Automated Essay Scoring Using Incremental Latent Semantic Analysis

Mingqing Zhang[a], Shudong Hao[a], Yanyan Xu[a]*, Dengfeng Ke[b], Hengli Peng[c]
[a] School of Information Science and Technology, Beijing Forestry University
Email: xuyyxu@gmail.com
[b] Institute of Automation, Chinese Academy of Sciences
Email: dengfeng.ke@ia.ac.cn
[c] Institute of Educational Measurement, Beijing Language and Culture University
penghl6402@aliyun.com

Abstract— **Writing has been increasingly regarded by the testers of language tests as an important indicator to assess the language skill of testees. As such tests become more and more popular and the number of testees becomes larger, it is a huge task to score so many essays by raters. So far, many methods have been used to solve this problem and the traditional method is Latent Semantic Analysis (LSA). In this paper, we introduce a new incremental method of LSA to score essays effectively when the dataset is massive. By comparison of the traditional method and our new incremental method, concerning the running time and memory usage, experimental results make it obvious that the incremental method has a huge advantage over the traditional method. Furthermore, we use real corpora of test essays submitted to the MHK test (Chinese Proficiency Test for Minorities), to demonstrate that the incremental method is not only efficient but also effective in performing LSA. The experimental results also show that when using incremental LSA, the scoring accuracy can reach 88.8%.**

Index Terms— **automated essay scoring, incremental latent semantic analysis, singular value decomposition**

I. Introduction

WRITING is an essential part of language tests, and it is an important indicator to assess a students' language skill. In such tests, students are usually required to write an essay based on a given question, and raters score these essays according to some given criteria. Traditionally, because the raters usually combine their subjective judgements with these criteria, the human scoring methods will lead to an inaccuracy of grading [1]. At the same time, as the amount of testees increases rapidly, human scoring is a labor-intensive task [2] [3]. Therefore, a more accurate and faster automated scoring method has been desired for a long time.

Researchers have devoted to the automated scoring field for twenty years. PEG [4] is the earliest system for automated scoring, in which the part of speech, the amount and the usage of words are the primary concerns. E-rater [5] is of great use for GMAT, developed by Educational Test Service (ETS). Its features include the analysis of the discourse structure, the syntactic structure and the vocabulary usage [6]. Bayesian Essay Test Scoring System (BETSY) [7] is based on Multivariate Bernoulli Model and the Bernoulli Model. Intelligent Essay Assessor (IEA) [8], which is based on LSA, compares the contents among essays. By comparison of words presented in the dataset and making the relations of the words clear, any essay can be scored under the semantic space constructed by the dataset. In addition to these systems, some other novel methods have also been introduced, such as multi-classifier fusion [9] and so forth.

Usually, there are three levels upon which to evaluate an essay [10]: $word$ (including characters in Chinese or other Asian languages, or spelling in western languages), $sentence$ and $paragraph$. $Word$ is the basic part and testees are required to use correct characters, spelling and meaningful words; $sentence$ means students are required to think about the confluence of sentences and inter-sentences, and the relations between the topic and these sentences; $paragraph$ is a consideration about the logic relations among paragraphs and even the whole passage.

As to the current technique on automated essay scoring, $word$ remains an basic and crucial part of assessing an essay [11]. In real world, there are numerous ways to express one thing. This character of any natural language not only brings up the synonyms and ambiguities in the semantics, but also the sparsity and scalability in natural language processing, which leads to difficulties in computation.

LSA is a technique that has been successfully applied into a wide range of fields and industries [12] [13] [14] [15], such as bioinformation [16] [17], web document comprehending [18], language processing [19] [20] [21] and signal processing [22]. It is used for comparing the essays in a reduced dimensionality semantic space based on the words they contain [23] [24]. Its main tasks are to avoid the surface of the language comlexity, and to understand the true meaning the words are expressed from a semantic perspective. According to the words appeared in the dataset, a weighted matrix is used to reduce the

*The corresponding author is Yanyan Xu.

This work was supported in part by the Beijing Higher Education Young Elite Teacher Project, the Fundamental Research Funds for the Central Universities (GrantNo.201322) and the National Natural Science Foundation of China (GrantNo.61103152).

dimensionality, and to represent the true significance of the words.

Singular value decomposition (SVD) is the main algorithm for LSA to produce low-rank approximations [25]. By SVD, the original dictionary-based space will be divided into three subspaces whose elements are often regarded as semantics. Obsoleting useless semantics and multiplying the new matrices will reconstruct the semantic-based space. This technique which shows viability theoretically, however, cannot be used effectively even practically when faced with massive streams of data. Earlier experiments showed that neither memory usage nor time consuming can support such huge a dataset for SVD.

In order to resolve the problems regarding memory usage and time consuming, incremental SVD has been successfully introduced by Matthew Brand [26], and has been implemented into the fields of image processing [27], information retrieval such as recommender systems [28] [29] and natural language processing [23]. Automated essay scoring using incremental LSA, however, is a new method. Although many researchers have introduced LSA [30] [31] [32], no one has ever used incremental LSA and incremental SVD. In this paper, we use incremental SVD as a part of incremental LSA, to process huge datasets of test essays. Experimental results show that this incremental method is effective to reduce the usage of memory and time consuming without lowering the performance of automated scoring in comparison with human scoring.

The rest of the paper is organized as follows. In section II we introduce the basic idea of SVD and incremental SVD. In section III we describe the proposed method and we show our experimental results and discussion in section IV. Finally, in section V we conclude this paper and point out the future work.

II. Preliminaries

A. Conventional batch SVD

The underlying algorithm of LSA is SVD, which can construct a semantic space of a given dataset. Given r-rank matrix M, upon which we apply SVD:

$$M = U\Sigma V^T \tag{1}$$

where U and V are orthogonal matrices, and the elements in Σ are singular values those are in descending order.

Specially, in natural language processing, maintaining only $k \ll r$ will produce a lower dimensionality and better approximation about the original matrix M. By removing the (r - k) diagonal elements, (r - k) columns in U and (r - k) rows in V and (r - k) elements in Σ where the elements with far too small values are considered to be noise and unnecessary, we can multiply the matrices and get the approximation to the original matrix:

$$M'_{m\times n} = U_{m\times k}\Sigma_{k\times k}V^T_{k\times n}. \tag{2}$$

B. Update SVD by adding columns

Suppose the matrix C, which consists of additional columns, will be added to the original matrix M, then we firstly apply traditional SVD on M and get the result of formula (1). With being mathematically proved, by adding additional C, we can get:

$$[MC] = [UJ]\begin{bmatrix}\Sigma & L\\ 0 & K\end{bmatrix}\begin{bmatrix}V^T & 0\\ 0 & I\end{bmatrix}^T \tag{3}$$

where $L = U^TC$. Let $H = C - UL$, then by QR decomposition or other methods upon H, we will get $H \xrightarrow{QR} JK$. The matrix in the middle $\begin{bmatrix}\Sigma & L\\ 0 & K\end{bmatrix}$ will be continually applied with traditional SVD. Finally, we multiply all medium results and get the updated result:

$$[MC] = [UJ]U' \cdot \Sigma' \cdot \begin{bmatrix}V^T & 0\\ 0 & I\end{bmatrix}^T V'^T = U'' \cdot \Sigma' \cdot V''^T. \tag{4}$$

For a more specific proof and geometric interpretation, please refer to [26] in detail.

III. The proposed Method

Traditionally, LSA can be performed through five sub-steps and these five sub-steps are text preprocessing, weighting, calculating SVD, correlation measurement and correlation method [33]. However, calculating SVD becomes a hard even impossible task when faced with a massive dataset. So, we use incremental LSA to resolve the problem effectively, and its flowchart is shown in Figure 1. At first, all the essays will be segmented into words and constructed into the essay vectors which form the original dataset matrix as Figure 2 shows. Next, applying incremental algorithm on that matrix will establish a semantic space, where all the essay vectors can be re-projected. Finally, all the essay vectors with their human scoring results as labels will be sent into the support vector machine for training. Thus, all the essays in the dataset will have their own predicted automated scoring.

A. Text pre-processing

Text pre-processing is the first sub-step and it includes two parts: segmenting and producing t-d matrix. The t-d (term-document) matrix is based on the amount of words (terms) appeared in the essays (documents). For example, $a_{i,j}$ is the number of times of the i-th word appeared in the j-th essay in the dataset. In this paper, we regard each column as an *essay vector* which is denoted as d_j, whose feature is the number of words in the training set or that of latent semantics.

Because the essays in the dataset are represented as texts without space as delimiter of words, it is necessary to segment the texts reasonably. There are many methods that have been developed, for example, tri-gram-based method that is used for disperse string detection [34], using lexicon dictionary [35] method and other methods.

Due to that word segmenting is a direct factor that influences the preciseness of t-d matrix, even further, that

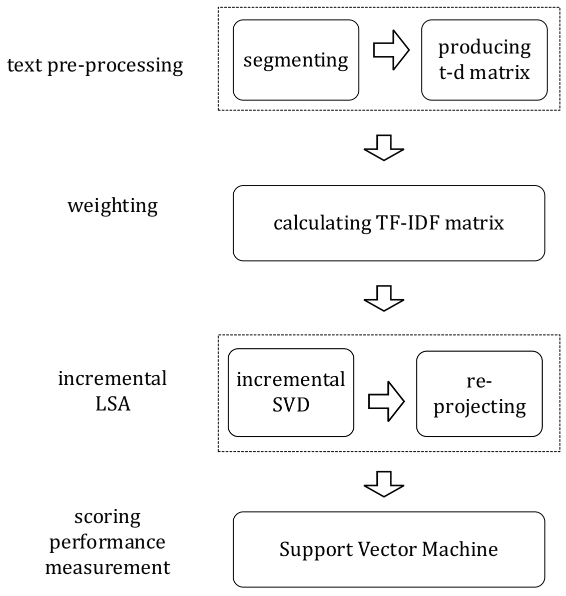

Figure 1. The flowchart of the proposed method

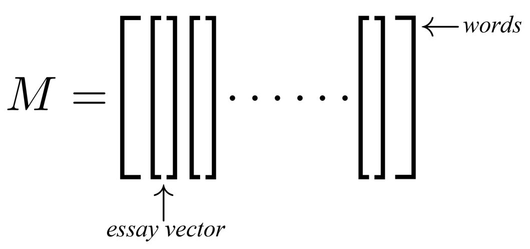

Figure 2. Original dataset matrix

of TF-IDF matrix, we use weighted finite-state transducer(WFST), which is based on our previous research [36], to produce reasonable segmentation.

Additionally, eliminating stopwords is a necessary step, since they are of high frequencies without real meaning, such as preposition, verbal auxiliaries and so forth. After the t-d matrix of the dataset has been generated, the word corpus will be confirmed as well. The subsequent steps, including matrix weighting, will base on this word corpus. Specifically, the training set and the testing set are sharing the same word corpus confirmed by the training set.

B. Weighting

The TF-IDF matrix, where TF stands for the term frequency and IDF for inverse document frequency, is a common method for weighting [37]. Every element in the m-n matrix can be weighted as

$$w_{i,j} = TF_{i,j} \times IDF_{i,j} \tag{5}$$

The term frequency, $TF_{i,j}$ is defined as

$$TF_{i,j} = \frac{num_{i,j}}{\sum_{k=1}^{m} num_{k,j}} \tag{6}$$

where $num_{i,j}$ is the number of the i-th word appeared in the j-th essay, and m is the number of the words in the training set. As to IDF, it can be calculated as

$$IDF_{i,j} = log\frac{n}{1 + DF_i} \tag{7}$$

where n is the size of the training set (or the amount of essay vectors), and DF_i is the number of essays which contain the i-th word.

For generating the TF-IDF matrix of the testing set, a special situation exists that no essay contains the i-th word, so, in case of dividing zero, we add 1 to the denominator.

Algorithm 1 Incremental SVD

```
Input: Matrix $M$ with $n$ essay vectors
Output: Matrix $U, \Sigma, V$
 1: $n' \leftarrow initialized\ value$
 2: $add\_num \leftarrow batch\ size$
 3: $M' \leftarrow$ the first $n'$ essay vectors of $M$
 4: $[U\Sigma V] \leftarrow$ svd($M'$,0)
 5: the first $n'$ essay vectors $\leftarrow 0$
 6: repeat
 7:     Matrix $C \leftarrow$ the first $add\_num$ essay vectors $\neq 0$ of $M$
 8:     $current\_rank \leftarrow U.rank$
 9:     $[tU, t\Sigma, tV] \leftarrow svds([M'C], current\_rank)$
10:     Update($[U, \Sigma, V], [tU, t\Sigma, tV]$)
11:     the first $add\_num$ essay vectors of $M \leftarrow 0$
12: until M=0
```

C. Incremental LSA

Incremental LSA is composed of two parts: incremental decomposition of a dictionary-based space and re-projection of any essay vector under the reconstructed semantic space. The first part can be completed by incremental SVD, which avoids the synonyms and ambiguities about words and enables the essay vectors to be projected onto the low-dimension semantic space. The second part is called re-projection, and any essay vector based on term-frequency can be represented under that space. These components will be described below.

1) Incremental SVD: The core of LSA is performing SVD on TF-IDF matrix for training. Algorithm 1 is our method of incremental SVD.

We get the first M essay vectors, called $initialized\ value$, to apply the conventional SVD, which is faster because of the less essay vectors and hence far too smaller matrix. Then, we obtain a mediate result of decomposition.

The next step is to update the mediate result by adding N columns, called *batch size*, from the rest part of TF-IDF matrix. According to the mathematical derivation we introduced previously, it is easier and faster to update the mediate result, and it has been proved that this updating result is approximate to the conventional SVD of the original matrix [26]. This process of updating will repeat until all the essay vectors in the TF-IDF has been added to the mediate result. Finally, we can get the result of incremental SVD.

During the process, an important problem is to maintain necessary semantics used to construct the semantic space. As the decomposition goes, the rank of Σ will possibly increase, and thus, the redundant semantics will be produced. Producing too much noise will not only lower the preciseness of the semantic space, but also has impact on computational performance, i.e, larger memory usage for saving data and longer running time. Therefore, we maintain k latent semantics, called $threshold\ value$, to guarantee the effectiveness during the updating procedure and removing useless semantics immediately.

Incremental SVD is a faster algorithm than conventional method, especially for massive datasets. In fact, as the batch size become larger, the processing time is relatively becoming less and the usage of memory less either. Experimental results will be given in the next section.

2) Re-projecting: Applying formula (2), where k is the threshold value or the number of semantics remained, will construct a semantic space of the dataset. Any essay vector based on term-frequency and the same word corpus, d_j, can be re-projected to the semantic space.

Suppose that $U\cdot\Sigma\cdot V^T$ is the final result of incremental SVD performed on TF-IDF matrix of the training set, and d_j is an essay vector based on the same word corpus. Then by Formula (8), we will re-project d_j to the semantic space as :

$$\widehat{d_j} = \Sigma^{-1}\cdot U^T\cdot d_j \quad (8)$$

3) Scoring performance measurement: In this final part, we use Support Vector Machine (SVM) to automatically score the essays.

The incremental method is effective for processing massive datasets, especially for automated essay scoring. The word corpus is established on the training set, which is used to produce t-d matrix for the testing set. Additionally, all the essay vectors in the testing set will be re-project to the semantic space constructed by the training set.

According to [38], the optimized solution of SVM can be expressed as follows:

$$\min_{w,b,\xi} \quad \frac{1}{2}w^Tw + C\sum_{i=1}^{l}\xi_i \quad (9)$$
$$\text{subject to} \quad y_i(w^T\phi(x_i)+b)\geq 1-\xi_i,$$
$$\xi_i\geq 0, i=1,\ldots,l,$$

where y_i can be used as human scoring in our experiments. The number n is the threshold value, $x_i \in R^n$ is the essay vector which will be mapped into a high-dimensional space by $\phi(x_i)$, and C is the regularization parameter.

$$\begin{aligned} K(x_i,x_j) &= \phi(x_i)^T\phi(x_j) \\ &= exp(-\gamma||x_i-x_j||^2), \gamma>0 \end{aligned} \quad (10)$$

Essay vectors from the training set and their human scores are used to generate support vectors, and to establish decision benchmarks. Upon those benchmarks, the essay vectors from the testing set are classified according to support vectors. By this kind of classification, their class labels of essay vectors from the testing set are the automated scores predicted by SVM.

Comparing the human scores with the predicted scores from the testing set, we can estimate the performance of incremental LSA used in automated essay scoring with massive datasets.

IV. Experiments

A. Enviroment and datasets

In text pre-processing, we use our previous study WFST written in C++ language [36] to segment the essays. In the incremental procedure, we use MATLAB and C language to compute matrices. Finally, we use LIBSVM [39] to complete automated scoring. These parts are integrated into our automated essay scoring system where all of them run on a Linux machine with the CPU 2.0GHz Intel Xeno E5-2620 and 96G memory. In order to estimate the performance of incremental SVD, we use 15,776 test essays from MHK as the training set [10], and 1,000 test essays as the testing set.

When searching for the optimum batch size, we decompose the matrices firstly with the batch size from 1 to 10 at intervals of 1, and then from 100 to 1000 at intervals of 25. After the least time usage is confirmed, we refine the range to find out the optimum batch size more precisely. For the training set, the corresponding human scores (from 0 to 6 at intervals of 0.5) will be added to the essay vectors in order to train the support vector machine model. In our experiments, the tool we use for training is LIBSVM [39], and we use automatic training tools to find out the best parameters C and , which are 2048.0 and 0.0078125 respectively. Using the SVM model, we classify the testing set.

Note that the process of computing is unstable which leads to the fluctuation of the results, so in our experiments, we run each round for 10 times and use the average values.

B. Computing time

According to the training set, the TF-IDF is a 12,975-15,776 matrix, where the size of the word corpus confirmed by the training set is 12,975, and contains 15,776 essays. As previously stated, it is an impossible task for conventional SVD as the dataset is too big. In contrast, incremental method is far less time consuming. The computing results of incremental SVD are shown in Table I and Figure 3. In these experiments, we set the initial batch size from 10 to 1,000, at intervals of 10 and 100, and the initialized value and threshold value are both 100. As we know, when the batch size is 0, there is no difference between conventional and incremental SVD. But as the batch size increases, the running time of incremental SVD decreases sharply at first, and continues increasing gradually.

When we concentrate on the results where the curve shows the optimum batch size, from 200 to 400, as are shown in Table II and Figure 4, we can get our optimum

TABLE I.
THE RUNNING TIME OF INCREMENTAL SVD

Batch size	Time (s)
10	463.7294
20	255.4206
30	180.6784
40	148.7053
50	124.1791
60	108.4457
70	103.2142
80	92.754
90	86.0899
100	82.7572
200	69.81877
300	66.0840
400	68.7404
500	69.9703
600	79.8691
700	87.6521
800	94.3936
1000	105.9955

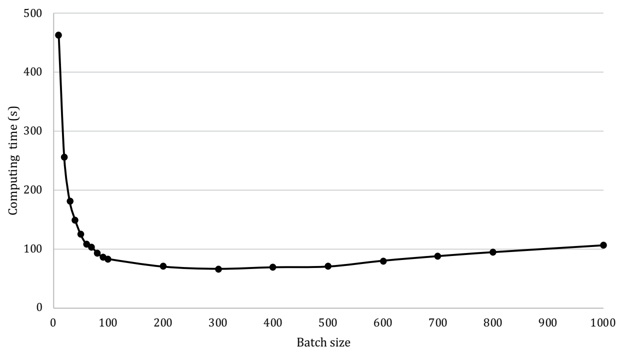

Figure 3. The running time of incremental SVD

batch size is about 320. Therefore, in the subsequent experiments the batch size is set to 320.

We compare conventional SVD with incremental SVD in Table III and Figure 5. In our experiments we increase the size of the training set, from 31,552 to 157,760, and it shows that when the size grows larger and larger, incremental SVD is far more efficient than conventional SVD. When the size grows to 110,432, the running time of conventional SVD is more than two hours, as is shown in Table III and Figure 5, so it is obvious that incremental SVD performs much better.

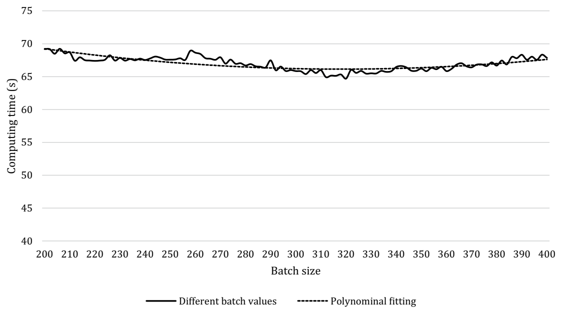

Figure 4. Optimum batch size

TABLE II.
OPTIMUM BATCH SIZE

Batch size	Time (s)
200	69.81877
210	68.83114
220	68.25197
230	68.29288
240	67.64322
250	67.90268
260	68.84589
270	68.03251
280	65.89831
290	66.77077
300	66.08409
310	65.95101
320	**65.23065**
330	66.52441
340	66.03742
350	66.58706
360	66.34916
370	67.18953
380	67.26448
390	68.7329
400	68.7404

TABLE III.
COMPARISON OF CONVENTIONAL SVD AND INCREMENTAL SVD

Size of the training set	Time (s)	
	incremental	conventional
31552	141.661	1111.324
47328	231.759	1692.691
63104	339.327	2617.137
78880	460.742	3858.184
94656	592.244	5466.019
110432	680.372	7929.217
126208	824.693	N\A
141984	978.254	N\A
157760	1145.381	N\A

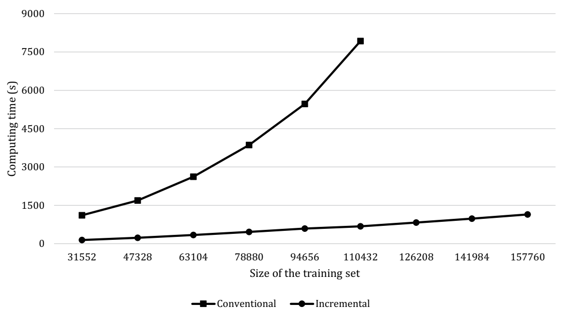

Figure 5. Comparison of conventional SVD and incremental SVD

TABLE IV.
OPTIMUM BATCH SIZES FOR DIFFERENT SIZES OF DATASETS

Size	**10000**	**20000**	**30000**	**40000**
Optimum batch size	296	308	320	348
Size	**50000**	**60000**	**70000**	**80000**
Optimum batch size	360	432	476	490

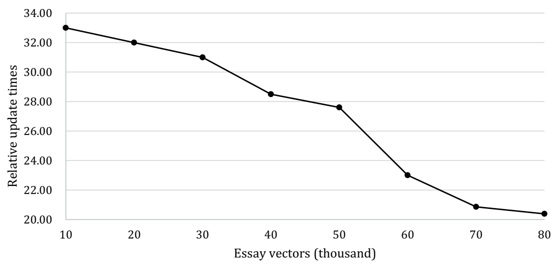

Figure 6. Relative update times

C. Relative update times

According to the size of the training set, the optimum batch size varies, and hence the update times. The advantage of incremental SVD, however, is obvious when facing large datasets. To make it clear, a series of experiments have been conducted, and the results are shown in Table IV.

In these experiments, we increase N from 10,000 to 80,000 at intervals of 10,000 with $n' = 100$. Fixing N and n', by performing various batch sizes on the decomposition and observing the least time consuming, we choose the *optimum batch sizes* according to different sizes of datasets in Table IV.

From Table IV, we can see that as the size of the dataset grows, the optimum batch size grows as well. For showing that the performance of the incremental method does not become worse, we compute *relative update times*.

The relative update times is used to observe the performance of the incremental method based on the size of the dataset and the optimum batch size:

$$relative\ update\ times = \frac{(N - n')}{optimum\ batch\ size} \times \frac{base}{N} \tag{11}$$

where N is the amount of essay vectors (the size of the original dataset matrix), and n' is the initialized value. The $base$ means the size of the dataset as the unit. For example, in our experiment, the $base$ is 10,000 since N starts growing at that number. In Formula (11), the first part of the multiplication is to calculate the update times, and the second part finishes the unification. The relative update times reflects the calculating ability of incremental SVD facing different sizes of datasets.

Figure 6 shows the experimental results about relative update times. From Figure 6, we can see that as N grows, relative update times declines, which means that the incremental method will perform better and more efficiently when the dataset grows larger.

D. Memory usage

In addition to computing time, economic memory usage is an another huge advantage of incremental SVD. Figure 7 shows the comparison of these two methods in memory usage as the size of the training set grows.

From Figure 7, we can see that the maximum memory usage of incremental method is only 492M and incremental method performs relatively stably. In contrast, conventional method uses much more memory and increases distinctly. In practical application, language tests usually produce a massive dataset, so, even given huge memory, the decomposition task of conventional SVD seems impossible, but it is viable for incremental SVD to perform it.

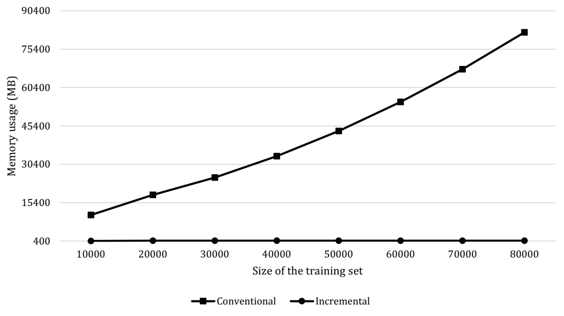

Figure 7. Comparison of conventional and incremental SVD in memory usage

TABLE V.
SCORING ACCURACY OF CONVENTIONAL AND INCREMENTAL SVD

	Incremental	Conventional
Size of testing set	1000	1000
Tolerable scorings	888	888
Intolerable scorings	112	112
Scoring accuracy	88.8 %	88.8%

E. Scoring performance

From the results above, it is obvious that incremental SVD is of huge advantages both in running time and memory usage. When it is used in LSA, what is important is that it will not reduce the *scoring accuracy*, which is calculated as follows:

$$\text{scoring accuracy} = \frac{\sum_{i=1}^{n} t(hs_i, ps_i)}{n} \tag{12}$$

where n is the size of the testing set, and hs_i and ps_i are the human scoring and the automated scoring of the i-th essay respectively. The function $t(hs_i,\, ps_i)$ is binary which can be defined as:

$$t(hs, ps) = \begin{cases} 1 & |hs - ps| \leq 1 \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

It is tolerable when the difference between human and automated scoring is less than 1. Table V shows a comparison of incremental and conventional methods in scoring accuracy and it verifies that incremental SVD is as effective as conventional method, and both their values of scoring accuracy are 88.8%, which is acceptable.

V. Conclusions and Future Works

Latent semantic analysis is of prevalence in the fields of information retrieval and natural language processing. Its primary algorithm, singular value decomposition, however, cannot deal with huge datasets. Our incremental method has a sharp contrast with traditional method when faced with huge datasets, using far less memory and time consuming. Experimental results also show that this method can be effectively used in automated essay scoring.

Our future works include the incremental algorithm and the performance of automated scoring. In order to get the optimum batch size, we need confirm the best threshold value and initialized value, which will influence the performance of the incremental algorithm. Secondly, it is quite important to improve the correlation between the human and the predicted scoring, for a wider application of automated essay scoring.

ACKNOWLEDGMENT

The authors are grateful to the anonymous referees for their valuable comments and suggestions.

REFERENCES

[1] H. Peng and Y. Yu, "Research on controlling central rating in net-based scoring of subjective test items," *China Examinations*, no. 6, pp. 3–9, 2013.

[2] A. Fazal, T. Dillon, and E. Chang, "Noise reduction in essay datasets for automated essay grading," in *On the Move to Meaningful Internet Systems: OTM 2011 Workshops*. Springer, 2011, pp. 484–493.

[3] R. A. Hardy, "Examining the costs of performance assessment," *Applied Measurement in Education*, vol. 8, no. 2, pp. 121–134, 1995.

[4] E. B. Page, "Computer grading of student prose, using modern concepts and software," *The Journal of experimental education*, vol. 62, no. 2, pp. 127–142, 1994.

[5] J. Burstein, K. Kukich, S. Wolff, J. Lu, and M. Chodorow, *Enriching automated essay scoring using discourse marking*. ERIC Clearinghouse, 2001.

[6] S. Valenti, F. Neri, and A. Cucchiarelli, "An overview of current research on automated essay grading," *Journal of Information Technology Education*, vol. 2, pp. 319–330, 2003.

[7] L. M. Rudner and T. Liang, "Automated essay scoring using bayes' theorem," *The Journal of Technology, Learning and Assessment*, vol. 1, no. 2, 2002.

[8] P. W. Foltz, D. Laham, and T. K. Landauer, "The intelligent essay assessor: Applications to educational technology," *Interactive Multimedia Electronic Journal of Computer-Enhanced Learning*, vol. 1, no. 2, 1999.

[9] B. Li and J.-M. Yao, "Automated essay scoring using multi-classifier fusion," in *Computing and Intelligent Systems*. Springer, 2011, pp. 151–157.

[10] H. Peng, "The minorities-oriented chinese level test," *China Examinations*, no. 10, pp. 57–59, 2005.

[11] D. Ke, X. Peng, Z. Zhao, Z. Chen, and J. Wang, "Word-level-based automated chinese essay scoring method," in *Proceedings of National Conference on Man-Machine Speech Communication*, Xi'an China, 2011, pp. 57–59.

[12] Y. Tonta and H. R. Darvish, "Diffusion of latent semantic analysis as a research tool: A social network analysis approach," *Journal of Informetrics*, vol. 4, no. 2, pp. 166–174, 2010.

[13] J. McInerney, A. Rogers, and N. R. Jennings, "Improving location prediction services for new users with probabilistic latent semantic analysis," in *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*. ACM, 2012, pp. 906–910.

[14] W. Wang and B. Yu, "Text categorization based on combination of modified back propagation neural network and latent semantic analysis," *Neural Computing and Applications*, vol. 18, no. 8, pp. 875–881, 2009.

[15] Y. Jin, Y. Gao, Y. Shi, L. Shang, R. Wang, and Y. Yang, "P2lsa and p2lsa+: Two paralleled probabilistic latent semantic analysis algorithms based on the mapreduce model," in *Intelligent Data Engineering and Automated Learning-IDEAL 2011*. Springer, 2011, pp. 385–393.

[16] S. Ismail, R. M. Othman, and S. Kasim, "Pairwise protein substring alignment with latent semantic analysis and support vector machines to detect remote protein homology," in *Ubiquitous Computing and Multimedia Applications*. Springer, 2011, pp. 526–546.

[17] L. Lin, B. Liu, X. Wang, X. Wang, and B. Tang, "Protein remote homology detection and fold recognition based on features extracted from frequency profiles," *Journal of Computers*, vol. 6, no. 2, pp. 321–328, 2011.

[18] S. Zhong, M. Shang, and Z. Deng, "A design of the inverted index based on web document comprehending," *Journal of Computers*, vol. 6, no. 4, pp. 664–670, 2011.

[19] L. Wang and Y. Wan, "Sentiment classification of documents based on latent semantic analysis," in *Advanced Research on Computer Education, Simulation and Modeling*. Springer, 2011, pp. 356–361.

[20] C. Liu, Y. Wang, F. Zheng, and D. Liu, "Using lsa and text segmentation to improve automatic chinese dialogue text summarization," *Journal of Zhejiang University Science A*, vol. 8, no. 1, pp. 79–87, 2007.

[21] J. Yeh, H. Ke, and W. Yang, "Chinese text summarization using a trainable summarizer and latent semantic analysis," in *Proceedings of the 5th international conference on Asian digital libraries (ICADL'02)*. Springer, 2002, pp. 76–87.

[22] A. Mesaros, T. Heittola, and A. Klapuri, "Latent semantic analysis in sound event detection," in *19th European Signal Processing Conference*, 2011, pp. 1307–1311.

[23] G. Gorrell, "Generalized hebbian algorithm for incremental singular value decomposition in natural language processing." in *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL-06)*, 2006, pp. 97–104.

[24] A. Atreya and C. Elkan, "Latent semantic indexing (lsi) fails for trec collections," *ACM SIGKDD Explorations Newsletter*, vol. 12, no. 2, pp. 5–10, 2011.

[25] Y. Li, "On incremental and robust subspace learning," *Pattern recognition*, vol. 37, no. 7, pp. 1509–1518, 2004.

[26] M. Brand, "Incremental singular value decomposition of uncertain data with missing values," in *Computer Vision ECCV 2002*. Springer, 2002, pp. 707–720.

[27] T. J. Chin, K. Schindler, and D. Suter, "Incremental kernel svd for face recognition with image sets," in *7th International Conference on Automatic Face and Gesture Recognition. FGR*. IEEE, 2006, pp. 461–466.

[28] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Incremental singular value decomposition algorithms for highly scalable recommender systems," in *5th International Conference on Computer and Information Science*, 2002, pp. 27–28.

[29] M. Brand, "Fast online svd revisions for lightweight recommender systems," in *SIAM International Conference on Data Mining*, 2003, pp. 37–46.

[30] Y. Cao and C. Yang, "Automated chinese essay scoring with latent semantic analysis," *Examinations Research*, vol. 3, no. 1, pp. 63–71, 2007.

[31] Y. Li, "Automated essay scoring for testing chinese as a second language," Ph.D. dissertation, Beijing Language and Culture University, 2006.

[32] M. M. Islam and A. Hoque, "Automated essay scoring using generalized latent semantic analysis," *Journal of Computers*, vol. 7, no. 3, pp. 616–626, 2012.

[33] F. Wild, C. Stahl, G. Stermsek, and G. Neumann, "Parameters driving effectiveness of automated essay scoring with lsa," in *9th Internaional Computer-Assisted Assessment Conference*, 2005, pp. 485–494.

[34] C. Chang, "A pilot study on automatic chinese spelling error correction," *Communication of COLIPS*, vol. 4, no. 2, pp. 143–149, 1994.

[35] J. Ma, Y. Zhang, T. Liu, and S. Li, "Detecting chinese text errors based on trigram and dependency parsing," *Journal of The China Society For Scientific and Technical Information*, vol. 6, p. 014, 2004.

[36] S. Hao, Z. Gao, M. Zhang, Y. Xu, H. Peng, K. Su, and D. Ke, "Automated error detection and correction of chinese characters in written essays based on weighted finite-state transducer," in *12th International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2013, pp. 763–767.

[37] M. Sahami and T. D. Heilman, "A web-based kernel function for measuring the similarity of short text snippets," in *Proceedings of the 15th international conference on World Wide Web*. ACM, 2006, pp. 377–386.

[38] X. Peng and Y. Wang, "Cch-based geometric algorithms for svm and applications," *Applied Mathematics and Mechanics*, vol. 30, no. 1, pp. 89–100, 2009.

[39] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, pp. 27:1–27:27, 2011, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

Mingqing Zhang was born in Shandong, China. He studied in Beijing Forestry University since 2011.

He joined Institute of Artificial Intelligence in Beijing Forestry University in 2012 as a research assistant. His publications include: "Automated Error Detection and Correction of Chinese Characters in Written Essays Based on Weighted Finite-State Transducer", International Conference on Document Analysis and Recognition, ICDAR2013. His major research interests include "Automated Essay Scoring" and "Speech Recognition".

Shudong Hao was born in Beijing, China. He studied in Beijing Forestry University since 2011.

He joined Institute of Artificial Intelligence in Beijing Forestry University in 2012 as a research assistant. His publications include: "Automated Error Detection and Correction of Chinese Characters in Written Essays Based on Weighted Finite-State Transducer", International Conference on Document Analysis and Recognition, ICDAR2013. His major research interests include "Machine Learning" and "Natural Language Processing".

Yanyan Xu was born in China in 1981. She obtained the bachelor degree with the major field computer science and technology and the master degree with the major field computer software and theory from Sun Yetsen University in Guangzhou in 2003 and 2005 respectively, and then she majored in computer software and theory and obtained the PhD from Institute of Software, Chinese Academy of Science in 2009 in Beijing.

She joined Beijing Forestry University in 2009 as a lecturer and promoted as an associate professor in 2013. She has published over 10 peer-refereed papers in quality journals and conferences. Her publications include: "Solving Difficult SAT Problems by Using OBDDs and Greedy Clique Decomposition", International Frontiers of Algorithms Workshop, Springer, FAW2012; "CacBDD: A BDD Package with Dynamic Cache Management", the 25th International Conference on Computer Aided Verification, CAV2013; and "Automated Error Detection and Correction of Chinese Characters in Written Essays Based on Weighted Finite-State Transducer", ICDAR2013, and so on. Her research interests include "Formal methods", "Artificial Intelligence" and "Algorithm Design and Analysis".

Prof. Xu has been the director of Institute of Artificial Intelligence in Beijing Forestry University since 2012 and currently she is supervising 6 students. She got the academic professional excellence award from Beijing Forestry University in 2012.

Dengfeng Ke was born in GuangDong, China in 1980. He obtained the bachelor degree with the major field computer science and technology and the master degree with the major field computer software and theory from Sun Yetsen University in Guangzhou in 2003 and 2005 respectively, and then he majored in pattern recognition and intelligence system and obtained the PhD from Institute of Automation, Chinese Academy of Science in 2009 in Beijing.

He joined Institute of Automation, Chinese Academiy of Science in 2009 as a research assistant and promoted as an associate professor in 2013. He has published over 20 peer-refereed papers in quality journals and conferences. His publications include: "Automated Chinese essay scoring using vector space models", IUCS 2010; "Automated essay scoring based on finite state transducer: towards ASR transcription of oral English speech", ACL2012; and "Automated Error Detection and Correction of Chinese Characters in Written Essays Based on Weighted Finite-State Transducer", ICDAR2013, and so on. His research interests include "Speech Recognition", "Artificial Intelligence" and "Algorithm Design and Analysis".

Prof. Ke has been invited to worked in Institute of Digital Media in Singapore since 2013.

Hengli Peng was born in China in 1964. He obtained his bachelor degree with the major field in Philosophy from Shandong University in Jinan in 1987, and the master degree in Chinese Philology from Beijing Language and Culture University in 2003.

He has worked in Beijing Language and Culture University since 1987,and has engaged on the development and management of Chinese Proficiency Test (HSK) for several years. Now he is an associate professor and the director of Institute of Educational Measurement in Beijing Language and Culture University (BLCU-IEM). He is also the Project Leader of the Chinese Proficiency Test for Minorities (MHK) Program. His publications include The Second Series of Collected Works for Test Research (Beijing, China: Economic Science Press, 2004), Suggested Improvements for AAT Based on Statistic Analysis (In The Sixth Collected Works for Test Research, 2011), Research on Controlling Central Rating in Net-based Scoring of Subjective Test Items (Beijing, China: China Examinations, 2013) and so on. His current research interest covers computer-automated scoring of subjective items, detection of test cheating and error controlling for the scoring of subjective items.

Prof. Peng has taken part in several tests development, including Chinese Proficiency Test (HSK), Chinese Proficiency Test for Minorities (MHK), Test for Certifying the Ability to Teach Chinese as a Foreign Language (MJK), Chinese Character Proficiency Test (HZC), Mongol Proficiency Test, and Uighur language Proficiency Test, for the past several years.