.w_M) \quad (3)$$

where E is the word embeddings matrix and will be learnt during training.

Convolution Layer: Once the dense representation of the input sequence W is calculated, it is fed into the recurrent layer of the network. How-ever, it might be beneficial for the network to ex-tract local features from the sequence before applying the recurrent operation. This optional characteristic can be achieved by applying a convolution layer on the output of the lookup table layer. In order to extract local features from the sequence, the convolution layer applies a linear transformation to all M windows in the given sequence of vectors 4 . Given a window of dense word representations x1, x2, · · · , xl, the convolution layer first concatenates these vectors to form a vector $\bar{x}$ of length l.dLT and then uses Equation 4 to calculate the out-put vector of length dc:

$$Conv(\bar{x}) = W.\bar{x} + b \quad (4)$$

In Equation 4, W and b are the parameters of the network and are shared across all windows in the sequence.

The convolution layer can be seen as a function that extracts feature vectors from n-grams. Since this layer provides n-gram level information to the subsequent layers of the neural network, it can potentially capture local contextual dependencies in the essay and consequently improve the performance of the system.

Recurrent Layer: After generating embeddings (whether from the convolution layer or directly from the lookup table layer), the recurrent layer starts processing the input to generate a representation for the given essay. This representation should ideally en-code all the information required for grading the essay. However, since the essays are usually long, consisting of hundreds of words, the learnt vector representation might not be sufficient for accurate scoring. For this reason, we preserve all the inter-mediate states of the

recurrent layer to keep track of the important bits of information from processing the essay. We experimented with basic recur-rent units (RNN) (Elman, 1990), gated recurrent units (GRU) (Cho et al., 2014), and long short-term memory units (LSTM) (Hochreiter and Schmidhu-ber, 1997) to identify the best choice for our task. LSTM outperforms a strong baseline by 5.6% in terms of QWK.

Long short-term memory units are modified re-current units that can cope with the problem of vanishing gradients more effectively (Pascanu et al., 2013). LSTMs can learn to preserve or forget the information required for the final representation. In order to control the flow of information during processing of the input sequence, LSTM units make use of three gates to discard (forget) or pass the information through time. The following equations formally describe the LSTM function:

$$
\begin{aligned}
i_t &= \sigma(W_i.x_t + U_i.h_{t-1} + b_i) \\
f_t &= \sigma(W_f.x_t + U_f.h_{t-1} + b_f) \\
\tilde{c}_t &= \tanh(W_c.x_t + U_c.h_{t-1} + b_c) \qquad (5) \\
c_t &= i_t \circ \tilde{c}_t + f_t \circ c_{t-1} \\
o_t &= \sigma(W_o.x_t + U_o.h_{t-1} + b_o) \\
h_t &= o_t \circ \tanh(c_t)
\end{aligned}
$$

where x_t and ht are the input and output vectors at time t, respectively. W_i, W_f , W_c, W_o, U_i, U_f , U_c, and U_o are weight matrices and bi, bf , b_c, and b_o are bias vectors. The symbol ◦ denotes element-wise multiplication and σ represents the sigmoid function.

Mean over Time: The outputs of the recurrent layer, H = (h_1, h_2, · · · , h_M), are fed into the mean-over-time layer. This layer receives M vectors of length dr as input and calculates an average vector of the same length. The mean-over-time layer is responsible for aggregating the variable number of inputs into a fixed length vector. Once this vector is calculated, it is fed into the linear layer to be mapped into a score.

Linear Layer with Sigmoid Activation: The linear layer maps its input vector generated by the mean-over-time layer to a scalar value. This map-ping is simply a linear transformation of the in-put vector and therefore, the computed value is not bounded. Since we need a bounded value in the range of valid scores for each prompt, we apply a sigmoid function to limit the possible scores to the range of (0, 1). The mapping of the linear layer after applying

the sigmoid activation function is given by Equation 7:

$$S(x) = \text{sigmoid}(w.x + b) \quad (7)$$

where x is the input vector (M oT (H)), w is the weight vector, and b is the bias value.

We normalize all gold-standard scores to [0, 1] and use them to train the network. However, during testing, we rescale the output of the network to the original score range and use the rescaled scores to evaluate the system.

B. Training

We use the RMSProp optimization algorithm (Dauphin et al., 2015) to minimize the mean squared error (MSE) loss function over the training data. Given N training essays and their corresponding normalized gold-standard scores s∗ i, the model computes the predicted scores si for all training essays and then updates the network parameters such that the mean squared error is minimized. Additionally we make use of dropout regularization to avoid overfitting. We also clip the gradient if the norm of the gradient is greater than a threshold.

We do not use any early stopping methods. In-stead, we train the neural network model for a fixed number of epochs and monitor the performance of the model on the development set after each epoch. Once training is finished, we select the model with the best QWK score on the development set.

V. EXPERIMENTS

In this section, we describe our experimental setup and present the results. Moreover, an analysis of the results and some discussion are provided in this section.

A. Setup

The dataset that we have used in our experiments is the same dataset used in the ASAP competition run by Kaggle (see Table 1 for some statistics). We use quadratic weighted Kappa as the evaluation metric, following the ASAP competition. Since the test set used in the competition is not publicly available, we use 5-fold cross validation to evaluate our systems. In each fold, 60% of the data is used as our training set, 20% as the development set, and 20% as the test set. We train the model for a fixed number of epochs and then choose the best model based on the development set. We tokenize the essays using the NLTK

tokenizer, lowercase the text, and normalize the gold-standard scores to the range of [0, 1]. During testing, we rescale the system-generated normalized scores to the original range of scores and mea-sure the performance.

Our system has several hyper-parameters that need to be set. We use the RMSProp optimizer with decay rate (ρ) set to 0.9 to train the network and we set the base learning rate to 0.001. The mini-batch size is 32 in our experiments and we train the net-work for 50 epochs. The vocabulary is the 4,000 most frequent words in the training data and all other words are mapped to a special token that represents unknown words. We regularize the network by using dropout (Srivastava et al., 2014) and we set the dropout probability to 0.5.

During training, the norm of the gradient is clipped to a maximum value of 10. We set the word embedding dimension (dLT) to 50 and the output dimension of the recurrent layer (dr) to 300. If a convolution layer is used, the window size (l) is set to 3 and the output dimension of this layer (dc) is set to 50. Finally, we initialize the lookup table layer using pre-trained word embeddings released by Zou et al. (2013). Moreover, the bias value of the linear layer is initialized such that the network's output before training is almost equal to the average score in the training data.

CONCLUSION

In this paper, we have proposed an approach based on recurrent neural networks to tackle the task of automated essay scoring. Our method does not rely on any feature engineering and automatically learns the representations required for the task. We have explored a variety of neural network model architectures for automated essay scoring and have achieved significant improvements over a strong open source baseline.

Our best system outperforms the baseline by 5.6% in terms of quadratic weighted Kappa. Furthermore, an analysis of the network has been per-formed to get an insight of the recurrent neural net-work model and we show that the method effectively utilizes essay content to extract the required information for scoring essays.

ACKNOWLEDGEMENT

God, we thank you for all the blessings bestowed upon us during the course of this project. The entire department especially

our HOD Dr. Surekha Mariam Varghese, group tutor Prof. Eldo P Elias and our guide Prof. Aby Abahai was very supportive and helped with all our doubts. We thank all of them for their help and support.

REFERENCES

1) Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase represen-tations using RNN encoder–decoder for statistical ma-chine translation. In Proceedings of the 2014 Confer-ence on Empirical Methods in Natural Language Pro-cessing.

2) Yigal Attali and Jill Burstein. 2004. Automated essay scoring with e-rater R v. 2.0. Technical report, Educa-tional Testing Service.

3) Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Ben-gio. 2015. Neural machine translation by jointly learning to align and translate. In Proceedings of the 3rd International Conference on Learning Represen-tations.

4) Ozan Irsoy and Claire Cardie. 2014. Opinion mining with deep recurrent neural networks. In Proceedings of the 2014 Conference on Empirical Methods in Nat-ural Language Processing.

5) Dimitrios Alikaniotis, Helen Yannakoudakis, and Marek Rei. 2016. Automatic text scoring using neural net-works. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics.