A Study on Performance Sensitivity to Data Sparsity for Automated Essay Scoring

Yanhua Ran, Ben He(✉), and Jungang Xu

School of Computer and Control Engineering,
University of Chinese Academy of Sciences, Beijing 101408, China
ranyanhua16@mails.ucas.ac.cn, {benhe,xujg}@ucas.ac.cn

Abstract. Automated essay scoring (AES) attempts to rate essays automatically using machine learning and natural language processing techniques, hoping to dramatically reduce the manual efforts involved. Given a target prompt and a set of essays (for the target prompt) to rate, established AES algorithms are mostly prompt-dependent, thereby heavily relying on labeled essays for the particular target prompt as training data, making the availability and the completeness of the labeled essays essential for an AES model to perform. In aware of this, this paper sets out to investigate the impact of data sparsity on the effectiveness of several state-of-the-art AES models. Specifically, on the publicly available ASAP dataset, the effectiveness of different AES algorithms is compared relative to different levels of data completeness, which are simulated with random sampling. To this end, we show that the classical RankSVM and KNN models are more robust to the data sparsity, compared with the end-to-end deep neural network models, but the latter leads to better performance after being trained on sufficient data.

Keywords: Automated essay scoring · Data sparsity
Deep neural network

1 Introduction

Automated essay scoring (AES) is usually considered as a machine learning problem [3,7,20] where learning algorithms such as k-nearest neighbor (KNN) and support vector machines for ranking (RankSVM) are applied to learn a rating model for a given essay prompt, after being trained on a set of labeled essays rated by human assessors [5]. Currently, the AES systems have been widely used in large-scale English writing tests, e.g. Graduate Record Examination (GRE), to reduce the human efforts in the writing assessments.

Existing AES systems rely on handcrafted features which encode intuitive dimensions of semantics or writing quality, including lexical complexity, grammar errors, syntactic complexity, organization and development, and coherence etc. [21]. Such AES systems are mostly prompt-dependent in the sense that they can function well only on the essays for the prompt on which manually labeled essays are available for training. Such prompt-dependent natures sometimes make it hard to apply AES in reality especially when limited rated essays, if any, are available for training. For example, in a writing test, students are asked to write essays for a target prompt without any rated examples, where the prompt-dependent methods are unlikely to perform well due to the lack of training data. Actually, as suggested by a recent study on the task-independent features, the lack of prompt-specific training data is one of the major causes for the degrading performance of current AES methods [23], highlighting the importance of the completeness of training data for AES models. As far as our knowledge, however, it is still not clear to what extent the incompleteness could affect the performance of an AES model and, more importantly, what is the prerequisites for an AES model to function in terms of the number of training data. To mitigate this gap, this paper attempts to investigate the influences of the incompleteness of training data over several AES models by answering following research questions.

– RQ1: How the data sparsity problem affects the performance of different AES methods?
– RQ2: To make an AES model perform, how many rated essays are requested at least for training?

By answering the above questions, we hope to understand the reliances on the completeness of training data in different models and attempt to estimate the least manual workloads that required to make an AES system to perform. To this end, extensive empirical experiments are conducted on the standard Automated Student Assessment Prize (ASAP) dataset[1]. As shown by the results, the classical RankSVM and KNN models can learn effective AES models with as few as 20 labeled essays, and they tend to converge when being trained on 200 rated essays. Meanwhile, the AES models based on deep neural network normally require more training data, but could outperform the classical models right after being trained on enough labeled data.

2 Related Work

2.1 Automated Essay Scoring Algorithms

Most of existed Automated Essay Scoring (AES) algorithms view automated essay scoring as classification or regression problem [3,12–15]. They usually directly learn a classification or regression model based on hand crafted features, such as lexical and syntactic features, and estimate the score of an unseen essay with the prediction of the learned model. In addition, there are also works seeing AES as a ranking problem by applying pairwise learning to ranking algorithms on AES problem [5,22]. Instead of directly utilizing the prediction of learned

[1] https://www.kaggle.com/c/asap-aes.

model as the estimation of an unseen essay as classification or regression, they transform the ranking score given by learning to rank model into the estimation score of the unseen essay by heuristic or learning methods. Intuitively, AES algorithms based on learn to rank take the relative writing quantity between a given pair of essays compared to algorithms based on classification or regression. Experimental results in [5,22] also show the improvement of learning to rank approaches over traditional classification and regression algorithms. Specifically, Chen and He propose to incorporate the evaluation metric in AES into the loss function to directly optimize the agreement between human and machine raters [4].

Traditional AES models require much work on feature engineering to be effective. Recent years there have been efforts in developing AES approaches based on deep neural networks (DNN), for which feature engineering is no longer required. Alikaniotis et al. propose to learn score-specific word embeddings and utilize a two-layer bi-directional Long-Short Term Memory networks (bi-LSTM) followed by several dense layers to predict essay score [1]. Taghipour and Ng explore a variety of neural network model architectures based on recurrent neural networks [17]. Tay et al. further extend [17] by incorporating neural coherence features [18]. Some works first utilize CNN layers to learn representations of sentences and then CNN [8] or RNN [9] layers are applied to further learn representations of essays.

There are also some works working on the adaption problems between different essay prompts. Phandi et al. propose to apply correlated linear regression to solve the domain adaption problem in AES [14]. They train their AES model using the data from source prompt and a few target prompt essays, such as 10, 25, 50, 100 target prompt essays. Cummins et al. use a constrained multi-task pairwise preference learning approach to combine the data from different prompts to improve the performance [6]. They also find a few target prompt essays is needed to obtain effective results in terms of kappa, a prime evaluation metric for AES.

2.2 Pre-defined Essay Features

In this section, we introduce the hand crafted features used in this work for traditional AES models, such as RankSVM and KNN. These features are widely used in previous works [3,4,10,16] and are concluded in Table 1. The detailed description of the pre-defined hand crafted features are as follows:

Lexical Features

– *Statistics of word length*: The number of words with length in characters larger than 4, 6, 8, 10, 12 in each essay respectively. The *mean* and *variance* of word length in characters in each essay. In general, hard words are likely much longer in length. Statistics of word length are expected to reflect one's grasp of complex words.

Table 1. Hand-crafted features.

No.	Feature
1	Mean and variance word length in characters
2	Mean length of clauses
3	Essay length in characters and words
4	Number of spelling errors
5	The number of prepositions and commas
6	Mean number of clauses per sentence
7	Mean and variance of sentence length in words
8	Maximum number of clauses of a sentence
9	Semantic vector similarity based on *LSA*
10	Mean cosine similarity of word vectors by *tf-idf*
11	The average height of the parser tree of each sentence in an essay
12	Word bigram/trigram frequency *tf* divided by collection frequency *TF*
13	POS bigram/trigram frequency *tf* divided by collection frequency *TF*

– *Unique words*: The number of unique words in each essay, normalized by the essay length in words. This is expected to reflect a student's quantity of vocabulary.
– *Grammatical/Spelling errors*: The number of grammatical or spelling errors in each essay. An essay with too many grammatical or spelling errors usually hints a bad grasp of word spelling.

Syntactical Features

– *Statistics of sentence length*: The number of sentences whose length in words are larger than 10, 18 and 25 respectively. The *mean* and *variance* of sentence length in words. The variety of the length of sentences potentially reflects the complexity of syntactics.
– *Clauses*: The *mean* number of clauses in each sentence, normalized by the number of sentence in an essay. The *maximum* number of clauses of a sentence in an essay.
– *Sentence structure*: The average height of the parser tree of each sentence in an essay. The average of the sum of the depth of all nodes in a parser tree of each sentence in an essay.
– *Preposition and Comma*: The number of prepositions and commas in each sentence, normalized by sentence length in words.

Grammar and Fluency Features

– *Word bigram and trigram*: The grammar and fluency of an essay can be measured by the mean tf/TF of n-grams where tf is the frequency of n-gram in a single essay and TF is the corresponding frequency in the whole essay

collection. A n-gram with high tf/TF generally indicates it is wrong use. We utilize bigram and trigram tf/TF feature in this work.
– *POS bigram and trigram*: Similar with bigram and trigram features, the mean *tf/TF* of POS bigrams and trigrams are also incorporated.

Content and Prompt-Specific Features

– *Essay length*: The number of words and characters in an essay. Since the fourth root of essay length in words is proved to be highly correlated with the essay score [16], we use the fourth root of the essay length measured by the number of words and characters in experiments.
– *Word vector similarity*: Mean cosine similarity of word vectors, in which the element is the term frequency multiplied by inverse document frequency (tf-idf) of each word. It is calculated as the weighted mean of cosine similarities and the weight is set as the corresponding essay score.
– *Semantic vector similarity*: Semantic vectors are generated by Latent Semantic Analysis. The calculation of mean cosine similarity of semantic vectors is the same with word vector similarity.

All the features are normalized through min-max method to avoid several features are dominant compared to other features. For traditional models, it is important to have elaborate feature engineering procedures if we want obtain effective performance. These features are proved to be effective in previous works [3,4,10,16], thus chosen in this work.

3 Data Sparsity Simulation

In this section, we introduce the simulation method of data sparsity in automated essay scoring. We start with a brief introduction to the dataset and the evaluation metric used, followed by the data sparsity simulation method.

Dataset. The dataset used in the experiments is the Automated Student Assessment Prize (ASAP) dataset, the dataset used in the ASAP competition by Kaggle, which is widely used for AES [1,4,9]. The statistics of the dataset are summarized in Table 2. There are 8 sets of essays in the dataset from different prompts. All essays are written by students ranging in grade 7 to grade 10. And the essays are different in essay length and score range.

Evaluation Metric. Quadratic weighted Kappa (QWK) is used to measure the agreement between the predicted scores and the corresponding scores from human raters. It is the official evaluation metric in the ASAP competition and is widely used in previous works [1,4,22]. QWK is calculated as follows:

$$\kappa = 1 - \frac{\sum_{i,j} w_{i,j} O_{i,j}}{\sum_{i,j} w_{i,j} E_{i,j}}$$

where w, O and E are matrices of weights, observed scores and expected scores. $O_{i,j}$ is the number of essays that receive a score i from the first rater and a score j from the second rater. $w_{i,j} = (i - j)^2/(N - 1)^2$, where N is the number of possible scores. E is the outer product between the score vectors of the two raters, normalized to have the same sum as O.

Simulating Data Sparsity. For each of the eight essay sets in ASAP, we divide the dataset into training, validation and testing subsets in line with [1] by utilizing the ids set of validation and test set essays released by them. Concretely, 80% of each essay set are used for training and the remaining 20% for testing. Additionally, 20% of the training data are reserved for validation. The remaining training essays are then deemed a complete training set, out of which essays are sampled to simulate data incompleteness. A simple way of doing so is to randomly select essays from the training set. However, a drawback of this simple simulation would be that the random sample may not reflect the actual score distribution in the complete training set, which in turn biases the training of the AES model. Therefore, in this paper, in order to simulate the data sparsity, the training essays are first sorted by their actual scores, and then partitioned into 5 equal-size bins. From each bin, equal number of essays are randomly sampled as the incomplete training set. In this work, the following training set sizes are sampled to simulate the data sparsity: [5, 10, 15, 20, 25, 30, 50, 100, 150, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100]. In consideration of the randomness may lead to large variance especially when training set size is small, the random sampling is repeated 10 time for each sample size, and the average performance trained from the 10 random samples are reported. Compared to the simple simulation that samples from the whole training set, sampling from the sorted bins is expected to preserve the score distribution such that the effect of the random sampling on learned AES model can be minimized.

Table 2. Statistics for the ASAP dataset.

Prompt	#Essays	Grade level	Avg length	Score range
1	1783	8	350	2–12
2	1800	10	350	1–6
3	1726	10	150	0–3
4	1772	10	150	0–3
5	1805	8	150	0–4
6	1800	10	150	0–4
7	1569	7	250	0–30
8	723	10	650	0–60

4 Experimental Setup

In this section, we first introduce the dataset, AES models and evaluation metric. Then we describe the details about how we simulate the data sparsity.

4.1 AES Models

This study uses two classical AES methods based on RankSVM and k-nearest neighbor algorithms, respectively. In addition, a recent state-of-the-art AES model based on end-to-end deep neural networks is also involved in this study. Details of the AES models used are given below.

- **RankSVM** is a pairwise learning to rank algorithm and is applied on AES problem in many works [5,22]. RankSVM regards AES problem as a ranking problem by taking the relative writing quality for a given essay pair into consideration. First, it assigns a ranking score to each essay in both training and testing essay sets. Then the predicted score of an essay in the testing set is estimated by averaging the human rated score of the K training essays whose ranking scores are nearest the testing essay's. The value of K is chosen from [6, 8, 10, 12, 14] by maximizing the performance on validation set. The linear kernel RankSVM[2] is used with the parameter C amongst [3, 4, 5, 6, 7].
- **K nearest neighbors.** K nearest neighbors (KNN) [2] is a simple non-parametric algorithm, of which the main idea is to estimate one's properties by considering its K nearest neighbors. In this work, we estimate the score of an essay in testing set by averaging the human rated scores of it's K nearest neighbor essays in training set. The distance is measured by euclidean distance. The value of K is also amongst [4, 6, 8, 10, 12, 14].
- **Neural Model.** Deep Neural network (DNN) is popular in recent years for their good performance on many tasks without feature engineering and has already been successfully applied on AES [1,9,17,18]. In this work, we utilize the architecture of the neural network in [17], which is a state-of-art neural model for AES. First, a LSTM network is applied on the word embedding sequence of a given essay to obtain a list of hidden states. Then these hidden states are averaged and fed into a dense layer with sigmoid activation to get the predicted score, ranging in [0, 1]. The RMSProp optimization algorithm is used to minimized the mean squared error (MSE) loss. The batch size are set to 32. Since the human rated scores are not in [0, 1], we transform the original scores into [0, 1] for training and back into the original range for evaluation. Word embeddings are initialized by the pre-trained embeddings released by [24]. Other setups are also completely in line with [17], we recommend to refer to [17] for space reason.

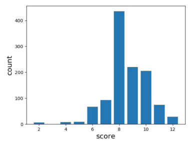
5 Results and Analysis

In this section, we present the experimental results. The kappa performance of RankSVM, KNN and DNN on incomplete training sets with different sizes are presented in Tables 3, 4 and 5, respectively. As it is widely accepted that the human-machine agreement of an AES system, measured by Kappa, should be

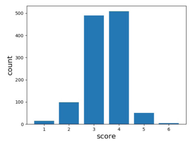
[2] http://svmlight.joachims.org/.

Table 3. The kappa performance of *RankSVM* on incomplete training sets with different sizes.

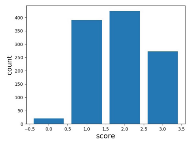
Size	Prompt Id							
	1	2	3	4	5	6	7	8
5	0.3212	0.2350	0.1793	0.3454	0.2611	0.3292	0.2687	0.2697
10	0.6063	0.4560	0.4141	0.5120	0.5997	0.4808	0.5050	0.4750
15	0.6592	0.5378	0.5041	0.5432	0.6695	0.5933	0.5654	0.5099
20	***0.7031***	0.5899	0.5736	0.5665	***0.7492***	0.6524	0.5973	0.5316
25	*0.7424*	0.6192	0.5533	0.5797	*0.749*	0.6570	0.6255	0.5674
30	*0.7514*	0.6457	0.5679	0.5805	*0.7613*	0.6601	0.6436	0.5524
50	*0.7515*	0.6522	0.5675	0.6386	*0.771*	0.6772	0.6609	0.5666
100	*0.7613*	0.6726	0.5943	0.6613	*0.7818*	***0.7005***	0.6974	0.5651
150	*0.7815*	0.6746	0.6318	0.6617	*0.7900*	*0.7130*	0.6981	0.5846
200	*0.7848*	0.6732	0.6378	0.6715	*0.7983*	*0.7261*	***0.7120***	0.5928
300	*0.7983*	0.6824	0.6378	0.6811	*0.8082*	*0.7304*	*0.7297*	0.5985
400	*0.8006*	0.6875	0.6378	0.6856	*0.8072*	*0.7471*	*0.7349*	0.5945
500	*0.8009*	0.6876	0.6461	0.6868	*0.8116*	*0.7508*	*0.7394*	0.5945
600	*0.8041*	0.6885	0.6492	0.6908	*0.8153*	*0.7522*	*0.7442*	-
700	*0.8125*	0.6885	0.6471	0.6945	*0.8201*	*0.7478*	*0.7469*	-
800	*0.8137*	0.6897	0.6471	0.6939	*0.8168*	*0.7499*	*0.7416*	-
900	*0.8137*	0.6897	0.6471	0.6963	*0.8181*	*0.7499*	*0.7412*	-
1000	*0.8133*	0.6895	0.6471	0.6968	*0.8191*	*0.7499*	-	-
1100	*0.8144*	0.6895	0.6520	0.6968	*0.8169*	*0.7499*	-	-



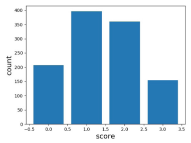
(a) Prompt 1



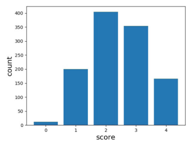
(b) Prompt 2



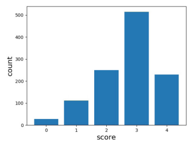
(c) Prompt 3



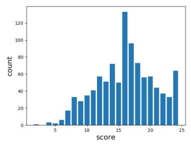
(d) Prompt 4



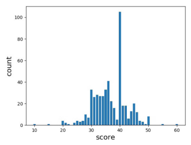
(e) Prompt 5



(f) Prompt 6



(g) Prompt 7



(h) Prompt 8

Fig. 1. The human rated score distribution of each prompt's training essays.

Table 4. The kappa performance of *KNN* on incomplete training sets with different sizes.

Size	Prompt Id							
	1	2	3	4	5	6	7	8
5	0.2770	0.2374	0.0480	0.2619	0.2842	0.1022	0.1379	0.1649
10	0.5488	0.4603	0.2648	0.5135	0.5674	0.3773	0.3701	0.4085
15	0.5998	0.5266	0.4573	0.5684	0.6614	0.5205	0.4341	0.4480
20	0.6520	0.5451	0.5504	0.5853	***0.7056***	0.5634	0.4924	0.4458
25	0.6500	0.5580	0.5712	0.6016	*0.7192*	0.5864	0.5418	0.4782
30	0.6547	0.5737	0.5972	0.6166	*0.7222*	0.6050	0.5472	0.5040
50	0.6706	0.5754	0.5999	0.6161	*0.7373*	0.6155	0.5637	0.5481
100	***0.7119***	0.5877	0.6008	0.6313	*0.7616*	0.6288	0.5897	0.5789
150	*0.7201*	0.6205	0.6062	0.6330	*0.7660*	0.6442	0.6218	0.5969
200	*0.7324*	0.6432	0.6078	0.6564	*0.7656*	0.6443	0.6400	0.6216
300	*0.7386*	0.6523	0.6082	0.6579	*0.7678*	0.6527	0.6423	0.6177
400	*0.7471*	0.6505	0.6034	0.6547	*0.7695*	0.6692	0.6575	0.6190
500	*0.7641*	0.6508	0.6093	0.6605	*0.7802*	0.6817	0.6682	0.6162
600	*0.7617*	0.6531	0.6178	0.6611	*0.7812*	0.6791	0.6675	-
700	*0.7639*	0.6535	0.6178	0.6594	*0.7812*	0.6790	0.6734	-
800	*0.7693*	0.6535	0.6178	0.6613	*0.7825*	0.6790	0.6753	-
900	*0.7658*	0.6591	0.6188	0.6613	*0.7825*	0.6857	0.6744	-
1000	*0.7654*	0.6585	0.6184	0.6589	*0.7841*	0.6881	-	-
1100	*0.7821*	0.6549	0.6184	0.6589	*0.7841*	0.6938	-	-

at least 0.70 [19], a training data set is considered sufficient if the learned model results in a performance that meets the 0.70 requirement. Therefore, in the tables, the kappa values larger than 0.70 are in *italic*, and the results improved from below 0.70 are in bold. From the experimental results in Tables 3, 4 and 5, we attempt to answer the two main research questions raised in Sect. 1 as follows.

For RQ1, the performance of traditional classification models (RankSVM and KNN) increases fast even if the number of training essays are very small, from 5 to 30. A likely cause for this observation is that the classical RankSVM and KNN models have relatively high tolerance to data sparsity, and can achieve reasonable performance with a small number of training data. Compared to RankSVM and KNN, the performance of the deep neural network model (DNN) appears to be sensitive to data sparsity. When the training set is small, the performance of DNN is much lower than RankSVM and KNN, and the Kappa value is lower than 0.70 on all 8 prompts until the training set size is increased to 200 (see Table 5). We believe this is due to the fact that, compared to RankSVM and KNN, the deep neural network model is complicated, with many parameters to

Table 5. The kappa performance of *DNN* on incomplete training sets with different sizes.

Size	Prompt Id							
	1	2	3	4	5	6	7	8
5	0.1077	0.2116	0.3265	0.3391	0.2611	0.2099	0.2509	0.2173
10	0.2826	0.1861	0.3634	0.4574	0.3172	0.1809	0.3704	0.1581
15	0.2097	0.2604	0.3264	0.5000	0.3909	0.2187	0.3838	0.2568
20	0.2643	0.2537	0.3624	0.5299	0.3283	0.2319	0.3517	0.2551
25	0.2921	0.2576	0.3635	0.5327	0.3973	0.2373	0.4440	0.2507
30	0.2134	0.2061	0.3902	0.4673	0.406	0.2878	0.4020	0.2872
50	0.2764	0.2982	0.4903	0.5462	0.4741	0.3373	0.5146	0.2981
100	0.3294	0.3432	0.5082	0.6191	0.6417	0.4088	0.6079	0.3441
150	0.4274	0.3757	0.5563	0.6411	0.6618	0.5000	0.6696	0.4792
200	0.4537	0.3953	0.6184	0.6803	***0.7200***	0.5398	0.6993	0.4788
300	0.5371	0.4806	0.6469	***0.7183***	*0.7808*	0.6174	***0.7194***	0.5107
400	0.5746	0.5138	0.6747	*0.7508*	*0.7971*	0.6507	*0.7595*	0.5676
500	0.6075	0.5510	0.6845	*0.7590*	*0.7991*	0.6987	*0.8029*	0.5651
600	0.6172	0.5828	0.6933	*0.7647*	*0.8063*	***0.7404***	*0.8106*	-
700	0.6615	0.6310	0.6867	*0.7615*	*0.8107*	*0.7564*	*0.8189*	-
800	***0.7740***	0.6348	0.6857	*0.7828*	*0.8142*	*0.7667*	*0.8254*	-
900	*0.7780*	0.6852	0.6975	*0.7809*	*0.8148*	*0.7632*	*0.8281*	-
1000	*0.7894*	0.6774	0.6916	*0.7889*	*0.8225*	*0.7859*	-	-
1100	*0.8097*	0.6998	0.6945	*0.7955*	*0.8195*	*0.7870*	-	-

learn, and consequently, requires a large amount of training data. On the other hand, the neural network model outperforms the two classical models with the use of all training data available, showing that the deep model has strong ability in learning rating patterns out of sufficient training data.

For RQ2, we find that the Kappa values of RankSVM on prompts 1 and 5 are larger than 0.7 when the training set size is only 20. In addition, RankSVM achieves Kappa ≥ 0.7 on prompts 6 and 7 with training set sizes of 100 and 200, respectively. As for KNN, it achieves Kappa ≥ 0.7 on prompts 1 and 5 with training set size of 100 and 20. Compared to traditional methods, DNN needs much more training data to surpass the 0.70 threshold, of which the sizes are 800, 300, 200, 600 and 300 for prompts 1, 4, 5, 6, 7, respectively. Overall, the minimal number of training essays required to learn an effective prompt-specific AES model depends on the data and the learning algorithm used. For example, RankSVM requires only 20 training essays to reach the Kappa = 0.70 threshold on prompts 1 and 5, but is unable to outperform this threshold on prompts 2–4 even if all training data available are used. To investigate this issue, we plot the real score distribution of essays for all 8 prompts in Fig. 1. As we can see

Table 6. The performance loss in Kappa in percentage against the use of all training data available.

Size	Model	Prompt Id							
		1	2	3	4	5	6	7	8
30	RankSVM	8.38	6.82	14.8	20.1	7.73	14.0	16.1	8.35
	KNN	19.5	14.9	3.62	7.26	8.57	14.7	23.4	23.3
	DNN	279	239	78.8	70.0	102	173	106	97.6
50	RankSVM	8.37	5.76	14.9	9.11	6.37	11.1	13.0	5.64
	KNN	16.6	14.6	3.16	7.34	6.35	12.7	19.8	13.4
	DNN	192	134	42.3	45.6	73.5	133	60.9	90.4
100	RankSVM	6.97	2.55	9.71	5.36	4.90	7.39	7.10	5.91
	KNN	9.86	12.1	3.00	4.75	2.95	10.3	14.5	7.39
	DNN	145	104	37.3	28.5	28.1	92.5	36.2	65.0
150	RankSVM	4.21	2.25	3.19	5.30	3.81	5.51	7.00	2.39
	KNN	8.60	6.23	2.08	4.48	2.36	7.70	8.60	4.14
	DNN	89.5	86.3	25.4	24.1	24.3	57.4	23.7	18.5
200	RankSVM	3.77	2.46	2.22	3.77	2.73	3.59	4.91	0.970
	KNN	6.78	2.46	1.82	0.750	2.41	7.69	5.52	0
	DNN	78.5	77.0	12.8	16.9	14.2	45.8	18.4	18.6

from this figure, the score distributions of essays written for prompts 1 and 5 are more diverse than those written for prompts 2–4, and it is likely the case that a majority of essays written for prompts 2–4 are of medium quality such that the learned model is unable to differentiate between essays with similar quality. Similar observation can also be made from the score distribution of prompt 8, where most essays receive scores from 30 to 50, and in particular, more than 20% of the training essays (100 out of 500) are rated 40. As a result, it is difficult for the AES model to learn the nuances between essays with very close ratings (e.g. 39 and 40). Note that for some of the prompts, the performance of the AES model can never reach the kappa = 0.70 threshold. We suggest that this is caused by the biased distribution of essays scores, and in this case, more training essays with diverse quality are needed for learning an effective AES model.

To further investigate, we present the performance loss in Kappa, with respect to the use of all training data available, with training data sizes of 30, 50, 100, 150, and 200 essays, in Table 6. When training set sizes are too small, we can find the performances of DNN are far from the corresponding best performance, i.e. the last row in Table 5. From Table 6 we find that when there are only 30 essays in training set, RankSVM is able to achieve good performances on prompts 2 and 5, of which the kappa losses compared to the best performance obtained on much more training essays are only 6.82% and 7.73% respectively. For KNN, the kappa losses on prompts 3, 4 and 5 at training set size 30 are 3.62%, 7.26% and 8.57%, showing that this model converges with only a small number of training essays. The kappa losses on all eight prompts at training sizes 100 and 200 are within 10% and 5% respectively for RankSVM. This demonstrates that traditional methods are able to achieve relatively stable performance with a few training essays at the cost of small kappa losses, whereas the DNN model requires a large amount of training data to learn, and can indeed lead to the performance that is better than RankSVM and KNN with the availability of enough training data, as shown in the results in Tables 3, 4 and 5.

6 Conclusions

This paper has conducted comprehensive experiments to investigate two key research questions (RQs) regarding the data sparsity problem in automated essay scoring. According to the results, for RQ1, compared to the classical RankSVM and KNN models, the recent deep neural networks are much more sensitive to data sparsity due to its model complexity. For RQ2, in general, the classical models like RankSVM and KNN require a small number of training essays due to their tolerance to data sparsity. They can learn an effective AES model with as few as 20 training essays. Both RankSVM and KNN converge with about 200 training essays at most. The DNN model, in contrast, is relatively data-hungry due to its model complexity. According to the results, the DNN model does not appear to converge even if all training data available are used. Even though, the performance of the AES model learned by DNN is better than RankSVM and KNN in most cases when more than 1,000 training essays are used.

As indicated by the results, in real-life applications, for a given essay prompt, i.e. there are only very few rated essays for training, or even no rated essays at all, it is recommended to use RankSVM for its high tolerance to data sparsity. It is also recommended to rate 20–100 essays by human assessors in order to learn an effective AES model by RankSVM. On the other hand, if there are enough, namely more than 1,000 rated essays available for a given prompt, it is recommended to use neural network model to learn the AES model to fully utilize DNN's power in learning patterns out of sufficient training data. Finally, the results indicate that the performance of prompt-independent AES [11] can be potentially improved by including a small number of labeled essays written for the target prompt.

Acknowledgments. This work is supported by the National Natural Science Foundation of China (61472391).

References

1. Alikaniotis, D., Yannakoudakis, H., Rei, M.: Automatic text scoring using neural networks. In: ACL (1). The Association for Computer Linguistics (2016)
2. Altman, N.S.: An introduction to kernel and nearest-neighbor nonparametric regression. Am. Stat. **46**(3), 175–185 (1992)
3. Attali, Y., Burstein, J.: Automated essay scoring with e-rater® v. 2. J. Technol. Learn. Assess. **4**(3), 1–31 (2006)

4. Chen, H., He, B.: Automated essay scoring by maximizing human-machine agreement. In: EMNLP, pp. 1741–1752. ACL (2013)
5. Chen, H., Jungang, X., He, B.: Automated essay scoring by capturing relative writing quality. Comput. J. **57**(9), 1318–1330 (2014)
6. Cummins, R., Zhang, M., Briscoe, T.: Constrained multi-task learning for automated essay scoring. In: ACL (1), pp. 789–799. The Association for Computer Linguistics (2016)
7. Dikli, S.: An overview of automated scoring of essays. J. Technol. Learn. Assess. **5**(1) (2006)
8. Dong, F., Zhang, Y.: Automatic features for essay scoring - an empirical study. In: EMNLP, pp. 1072–1077. The Association for Computational Linguistics (2016)
9. Dong, F., Zhang, Y., Yang, J.: Attention-based recurrent convolutional neural network for automatic essay scoring. In: CoNLL, pp. 153–162. Association for Computational Linguistics (2017)
10. Foltz, P.W., Laham, D., Landauer, T.K.: Automated essay scoring: applications to educational technology. In: World Conference on Educational Multimedia, Hypermedia and Telecommunications, pp. 939–944 (1999)
11. Jin, C., He, B., Hui, K., Sun, L.: TDNN: a two-stage deep neural network for prompt-independent automated essay scoring. In: ACL. The Association for Computer Linguistics (2018)
12. Larkey, L.S.: Automatic essay grading using text categorization techniques. In: SIGIR, pp. 90–95. ACM (1998)
13. Mcnamara, D.S., Crossley, S.A., Roscoe, R.D., Allen, L.K., Dai, J.: A hierarchical classification approach to automated essay scoring. Assess. Writ. **23**, 35–59 (2015)
14. Phandi, P., Chai, K.M.A., Ng, H.T.: Flexible domain adaptation for automated essay scoring using correlated linear regression. In: EMNLP, pp. 431–439. The Association for Computational Linguistics (2015)
15. Rudner, L.M.: Automated essay scoring using Bayes' theorem. Nat. Counc. Measur. Educ. New Orleans La **1**(2), 3–21 (2002)
16. Shermis, M.D., Burstein, J. (eds.): Automated Essay Scoring: A Cross Disciplinary Perspective. Lawrence Erlbaum Associates, Hillsdale (2003)
17. Taghipour, K., Ngm H.T.: A neural approach to automated essay scoring. In: EMNLP, pp. 1882–1891. The Association for Computational Linguistics (2016)
18. Tay, Y., Phan, M.C., Tuan, L.A., Hui, S.C.: SkipFlow: Incorporating neural coherence features for end-to-end automatic text scoring. CoRR, abs/1711.04981 (2017)
19. Williamson, D.M., Xi, X., Jay Breyer, F.: A framework for evaluation and use of automated scoring. Educ. Measur.: Issues Pract. **31**(1), 2–13 (2012)
20. Williamson, D.M.: A framework for implementing automated scoring. In: Annual Meeting of the American Educational Research Association and the National Council on Measurement in Education, San Diego, CA (2009)
21. Yang, Y., Buckendahl, C.W., Juszkiewicz, P.J., Bhola, D.S.: A review of strategies for validating computer-automated scoring. Appl. Measur. Educ. **15**(4), 391–412 (2002)
22. Yannakoudakis, H., Briscoe, T., Medlock, B.: A new dataset and method for automatically grading ESOL texts. In: ACL, pp. 180–189. The Association for Computer Linguistics (2011)
23. Zesch, T., Wojatzki, M., Scholten-Akoun, D.: Task-independent features for automated essay grading. In: BEA@NAACL-HLT, pp. 224–232. The Association for Computer Linguistics (2015)
24. Zou, W.Y., Socher, R., Cer, D.M., Manning, C.D.: Bilingual word embeddings for phrase-based machine translation. In: EMNLP, pp. 1393–1398. ACL (2013)