AUTOMATED ESSAY GRADING FROM SCRATCH

WON I. LEE, QIN LYU, AND ZELONG QIU

Abstract. Automated essay grading is an important problem for advancing curricular assessments in K-12 education, as it allows for rapid evaluation of student-constructed responses and opens the way for the assessment of more sophisticated analytical and reasoning skills. In this work, we investigate the performance of various deep architectures on the task of automated essay grading. We employ no hand-crafted features and explore both character-level and word-level models for this task, using both convolutional and recurrent architectures. We find that ... . In addition, we provide a more nuanced analysis of the strengths of various architectural and training choices by considering performance across a variety of metrics, showing that ... .

Introduction and Related Work

One prominent impediment preventing the enhancement of curricula in K-12 education to focus more on critical reasoning ability and analytical skills is the difficulty of scoring tests to measure these abilities. It is generally the case that measuring such skills requires inviting the student for open-ended answers rather than simple, multiple-choice ones; unfortunately, tests that require essays or other individually-written responses are difficult to score automatically and necessitate scoring by human graders. Such grading involves substantial time and expense from both testing companies and government agencies (for standardized exams), and as a result, most such tests have relied more on multiple-choice questions, limiting the ability to assess more sophisticated skills.

This has long motivated the development of automated system for essay grading. Much effort has been put into building a satisfactory system that could reliable generate scores in line with those of human judges. The Intelligent Essay Assessor (IEA) was first used to score essays for large undergraduate courses in 1994 [1], and the e-Rater, an automated reader developed by the Educational Testing Service was used to grade large-scale essays on standardized exams [2].

All of these examples used on the order of hundreds of hand-crafted features to generate the essay scores. Recently, however, the development of more automatic methods in natural language processing based on deep neural networks has suggested that systems for such tasks boasting comparable or even superior performance can be constructed "from scratch" without manually crafting features for the algorithm. For example, the work of Collobert et al. [3], which helped to inspire the current wave of deep learning applications in NLP, demonstrated that appropriately-trained neural network models incorporating no hand-crafted features could achieve performance rivaling that of state-of-the-art systems based on extensive feature crafting.

More recently, a variety of deep architectures have emerged for various tasks in NLP. In particular, the sequence-to-sequence model, incorporating two LSTMs (encoder and decoder), has emerged as the paradigm for sequential tasks such as machine translation [**?**]. Certain architectures often consider character-level inputs instead of or in addition to word-level inputs, often employing convolutional neural networks (CNNs) as a representation-generating layer before feeding into a fully-connected model or recurrent network [**?**].

All of these networks offer important features and benefits that are suitable for certain tasks. For the task of automated essay grading, however, we note that many of the advantages offered by each respective model are desirable. For example, as described below, it would be beneficial to employ character-level inputs for essay grading; in another vein, the sequential information captured by recurrent networks are often vital to judging the quality of essays. Thus, in this work, we report on a series of experiments testing which model architectures work best for the task of automated essay grading. Of course, we cannot be exhaustive in this search, but we aim to provide an illustrative and meaningful look at the advantages of certain architectures and training methods over others.

Essay Set	Score Range	Train Set Size	Valid Set Size	Test Set Size
1	2-12	1426	179	178
2	1-6	1416	193	191
3	0-3	1384	166	176
4	0-3	1407	183	180
5	0-4	1458	178	169
6	0-4	1458	157	185
7	2-24	1257	151	161
8	10-60	612	52	59
Total	0-60	10419	1260	1300

Table 1. Statistics for the data set of essays and scores.

Data

Our dataset consists of essays selected from 8 different essay prompts across a variety of exams. These essay sets differ in the length of essays with which they are associated, their possible score ranges, and the number of examples, among other qualities. We provide an illustration of the statistics associated with each essay set in Table 1, demonstrating the large variability in the score ranges and number of examples.

The dataset contains both the essay set ID and essay ID, as well as the entire text of the essay. For evaluation, the dataset contains one or more human scores, and the final resolved human score (generally an average of scores if there are multiple available for a given essay). The final resolved score is the measure of interest for this task, and in our experiments we only consider this measure as the quantity of interest.

Models

Character-Level CNN. A prominent issue with essay grading is that many of the words in candidate essays are misspelled, so that they lie outside a fixed vocabulary. Most models with fixed vocabulary word embeddings can only deal with such words by treating them as "UNK" tokens, which removes any information that such words may convey about the quality of the essay. Unlike most natural language tasks for which out-of-vocabulary words are infrequent and often unimportant, misspellings can provide valuable information regarding the quality of the writing for grading student essays.

One way to deal with such cases is to consider character-level models rather than word-level models. Thus, we consider a character-level convolutional neural network (CNN), which embeds each character into a d-dimensional vector ($d \approx 30$) and employs a series of 1D convolutions and max-pooling layers to extract efficient representations for inference. We stack fully-connected layers on top of the convolutional layers for final processing, and use either dense or one-hot encodings of the characters as input. We provide a schematic of our overall architecture in Figure 1. We consider a 6 convolutional layer model inspired by the work of Zhang, Zhao, and LeCun, 2015 [**?**] as well as a simpler, 1 convolutional layer model inspired by the work of Kim, 2014 [**?**].

Training. In this formulation, we primarily treated the essay score as a percentage of the score range (i.e. for essay set 1, which has range 2-12, an essay with score 6 would be treated as 0.4), and used logistic regression as our final output. We used standard stochastic gradient descent (SGD) with momentum for training, using binary cross-entropy loss as our objective.

Word-Level LSTM. An issue encountered in character-level models on this task, however, is that the models are unable to leverage semantic relationships between the words in order to make better predictions regarding essay quality.

Sentence-Level LSTM and Pairwise Ranking Model. In addition to character-level and word-level models, we also consider sentence-level model for essay grading. Suppose each sentence is encoded into a vector of fixed dimension, then essay can be seen as a series of sentence vectors. Hence it is natural to use LSTM model to capture sentence-level transition patterns in each essay and measure the quality of essays by assessing such patterns. There are two ways to encode a sentence into a vector:

(1) Bag-of-Words model: The sentence vector is the sum of word vectors of the words in the sentence.

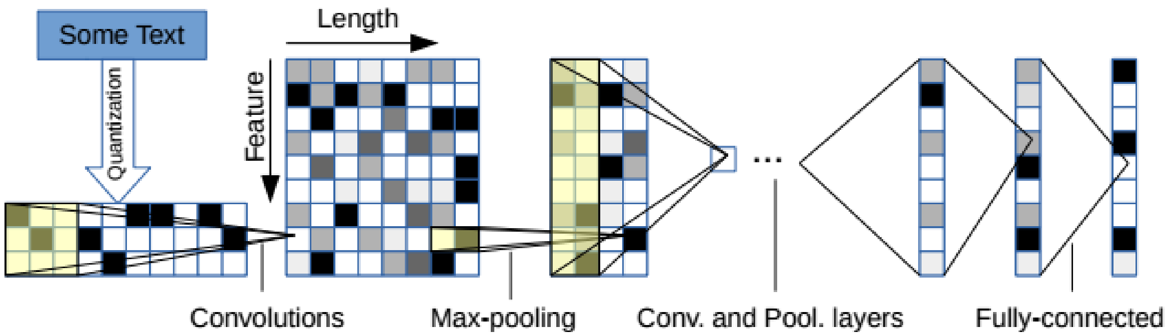

Figure 1. Architecture of character-level CNN model; figure credit to Zhang, X., J. Zhao, and Y. LeCun (2015).

(2) Sequence model: The sentence vector is the output of another LSTM model, whose input is the word vector sequence of the words in the sentence.

To reduce computational complexity, we use bag-of-words model to encode each sentence into vector, where the pre-trained word vectors are provided by GloVe [4].

One issue encountered in classification model and regression model in this task is that the score range is different across sets. In previous section, we address this issue by normalizing the score to real values in $[0, 1]$. However, essays from different set may not be comparable as the assessment standard is different from one set to another. Either classification model or regression model is trained with essays from all 8 sets with abosulte scores, which implicitly introduces incorrect order information. Hence we consider to use learning-to-rank model to learn the score order instead of learning the scores directly.

We make pairs of essays from the same set. For each pair (d_l, d_r), we assign label 0 if $score_l > score_r$ and label 1 if $score_l < score_r$. In other words, labels represent the order information of essay pairs. Denote by o_i the output score of sentence-level LSTM model with input essay d_i. We use the cross entropy loss function

$$L_{ij} \equiv L(o_{ij}) = -y_{ij} \log P_{ij} - (1 - y_{ij}) \log(1 - P_{ij})$$

where $o_{ij} \equiv o_i - o_j, P_{ij} \equiv 1/(1 + e^{-o_{ij}})$, and y_{ij} is the label of (d_i, d_j). In fact, this is the pairwise loss function proposed by Burges et al., 2005 [5].

The model architecture is shown in Figure 2. Note that we add one more LSTM layer with reverse direction to further exploit the trainsition pattern in essays.

Another issue is the number of essay pairs used in training. Note that the training set will grow exponentially large if we make all the pairs in each essay set. Since each essay set in our problem contains more than one thousand essays, it is computationally infeasible to use all the essay pairs as training data. To keep the order information in original dataset as much as possible, we sort the essays in each essay set by their scores, and pair the top K essays with the other essays in the same essay set. In other words, we keep most of the information for determining what is a good essay while no essay is discarded. Note that this also reduces the number of essay pairs in which the two essays have similar scores, which is a potential way to reduce noise as all essays were graded by human and it is a hazard to determine the order of essays with similar scores.

Metrics

For the purposes of investigating the respective strengths of the various models considered above, we also consider a number of different evaluation metrics, which highlight different performance properties. We note that these are *not* the metrics used during training, but rather metrics used to evaluate the performance of the models on the test set. For all formulas, we let p_i to be the predicted score on the essay i and y_i to be the actual score, with a total test size of N.

Classification Accuracy. As the essay grading problem is often treated as a classification problem (each discrete score being a "category" of essays), we consider evaluating on the simplest classification metric,

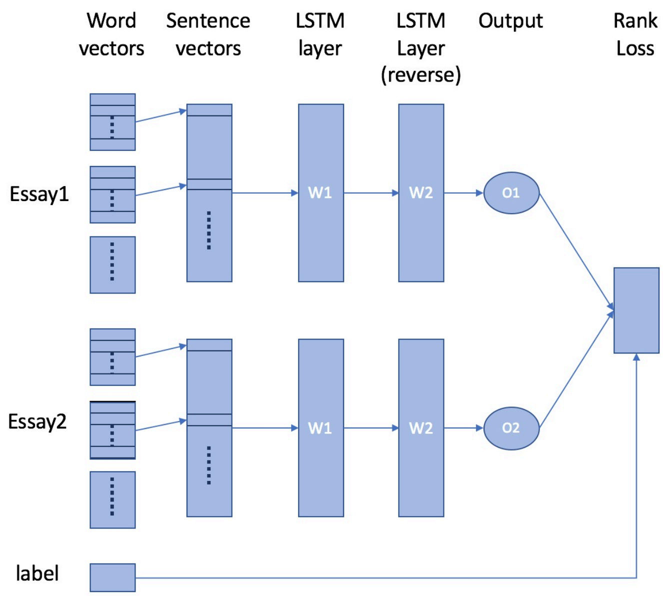

FIGURE 2. Architecture of sentence-level LSTM model with pairwise ranking loss. Note that the weights of the two LSTM models are shared.

namely that of accuracy on the test set:

$$ACC \equiv \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}_{p_i=y_i}$$

where $\mathbb{I}_{x=y}$ is an indicator of whether $x = y$.

Mean-Squared Error (MSE). One issue with a raw classification accuracy metric is that it does not provide any indication of the discrepancy between the actual and predicted scores. For example, on a scale of 2-12, a model predicting a score of 3 would incur the same loss as a model predicting 9 for an actual score of 10. We would clearly prefer the latter model over the former model, however; indeed, for grading purposes, it is not as important to achieve the exact same score so much as it is to obtain predictions that are on average close to the actual scores.

Thus, we consider the traditional mean-squared error (MSE) as one of our metrics, namely:

$$MSE \equiv \frac{1}{N}\sum_{i=1}^{N}(p_i - y_i)^2$$

Normalized Discounted Cumulative Gain (NDCG). In addition to the classification accuracy and MSE, we also report NDCG to measure the ranking quality of a model. In our problem, each essay has a discrete score that can be compared with the scores of other essays in the same essay set, while scores of essays from different essay sets are not comparable since the score range of essay sets are different. For each essay set, we compute the Discounted Cumulative Gain (DCG):

$$DCG = \sum_{i=1}^{n} \frac{score_i}{log_2(i+1)}$$

where $score_i$ is the score of essay that is ranked at the ith position among all essays from the same essay set by the model. The premise of DCG is that essay with high score but ranked lower by the model should be penalized, as the score is reduced logarithmically proportional to the rank given by the model.

In our problem, essay sets contain different number of essays, hence comparing a model's performance from one essay set to the other cannot be consistently achieved using DCG alone. To address this issue, we report the Normalized Discounted Cumulative Gain (NDCG) instead:

Model	Accuracy	MSE	NDCG	Quadratic Kappa
Char-level CNN (1-layer, one-hot)				
Char-level CNN (1-layer, embed)				
Char-level CNN (6-layer, one-hot)				
Char-level CNN (6-layer, embed)				
Sentence-level LSTM	–	–	0.9482	–

TABLE 2. Performance on test set evaluated via the metrics discussed above.

Model	Embed Dim.	Filter Sizes	Num. Filters	Pooling	Hidden Dim.
Char-level CNN (1-layer, one-hot)	–	64	64	64	32
Char-level CNN (1-layer, embed)					
Char-level CNN (6-layer, one-hot)	–				
Char-level CNN (6-layer, embed)					

TABLE 3. Hyperparameter and architectural choices as selected by performance on validation. Hyperparameter settings that are irrelevant for certain models are omitted (–).

$$NDCG = \frac{DCG}{IDCG}$$

where IDCG(Ideal DCG) is the maximum possible DCG obtained by sorting all essays in the set by their scores and then calculating the DCG.

Quadratic Kappa.

Results

Discussion

Member Contributions

Won Lee. Prior to the switch in our project from music-to-lyric generation to automated essay grading, I conducted all data generation (i.e. cleaning, formatting, compiling music-lyric pairs, etc.) and finalized models and training for training/evaluation on Harvard's clusters. With regard to essay grading, I developed all of the character-level and CNN-based models and conducted exhaustive grid search and training for these models. I also contributed to the evaluation script as well as various data-related auxiliary scripts for loading onto the models. Finally, I did most of the write-up for the current report as well as the proposal.

Qin Lyu. Prior to the switch, I designed the NN-based model and intensively explored its ability for music-lyric generation. With regard to essay grading, I developed all of the sentence-level LSTM models, and proposed to use the pairwise ranking loss to model the order information, and also proposed to use NDCG as an alternative measure of grading quality. I also contributed to the evaluation script (NDCG part) as well as data preprocessing script.

Zelong Qiu.

References

[1] Foltz, P. W., et al. (2013). "Implementation and applications of the Intelligent Essay Assessor." *Handbook of Automated Essay Evaluation*: 68-88.

[2] Attali, Y. and J. Burstein (2006). "Automated Essay Scoring with e-Rater V2." *The Journal of Technology, Learning and Assessment* 4.

[3] Collobert, R.

[4] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation.

[5] Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., and Hullender, G. (2005). Learning to rank using gradient descent. In Proceedings of the 22nd international conference on Machine learning (pp. 89-96). ACM.