Yanyan Xu*, Dengfeng Ke and Kaile Su

Contextualized Latent Semantic Indexing: A New Approach to Automated Chinese Essay Scoring

DOI 10.1515/jisys-2015-0048
Received May 15, 2015; previously published online March 1, 2016.

Abstract: The writing part in Chinese language tests is badly in need of a mature automated essay scoring system. In this paper, we propose a new approach applied to automated Chinese essay scoring (ACES), called contextualized latent semantic indexing (CLSI), of which Genuine CLSI and Modified CLSI are two versions. The *n*-gram language model and the weighted finite-state transducer (WFST), two critical components, are used to extract context information in our ACES system. Not only does CLSI improve conventional latent semantic indexing (LSI), but bridges the gap between latent semantics and their context information, which is absent in LSI. Moreover, CLSI can score essays from the perspectives of language fluency and contents, and address the local overrating and underrating problems caused by LSI. Experimental results show that CLSI outperforms LSI, Regularized LSI, and latent Dirichlet allocation in many aspects, and thus, proves to be an effective approach.

Keywords: Automated Chinese essay scoring, latent semantic indexing, *n*-gram language model, weighted finite-state transducer, natural language processing.

1 Introduction

Writing is important in language tests. Generally, there are two kinds of writing. One requires test takers to write an essay using their native languages, such as the writing part in the National Higher Education Entrance Examination in mainland China. The other kind of writing requires test takers to write an essay using foreign languages, such as the Minorities-Oriented Chinese Level Test (MHK) [25], so this kind of writing is relatively easier. Figure 1 shows these two kinds of writing. In this paper, we focus on the second kind of writing.

Traditionally, one essay needs to be graded by at least two human raters who are trained by studying the scoring criteria and sample essays with different scores. When the number of test takers grows bigger, this method is obviously time consuming, labor consuming, and far less accurate caused by subjective opinions and rater fatigue. Automated essay scoring (AES), a powerful intelligent system, however, can resolve these problems. Therefore, automated Chinese essay scoring (ACES) necessarily becomes an indispensable part of computer-aided education, from classrooms to large-scale tests.

***Corresponding author: Yanyan Xu,** School of Information Science and Technology, Beijing Forestry University, Beijing, China, e-mail: xuyanyan@bjfu.edu.cn
Dengfeng Ke: Institute of Automation, Chinese Academy of Sciences, Beijing, China
Kaile Su: Institute for Integrated and Intelligent Systems, Griffith University, Brisbane, Australia

根据上面的图片，写一篇不少于800字的作文。文体不限（诗歌除外）。
Write an essay inspired by the picture above. There are no limitations to genres except poems. You should write no less than 800 words.

The National Higher Education Entrance Examination

一位老和尚为了给自己选一个接班人，对他的两个徒弟说："去找一片你们最满意的树叶回来。"结果，第一个徒弟很快就回来了，递给师傅一片树叶，说："虽不完美，但却是我看到的最好的树叶。"第二个徒弟很晚才归，两手空空，对师傅说："我见到的树叶很多，但没有一片是完美的，所以没有一片是我最满意的。"老和尚看着第一个徒弟满意地笑了。

请根据上面的内容，结合自己的认识，写一篇读后感。自定题目，全文不少于350字。
Please write an essay, rephrasing the fable and expressing your thoughts according to your experiences. You should write no less than 350 words.

The Minorities-Oriented Chinese Level Test (MHK)

Figure 1: Examples of Two Kinds of Writing.

1.1 Background

AES originated from related research on English [29]. Earlier in the 1970s, Ellis Page developed the Project Essay Grader (PEG) for large-scale essay scoring upon the request of the College Board, which heralded the dawn of the field of scoring by intelligent systems and computers [24]. Since then, many studies have been conducted to improve the effectiveness and accuracy of AES, along with the emergence of commercial software in the education market and applied to industry. AES not only assists testers in assessing the proficiency of test takers, but also helps faculties and teachers to get knowledge about students.

PEG utilizes surface information to score essays, such as the lengths of passages, the number of words and misspellings in an essay, etc., but ignores deeper semantic structures, which makes PEG vulnerable: students can use more complex words, and write longer passages without any logical contents in essays to trick this system and get high scores. Intelligent Essay Assessor utilizes latent semantic indexing (LSI) to avoid the deficiencies of PEG and focuses on semantics instead of surface features [18, 19]. Educational Testing Service applies its E-rater to the writing section in the Test of English as a Foreign Language and analytical writing in Graduate Record Examination [1, 3, 27]. IntelliMetric, the first automated scoring system based on artificial intelligence combining natural language processing (NLP) with statistics, was developed by Vantage Learning [11, 31, 32]. Based on statistics and the Bayesian theorem, Lawrence M. Rudner developed the Bayesian Essay Test Scoring sYstem (BETSY) [28]. Similarly, BETSY uses lots of samples to train models, and evaluates database statistics and determines the classification accuracy.

Because automated English scoring emerged earlier and it is relatively easier to process English texts than Chinese ones, automated scoring systems for English have been applied widely. ACES, which started later, however, is left behind.

1.2 Related Work

Recently, ACES has attracted much more attention. Especially, when the scale of the MHK test grows larger, it is more necessary to replace conventional human scoring by AES. Unfortunately, there still lacks such a mature architecture caused by difficulties in processing Chinese texts.

For Chinese text processing, the first hurdle is segmentation. Unlike English or other Western languages, Chinese does not provide inter-word delimiters within a sentence, leading to one sentence possibly having different meanings according to different segmentations. Currently, there are several methods focusing on Chinese word segmentation, among which the integrated generative and discriminative character-based model proposed in Ref. [33] is the most comprehensive one. The methodology, however, needs to be adapted to ACES.

The second hurdle for ACES is feature extraction, which also exists in AES for any other language. Inspired by the studies for English, similar methods have been researched from various angles [16, 20, 26]. For instance, topic feature extraction methods using topic modeling strategies like regularized LSI (RLSI) [14, 34] and latent Dirichlet allocation (LDA) [2, 14] are novel ones. Other methods have also been tested; however, their results are relatively rare [4]. Semantic feature extraction using LSI [5, 36] explores the contents reflected by words, which is the most classical method. Currently, there are several LSI-based methods applied to AES [5, 15], which are as follows:

(i) *Conventional LSI*

LSI was proposed by Scott Deerwester et al. in Ref. [9], which starts with a term-document matrix $\mathbf{D}$ where every row $\mathbf{d}_{i,:}$ is a word and every column $\mathbf{d}_{:,j}$ is an essay. Typically, $\mathbf{D}$ is weighted by the term frequency-inverse document frequency (TF-IDF) method [23]. Next, LSI uses a mathematical strategy called singular value decomposition (SVD) to construct a semantic space by

$$\mathbf{D} \approx \mathbf{D}^* = \mathbf{U}^* \mathbf{\Sigma}^* \mathbf{V}^{\top *},$$

where $\mathbf{\Sigma}^*$ is a diagonal matrix with $\sigma_{1,1} \geq \sigma_{2,2} \geq \ldots \geq \sigma_{k,k}$ as the diagonal elements, and $\mathbf{D}^*$ is the least square best-fit approximation of $\mathbf{D}$. Finally, after transformation, we can train a scoring model.

According to the essays in MHK tests, sometimes there are few distinct differences between a higher-score essay and a lower-score essay from the perspective of word usage and occurrence merely. The reason may be that the essay with a higher score uses the words in an appropriate order and context, while the essay with a lower score does not. If two essays have identical words with different orders, they will get the same features when applying LSI to them, as Figure 2 shows. Consequently, they will get the same automated scores, which causes the so-called local overrating problem when low-score essays cannot be recognized, or the local underrating problem when high-score essays cannot be recognized.

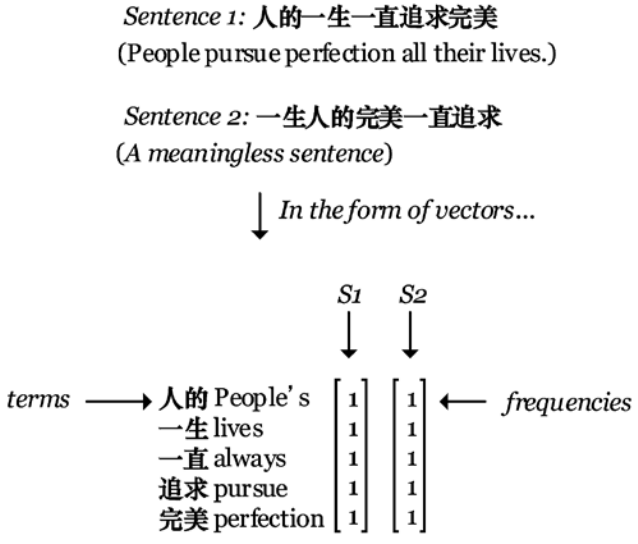

Figure 2: An Example Showing That a Meaningless Sentence (*Sentence*$_2$) Shares the Same Occurrence Information (Vectors S_1 and S_2) with a Normal Sentence (*Sentence*$_1$).

(ii) *Generalized latent semantic analysis*
Considering the lacking context information in conventional LSI, a method called generalized latent semantic analysis has been proposed [15]. By using *n*-gram as the atomic element of each row, it produces a very huge and sparse matrix **D**. For flexible languages such as Chinese, it is computationally prohibitive.

(iii) *RLSI*
RLSI is a recently proposed method in Ref. [34] in topic modeling and information retrieval, including the batch version and the online version. The first study of scoring Chinese essays from the topic perspective has been used in Ref. [14], with the batch version of RLSI. The target of RLSI is to decompose a term-document matrix **D** into a term-topic matrix **U** and a topic-document matrix **V**, minimizing the Frobenius norm $||\mathbf{D}-\mathbf{UV}||_F^2$ with ℓ_1-norm regularization on **U** and ℓ_2-norm regularization on **V**, which can be described as

$$\min_{\mathrm{U},\mathrm{V}}(||\mathbf{D}-\mathbf{UV}||_F^2+\lambda_1||\mathbf{U}||_1+\lambda_2||\mathbf{V}||_F^2), \tag{1}$$

where **U** reflects the relations between documents and latent topics. In Ref. [14], RLSI is combined with support vector machine (SVM) to score Chinese essays.
RLSI is a useful method to discover the topics in a collection of test essays and is effective to score essays from the topic perspective; however, a comprehensive scoring system requires more perspective.

1.3 Main Contributions

Faced with the difficulties of ACES and the limitation of LSI, we propose contextualized latent semantic indexing (CLSI), which has these contributions: integrating Chinese text segmentation and context information extraction, scoring essays from the perspective of language fluency [25], and addressing the overrating and underrating problems. As mentioned earlier, LSI may cause overrating and underrating problems. In language tests, low-score essays weigh much more than ordinary essays, because they can be used to discover the latent learning problems of students and test takers. If a system cannot recognize low-score essays, it will lose important test information, and hurt the fairness of the tests to some extent.

Instead of the traditional term-document matrix recording the occurrence information, CLSI uses a probabilistic matrix reflecting the context information from an *n*-gram language model, solving the problems of both "What are these words?" and "How these words are used?." More specifically, we propose two versions of CLSI: Genuine CLSI, which uses originally written essays, and Modified CLSI, which modifies possibly erroneous characters in essays. Experimental results show that both the Genuine and Modified versions are effective in feature extraction and AES.

The rest of the paper is organized as follows. In Section 2, we outline our methods and explain the terms used in this paper. Then, we thoroughly introduce all parts from Sections 3–5, including the language model construction, CLSI, and the SVM-based scoring model. Section 6 shows the experimental results and makes some discussions. Finally, in Section 7, we conclude this paper and point to our future work.

2 An Overview of ACES Using CLSI

A common method for AES includes three steps: text preprocessing, feature extracting, and scoring model training. According to the characteristics of Chinese tests, we propose the following framework: language model constructing, context information extracting, and scoring model training, as illustrated in Figure 3.

In language model constructing, we use a large corpus to build an *n*-gram language model, and then convert it to a weighted finite-state transducer (WFST). In context information extracting, we use WFST to extract context information. If modified CLSI is used, WFST will use a confusing-character table including

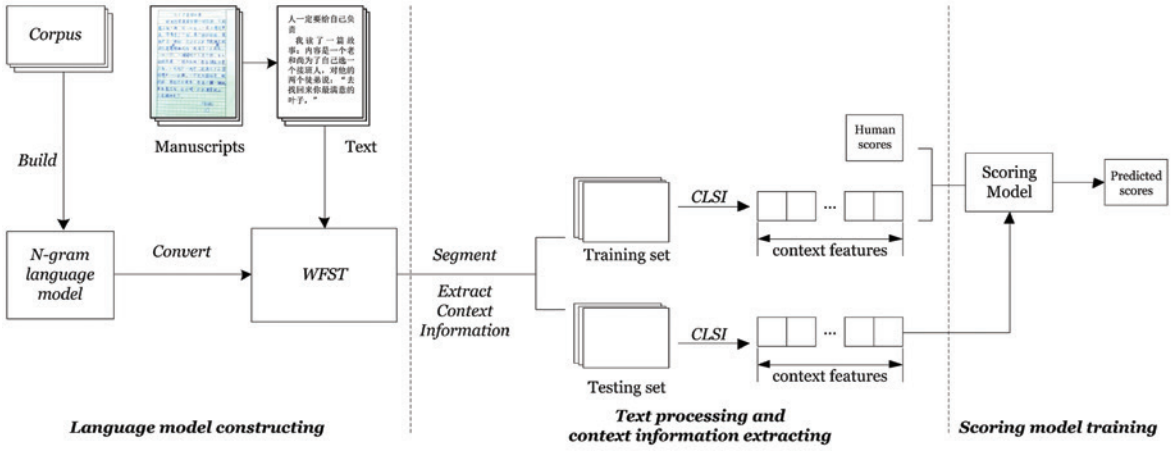

Figure 3: The Framework of Automated Chinese Essay Scoring Using CLSI.

frequently confusing characters to detect and correct erroneous characters. Then, essays will be organized as a matrix and transformed under the contextualized-semantic space. The essays, finally, will be sent to SVM to train the scoring model.

In order to be clearer, we introduce some terms and notations as preliminaries for the rest of this paper:

(i) *Essay collection*: When the written essays in the electronic text form are obtained, they are organized as a data set called *essay collection* $\mathcal{D}$. We denote $|\mathcal{D}|$ as the size of $\mathcal{D}$.

(ii) *Essay word list*: We define an ordered word list $\mathcal{V}$ containing the words that appeared in an essay, and $|\mathcal{V}|$ is the size of $\mathcal{V}$. The elements w_i $(i = 1, \ldots, |\mathcal{V}|)$ in $\mathcal{V}$ are ordered by the sequence they occur in the essay, and the list may have identical words.

(iii) *Collection word set*: We use $\mathcal{V}^\dagger$ to define a set containing all the words in an essay collection, namely, $\mathcal{V}^\dagger = \bigcup_{i=1}^{|\mathcal{D}|} \mathcal{V}_i$. Similarly, $|\mathcal{V}|^\dagger$ is the size of the $\mathcal{V}^\dagger$. There are no identical elements in $\mathcal{V}^\dagger$.

(iv) *Essay vector*: If we have an essay with a word list $\mathcal{V}$, then it can be presented as a vector $\mathbf{d} = [d_1, d_2, \ldots, d_M]^\top \in \mathbb{R}^M$ where $M = |\mathcal{V}|$. If d_i is the term frequency of the i^{th} term in the essay (like that in LSI) or context information (like that in CLSI), this vector is called a term-based essay vector.

(v) *Essay matrix*: When essays in an essay collection have been processed into essay word lists and presented as essay vectors, they will be arrayed together to a matrix $\mathbf{D} = [\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_N] \in \mathbb{R}^{M \times N}$, where $M = |\mathcal{V}^\dagger|$ and $N = |\mathcal{D}|$.

3 Language Model Construction

The first part of ACES using CLSI is to build an *n*-gram language model from a large-scale corpus, which will be used to construct a WFST to recognize specific word sequences of sentences in essays and extract context information.

3.1 *n*-Gram Language Model

In NLP, an *n*-gram language model is a representative statistical inference strategy. Assume that a sentence S is a sequence of words:

$$S = w_1 w_2 \ldots w_n. \tag{2}$$

Then, predicting the word w_n can be formularized as a probability:

$$P(w_n) = P(w_n | w_1, \ldots, w_{n-1}). \tag{3}$$

Consequently, predicting a sentence S can be formularized as

$$P(S) = \prod_{m=1}^{n} P(w_m) = P(w_1) P(w_2 | w_1) \ldots P(w_n | w_1, \ldots, w_{n-1}). \tag{4}$$

A more common method is to use the logarithm of $P(S)$, denoted as $LP(S)$.

Formally, it is called a uni-gram when there is no dependency between the current word and other words. When the first-order Markov assumption is applied, namely, that the current word depends on one previous word, it is called a bi-gram. Similarly, we call it a tri-gram when the second-order Markov assumption is applied where the current word depends on two previous words. In this paper, we use the tri-gram language model, including tri-grams, bi-grams, and uni-grams. This model can be built on any large-scale corpus. In our study, we use a word list including 47,500 words, which are used as uni-grams to train an *n*-gram language model by the SRI Language Modeling Toolkit (`SRILM`) (http://www.speech.sri.com/projects/srilm/) [30], a very successful tool in speech recognition. During the training, we use common smoothing methods, including the linear interpolation and Kneser–Ney discount [7]. According to different corpora, some words are pruned, so the sizes of uni-grams are slightly different. As it is not the focus of this paper, we omit these details.

3.2 Conversion from an *n*-Gram Language Model to WFST

A finite-state transducer (FST) is a finite automaton whose transitions are presented as arcs with input and output labels on them [22]. When designated with probabilities on arcs, it becomes a WFST. In this paper, for language processing, WFST is adapted to a directed graph $G = (S, A_{forward}, A_{backoff})$, where S, $A_{forward}$, and $A_{backoff}$ denote *States*, *Forward Arcs*, and *Backoff Arcs*, respectively.

States: We use ϵ (the epsilon state) as the very start of the proposed WFST. Each state S_i is defined as follows:

$$S_i = \{arc_0, arc_1, arc_2 \ldots arc_n\},$$

where these *arcs* are out-edges of S_i (except arc_0). In practice, because every state represents an *n*-gram with its lower order $(n-1)$-gram, we use arc_0 as the backoff arc whose probability is the backoff coefficient from the high order *n*-gram to its lower order (uni-gram to ϵ). ϵ is the starting state of a complete WFST, from which the decoding process starts. Additionally, each state can be regarded as a breakpoint for a segmentation and a start point for the remaining segmentation when decoding in a WFST.

Forward arcs: Each forward arc A_i represents a word connecting two *n*-grams with a conditional probability of the word. That is,

$$A_i = \{S_{in}, S_{out}, word, probability\},$$

where S_{in} records the previous state and S_{out} indicates the next state. The *word* corresponds to the word of a term in an *n*-gram language model and the *probability* is the probability from the *n*-gram language model.

Backoff arcs: Each state has one backoff arc, namely arc_0. By passing through this arc, the decoding process moves from a higher-order *n*-gram state to a lower-order $(n-1)$-gram state. The structure of the backoff arc is similar to the forward arc, except that the *word* is empty and the *probability* is the backoff coefficient.

Figure 4 shows an example of a conversion from an *n*-gram language model to WFST. We see that a forward arc can start from the state {今天} to the state {今天,天气} with the corresponding *probability* on it and the *word* ""天气"." In addition to forward arcs, backoff arcs in reverse directions are from higher-order states to lower-order ones. For more specific discussions, please refer to our previous studies [12, 13].

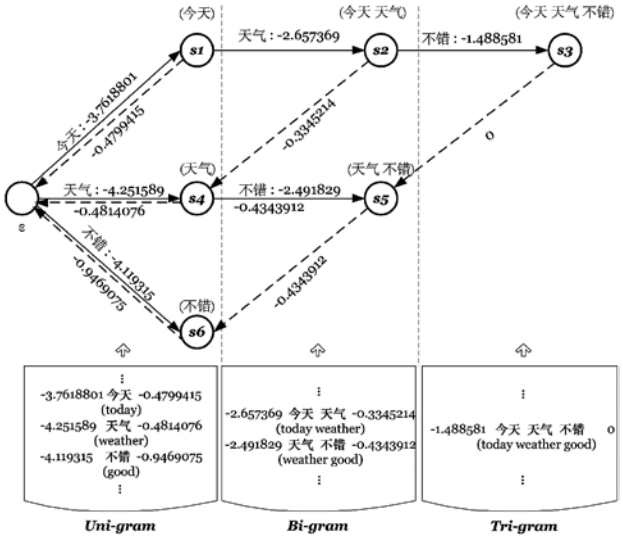

Figure 4: Conversion from an *n*-Gram Language Model to WFST.
We only show a small part of the whole WFST. In this figure, solid lines are forward arcs and dash lines are backoff ones. ϵ is the starting state.

4 CLSI

The CLSI methods proposed in this paper, both the Genuine version and the Modified one, consist of three parts: context information extracting, constructing contextualized-semantic space, and presenting the essays under the space. For the first part, Genuine CLSI and Modified CLSI perform differently, so we introduce them in Sections 4.1 and 4.2, respectively. For the remaining two parts, they perform in the same way, so we use Sections 4.3 and 4.4 to describe these two parts.

4.1 Genuine CLSI

Given an essay collection $\mathcal{D}$, WFST is used to recognize all essay word lists, and present each essay using a term-based essay vector $\mathbf{d}$. After that, CLSI is used to extract each word with its context information, and thus finish the feature extraction.

CLSI has an advantage that it extracts context information along with segmenting. As we know, a sentence may have different segmentations, and hence different context information. Our objective is to recognize the most reasonable sequence in an essay and extract its context information as accurately as possible, that is, to find out a word list $\mathcal{V}=\{w_i\}$ in the *n*-gram language model with the maximum probability, which can be formulated as

$$\hat{S}=\arg\max_{\{w_i\}} LP(S). \tag{5}$$

This task is a search problem in WFST whose goal is to find an optimal path from the starting state ϵ to a certain state. The intuition behind this method is obvious; that is, the more reasonable a sentence S is, the higher its probability $LP(S)$ is.

Given a specific context, context information is used to decide whether a word appears in a reasonable way. The *n*-gram language model provides a general standard, constructed from a large corpus with probabilities, and judging whether a sentence is understandable or not, which is helpful for detecting fluency and coherence among words. In Figure 2, there is no explicit information about how these words are organized in the two sentences. However, if we consider the processing in WFST and the *n*-gram language model, and replace term frequency or TF-IDF with probabilities, we will get a different result as Figure 5 shows. A simple

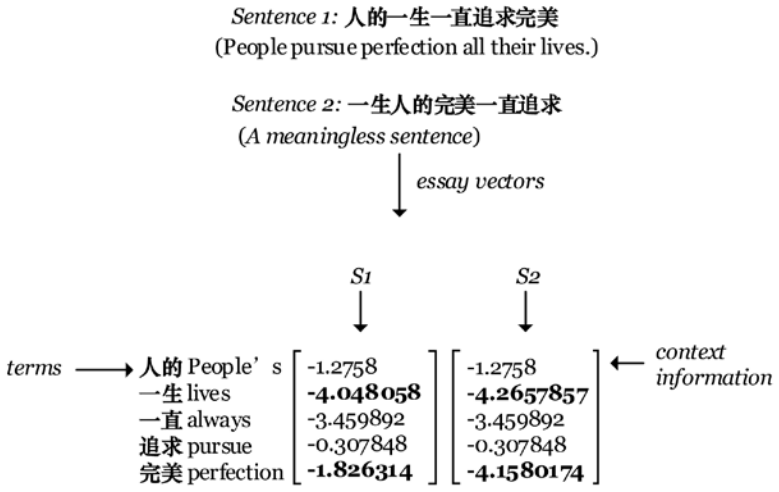

Figure 5: Context Information Applied to the Example in Figure 2.
When we consider context information extraction using WFST and the *n*-gram language model, we see that the two sentences are not identical anymore.

calculation, like summation, can make the difference: $PL(S1) > PL(S2)$, which makes sense because Sentence 2 is meaningless with an unreasonable sequence of words.

Context information provides an important indicator of scoring essays. According to our assumption about test essays, Sentence 1 may appear in good or excellent essays, whereas Sentence 2 may appear in poor essays. Thus, CLSI is better than LSI because it can make more obvious difference between poor essays and good ones. Figure 6 illustrates how to recognize a word list in a sentence using WFST, and Algorithms 1–4 show the details of Genuine CLSI.

In WFST, each possible solution is represented as a candidate, consisting of a segmented string (*segmented*), an unsegmented string (*unsegmented*), a log probability of the segmented string (*probability*), a current state (*state*), a word list ($\mathcal{V}$), and an essay vector (**d**). In Algorithm 1, to recognize a word in a sentence will pass a forward arc according to the word on the arc (*arc.word*), which is a step of context information extraction. A *candidate set* is used to define the set of promising solutions. As there is more than one candidate caused by different paths, we use a *beam width* to maintain the size of the candidate set to avoid unnecessary expansions. We sort the members in the candidate set in descending order, and prune the set during each step of segmenting, according to the members' probabilities.

In each iteration, we first pass each candidate S through all of its backoff arcs, arriving at its corresponding lower-order states (Algorithm 2). For example, if one candidate is stopping on a tri-gram state, we pass it to its corresponding bi-gram state, and then to the uni-gram state, and finally arrive at ϵ. Thus, three new candidates $S_i^*(i=0,\ 1,\ 2)$ will be generated whose current states are the bi-gram state, the uni-gram state, and ϵ, respectively, and corresponding log probabilities on the backoff arcs will be added. Although no segmentation happens, we still extract some important context information by using the statement "`Apply a certain Punishment Rule,`" in Algorithm 2, which will be explained later.

Next, in each iteration, we check every arc starting from the current state of the candidate S (Algorithm 3). Note that in line 16 in Algorithm 1, we use the statement "`Judge Passable Forward Arcs in Genuine/Modified CLSI,`" which means in the Genuine version, it will call the function in Algorithm 3. If the unsegmented part of S starts with the word of the forward arc *arc.word*, then we pass this arc, and transduce the state to its arriving state (Algorithm 4). The word on the arc will be added to the word set $\mathcal{V}$, and the corresponding word in the unsegmented part will be eliminated. Simultaneously, we extract the context information $arc_i.probability$ from the arc and append it to the tail of the essay vector **d** (note that we use $\cup$ to denote the appending).

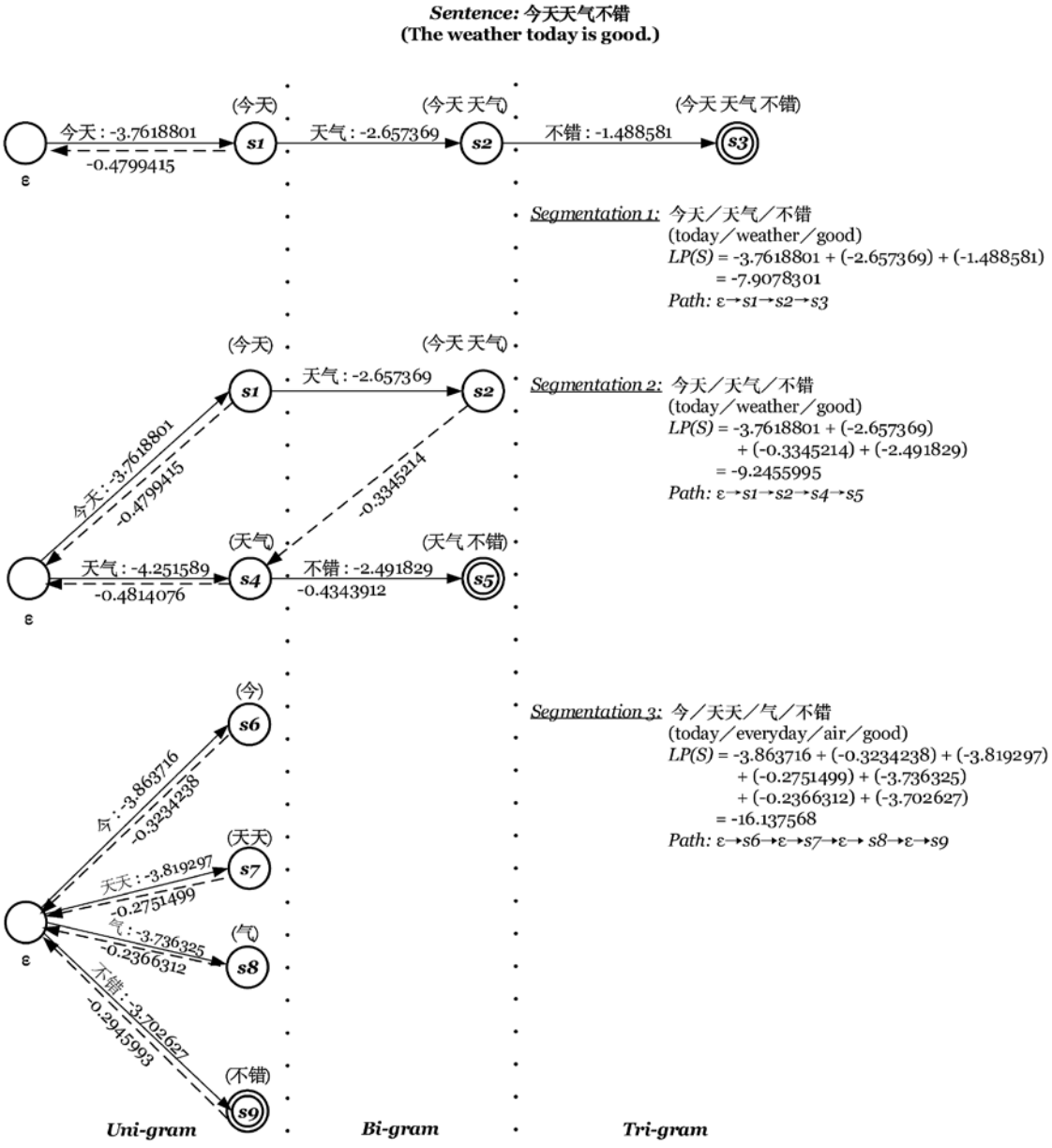

Figure 6: An Example Showing How to Recognize a Word List in a Sentence Using WFST.
For clarity, we separate the WFST into three parts, though actually they are in the same WFST.

In WFST, backoff arcs are crucial, because usually when a sequence w_i, w_{i+1} is unreasonable, it needs to backtrack from the higher-order state $\{w_i\}$ to the lower-order state. Therefore, adding a backoff coefficient lowers the probability, meaning that this sequence may be meaningless. However, a problem is encountered, that is, of which word, w_i or w_{i+1}, it is supposed to add the backoff coefficient to the context information. There exist three assumptions. First, we assume that w_i is correct but it is unreasonable to observe w_{i+1} after w_i. Second, we assume that w_{i+1} is correct but it is unreasonable to observe w_i before w_{i+1}. Third, we do not introduce backoff coefficients to the essay vector **d**. Based on these assumptions, we give three Punishment Rules, and Figure 7 can help understand this procedure.

– Punishment Rule 1: Add the backoff coefficient to the first word, meaning that the first word is not appropriate while the second is.

Algorithm 1: Context Information Extraction Using WFST in CLSI.

Input: Unsegmented Chinese sentence S;
Output: Essay vector $\mathbf{d}$;
$S.state \leftarrow \epsilon$;
$S.\mathcal{V} \leftarrow \emptyset$;
the candidate set $C \leftarrow \emptyset$;
$C \leftarrow C \cup \{S\}$;
repeat
  for *each* $S_i \in C$ *with the unsegmented part* **do**
    $C^* \leftarrow \emptyset$;
    for *all the backoff arcs to* ϵ **do**
      $S^* \leftarrow$ `Pass Backoff Arc` $(S_i$, backoff arc $arc_i)$;
      $C^* \leftarrow C^* \cup \{S^*\}$;
      if $|C^*| >$ *the beam width* **then**
        prune C^*;
      end
    end
    for *all the forward arcs* arc_i *starting from* $S_i.state$ **do**
      if `Judge Passable Forward Arcs in Genuine/Modified CLSI` $(S_i.unsegmented, arc_i.word)$ == *TRUE* **then**
        $S^* \leftarrow$ `Pass Forward Arc` (S_i, arc_i);
        $C^* \leftarrow C^* \cup \{S^*\}$;
        if $|C^*| >$ *the beam width* **then**
          prune C^*;
        end
      end
    end
  end
  for *each* $S_i \in C$ *that has been segmented completely* **do**
    $C^* \leftarrow C^* \cup \{S_i\}$;
    if $|C^*| >$ *the beam width* **then**
      prune C^*;
    end
  end
until $\forall S_i (S_i \in C \wedge S_i$ *has been completely processed*$)$;
$C \leftarrow C^*$;
Sort C according to $S_i.probability$ in descending order;
return *essay vector* $S_0.\mathbf{d}$

Algorithm 2: Pass Backoff Arc.

Input: S_i and backoff arc arc_i
Output: S^*
$S^* \leftarrow S_i$;
$S^*.state \leftarrow$ backoff arriving state;
$S^*.probability$ += *backoff coefficient*;
`Apply a Punishment Rule`;
return S^*

- Punishment Rule 2: Add the backoff coefficient to the second word, meaning that the second word is not appropriate while the first is.
- Punishment Rule 3: Do not add any backoff coefficient.

In practice, we will choose a Punishment Rule that performs best. In Algorithm 2, if we use Punishment Rule 1, we will add the backoff coefficient to the last dimension of the essay vector **d**; if we use Punishment Rule 2, we will append the backoff coefficient to **d** in advance. Thus, when the next word is recognized, its *arc.probability* will be added to this dimension; applying Punishment Rule 3 means skipping the statement "`Apply a certain Punishment Rule`."

Algorithm 3: Judge Passable Forward Arcs in Genuine CLSI

Input: S_i.*unsegmented and* arc_i.*word*
Output: TRUE or FALSE
1 **if** S_i.*unsegmented starts with* arc_i.*word* **then**
2 **return** *TRUE*
3 **end**
4 **return** *FALSE*

Algorithm 4: Pass Forward Arc

Input: S_i *and* arc_i
Output: S_i
1 $S^* \leftarrow S_i$;
2 S^*.*unsegmented* $-=$ arc_i.*word*;
3 $S^*.\mathcal{V} \leftarrow S^*.\mathcal{V} \cup \{arc_i.word\}$;
4 $S^*.\mathbf{d} \leftarrow S^*.\mathbf{d} \cup arc_i.probability$;
5 S^*.*probability* $+=$ arc_i.*probability*;
6 S^*.*state* $\leftarrow$ *forward arriving state*;
7 **return** S^*

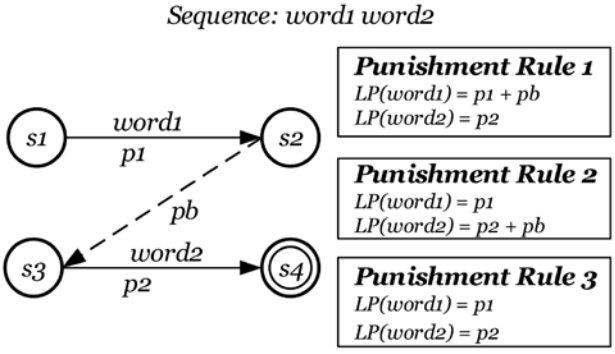

Figure 7: Three Punishment Rules.
In this figure, *p1* and *p2* stands for the probabilities, and *pb* for the backoff coefficient. $LP(word_i)$ is the context information of the i^{th} word. Note that the plus sign in this figure may have different meanings in the experiments, which we will explain later.

4.2 Modified CLSI

Essays with erroneous characters may interfere with both a human rater's and a machine's scoring. Moreover, automated essay correcting can help students learn languages. As for erroneous characters, for example, "今天 (today)" is a correct word, whereas "今田" is not. Genuine CLSI extracts the context information of "今田" into "今 (now)" and "田 (field)." If we detect and correct the erroneous character "田" to "天," we may extract a more accurate feature. Therefore, we propose Modified CLSI.

The difference between Modified CLSI and Genuine CLSI is that Modified CLSI detects and corrects erroneous characters before extracting context features. For instance, if we apply Modified CLSI, the erroneous character "田" in the above example will be corrected to "天," and the correct word "今天" with its context information will be extracted.

Generally speaking, there are four kinds of errors in essays, which are substitution, deletion, insertion, and transposition. By manual statistics from 200 essays from the MHK tests, we find that the most common kind of error is substitution, as Figure 8 shows. Therefore, we focus on substitution.

	Substitution(76%)	Deletion(13%)	Insertion(7%)	Transposition(4%)
Correct	休息 (rest)	五彩斑斓 (colorful)	爱好 (hobby)	原谅 (forgive)
Wrong	体息	五斑斓	爱好好	谅原

Figure 8: Examples of Four Kinds of Errors with Their Proportions.

A key component of Modified CLSI is the confusing-character table, including approximate and homophonic characters, as shown in Figure 9. We do not separate these two kinds of characters, because those approximate characters are usually homophonic with different tones in Chinese.

Figure 10 is a part of the confusing-character table. Characters in the same line are easily confused ones. Thus, when tentatively extracting a word and its context information in WFST, we can detect probable erroneous characters, and replace them with characters in the same line. For a certain character, its confused characters have the same chance to replace it. Figure 11 shows how this table helps detect and correct erroneous characters.

The only difference between the algorithm of Genuine CLSI and that of Modified CLSI is the judgment of whether an arc is passable or not. Therefore, we still use Algorithm 1 as the framework of Modified CLSI. However, in line 16, Algorithm 5 is used to replace Algorithm 3.

4.3 Constructing Contextualized-Semantic Space

The first step to construct a contextualized-semantic space is to convert essay vectors into an essay matrix. After that, SVD is used to reduce the dimension and construct a less noisy contextualized-semantic space.

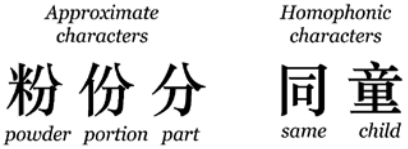

Figure 9: Examples of Approximate and Homophonic Characters.
The two homophonic characters are pronounced "tóng" and the three approximate characters are pronounced "fen" with different tones.

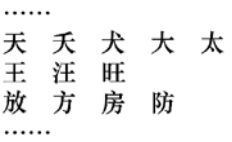

Figure 10: A Part of the Confusing-Character Table.

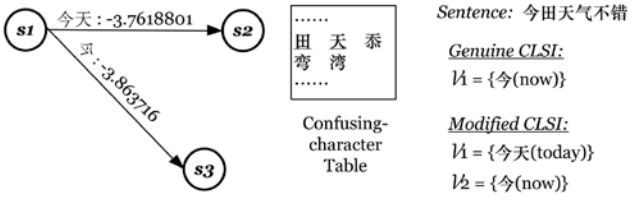

Figure 11: An Example to Show the Use of the Confusing-Character Table.
In this example, the second character "田(field)" of the input sentence is misused. If we use Genuine CLSI, we keep it unchanged, and thus, pass only one forward arc. When we use Modified CLSI, if we replace the erroneous character with its corresponding characters in the confusing-character table, then there will be another forward arc to pass, and thus, the erroneous character is corrected. "田" has several confusing characters in the table, like "天(sky)." As the word "今天(today)" includes one identical character "今" to the input sentence, and a confusing character "天," this arc is passable. Therefore, using Modified CLSI will have two possible word sets $\mathcal{V}_1$ and $\mathcal{V}_2$, whereas using Genuine CLSI has only $\mathcal{V}_1$.

Algorithm 5: Judge Passable Forward Arcs in Modified CLSI

Input: S_i.*unsegmented,* arc_i.*word* and the confusing-character table
Output: TRUE or FALSE
if S_i.*unsegmented starts with* arc_i.*word* **then**
  return *TRUE*;
end
if S_i.*unsegmented starts with* arc_i.*word after replacing certain characters in* S_i.*unsegmented according to the confusing-character table* **then**
  return *TRUE*;
else
  return *FALSE;*
end

Thus, any term-based essay vector can be transformed to that space, which provides a uniform metric under the same space for automated scoring.

4.3.1 From Essay Vectors to an Essay Matrix

After context information extraction, each essay is associated with its word list $\mathcal{V}$ and corresponding essay vector **d**. As in LSI, we need to array the essay vectors into an essay matrix. Note that $\mathcal{V}$ may contain identical words whereas the collection word set $\mathcal{V}^\dagger$ of an essay matrix does not contain identical words, so we have to merge these identical words in **d**. We propose two Merging Rules to merge them, which are also applied in the statement "`Applying a Punishment Rule`," where the backoff coefficient is merged to the context information of a certain word.

Merging Rule 1 (Plus): Using this rule, we simply add the same word's probabilities:

$$d^* = d_i + d_j \tag{6}$$

where d_i and d_j are context information of the same word in original essay vector **d**.

Merging Rule 2 (Log): As we use log probabilities in the *n*-gram language model, another way to calculate is

$$d^* = \log(10^{d_i} + 10^{d_j}). \tag{7}$$

After merging, the new word list $|\mathcal{V}^*| \le |\mathcal{V}|$. Moreover, the collection word set $\mathcal{V}^\dagger = \bigcup_{i=1}^{|\mathcal{D}|} \mathcal{V}_i^*$, and thus, $|\mathcal{V}^\dagger| \le \sum_{i=1}^{|\mathcal{D}|} |\mathcal{V}_i^*|$.

After all essay vectors are merged, they will be converted to an essay matrix. For certain words not included in an essay vector, we fill the corresponding elements with the log probability –99 (meaning the probability of 0, as $\log 0 \to -\infty$ and all the context information are log probabilities), denoting that these words are "impossible" words. Moreover, low-frequency words (those that only appear in less than two essays) and stop words (like auxiliary verbs) are eliminated, both from $\mathcal{V}^\dagger$ and the corresponding essay vectors. Additionally, if an essay collection is used as a testing set, it will use the same $\mathcal{V}^\dagger$, and the words absent in the training set will be eliminated as well. In other words, if the essay collection is separated into a training set and a testing set (or a validation set), $\mathcal{V}^\dagger$ is determined by the training set.

4.3.2 Singular Value Decomposition

We use the essay matrix to construct a semantic space. Given an essay matrix **D**, the optimization problem of CLSI performs as follows:

$$\begin{aligned} \min_{\mathbf{U},\mathbf{V}} \quad & ||\mathbf{D}-\mathbf{UV}||_F^2 \\ \text{subject to} \quad & \mathbf{U}^\top\mathbf{U}=\mathbf{I} \\ & \mathbf{V}^\top\mathbf{V}=\mathbf{I} \end{aligned}, \tag{8}$$

where $\mathbf{I}$ is an identity matrix. This problem can be solved by truncated SVD.

The standard mathematical strategy SVD decomposes the matrix into three matrices [17, 21].

$$\mathbf{D}=\mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top \tag{9}$$

$$\mathbf{D}^{M\times N}=\begin{bmatrix}\begin{bmatrix}u_{11}\\ \vdots \\ u_{M1}\end{bmatrix} & \cdots & \begin{bmatrix}u_{1M}\\ \vdots \\ u_{MM}\end{bmatrix}\end{bmatrix}\begin{bmatrix}\sigma_1 & & \\ & \ddots & \\ & & \sigma_{rank(\mathbf{D})} \\ 0 & \cdots & 0\end{bmatrix}\begin{bmatrix}[v_{11} & \cdots & v_{1N}]\\ & \vdots & \\ [v_{N1} & \cdots & v_{NN}]\end{bmatrix}, \tag{10}$$

where $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices and $\mathbf{\Sigma}$ is a diagonal matrix with $\sigma_1 \geq \sigma_2 \ldots \geq \sigma_{rank(\mathbf{D})}$ and extra zero.

We can truncate the original decomposition results into lower-dimensional matrices whose factorization is highly approximate to the matrix $\mathbf{D}$:

$$\mathbf{D}\approx\mathbf{D}^*=\mathbf{U}^*\mathbf{\Sigma}^*\mathbf{V}^{\top *}, \tag{11}$$

$$\mathbf{D}^{*M\times K}=\begin{bmatrix}\begin{bmatrix}u_{11}\\ \vdots \\ u_{M1}\end{bmatrix} & \cdots & \begin{bmatrix}u_{1K}\\ \vdots \\ u_{MK}\end{bmatrix}\end{bmatrix}\begin{bmatrix}\sigma_1 & & \\ & \ddots & \\ & & \sigma_K\end{bmatrix}\begin{bmatrix}[v_{11} & \cdots & v_{1N}]\\ & \vdots & \\ [v_{K1} & \cdots & v_{KN}]\end{bmatrix}. \tag{12}$$

This truncated SVD solves the optimization problem proposed in Eq. (8), and constructs a contextualized-semantic space.

4.4 Presenting the Essays Under the Space

Any essay vector is supposed to be projected to the contextualized-semantic space, in order to gain a uniform representation under the space for the scoring model. Therefore, given a term-based essay vector $\mathbf{d}=[d_1, d_2, \ldots, d_M]^\top$, it can be projected as follows:

$$\mathbf{d}^*=\mathbf{\Sigma}^{*-1}\mathbf{U}^{*\top}\mathbf{d} \tag{13}$$

where $\mathbf{d}^*\in\mathbb{R}^K$. So far, we complete the whole processing of CLSI methods.

5 SVM-Based Scoring Model

If we consider each scoring point as a label, AES can be processed as a kind of multiclass classification. We use the samples $(\mathbf{d}^*, y)$ from the training set to train a multiclass classifier as the scoring model, where $\mathbf{d}^*$ is a transformed essay vector and y is its human score. After training, given any transformed essay vector, we use this model to predict its label to be the predicted score. SVM is a state-of-the-art and effective solution that has been applied widely [8]. Therefore, we choose SVM to train the scoring model and score essays. Figure 12 shows how this model works.

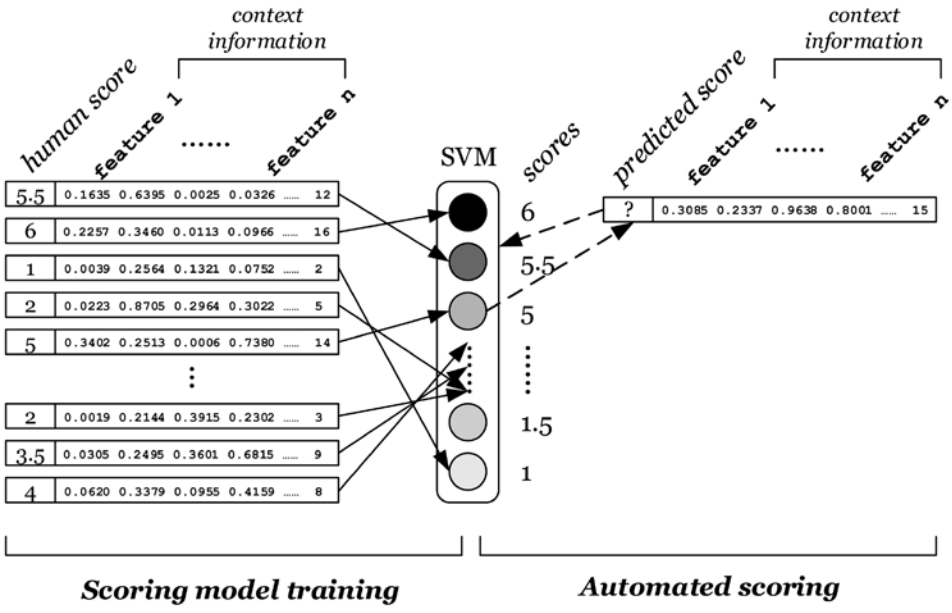

Figure 12: The SVM-Based Scoring Model.

6 Experiments

The programs in our experiments are written in C++. SVD and other matrix-related calculations are performed by using MATLAB. In scoring model training, we use `LIBSVM` (http://www.csie.ntu.edu.tw/223Ccjlin/libsvm) [6] to classify the essays. All experiments run on a Linux machine with a 2.9 GHz Intel Xeon E5-2690 CPU and 256G memory.

In this section, we first introduce our data sources and evaluation criteria in Sections 6.1 and 6.2, respectively. Next, we conduct experiments in Section 6.3 to select the best models for different methods introduced in this paper, and then use these selected models in Section 6.4 to compare our methods with existing methods to show that our methods perform best.

6.1 Data Sources

(i) *Language models (LM):* We have three kinds of corpora for training *n*-gram language models: Literature, Mixture, and News. The Literature corpus includes all the articles awarded the Mao Dun Literature Prize, the highest honor for Chinese writers. The Mixture corpus includes various sources like micro-blog posts, magazines, web pages, and so forth. The News corpus selects from *People's Daily*, the bestseller newspaper in mainland China. These corpora are very representative, which stand for daily use of Chinese, mixture of language usage, and standard Chinese, respectively.

(ii) *The confusing-character table:* The confusing-character table comes from *Modern Chinese Dictionary*, including the most confusing homophonic and approximate characters. Additionally, according to previous research about the MHK tests, we manually add some frequently confusing characters. Thus, the table includes 6674 characters in total.

(iii) *The sample set:* We use the written essays with their human scores from the MHK test assigned by a same topic as the sample set. It is easy to see that the high-score essay usually contains fluent sentences, whereas the poor essay presents incoherent sentences. Omitting the empty essays and those without human scores, we obtain two training sets with the sizes of 10,000 and 1400, a validation set with the size of 600 and a testing set with the size of 600. The training set is used to train a model; the validation set

is to determine details of models due to various settings; the testing set is to test and make comparison among different versions of CLSI, conventional LSI, and other methods. Figure 13 shows the human score distributions of these sets.

6.2 Performance Criteria

There are several criteria from different scopes to evaluate the performance of an AES system [35]. In this paper, we use *Rating Agreement*, *Fairness Coefficient*, *Quadratic Weighted Kappa*, *Spearman's Correlation*, and *Pearson's Correlation*.

(i) *Rating agreement:* Rating agreement is an overall evaluation criterion, including *exact agreement* and *adjacent agreement* [10]. The exact agreement calculates the exact equal scoring results between human scores and predicted scores, whereas the adjacent agreement is used to count the results with an acceptable bias. In our experiments, we use adjacent agreement (*R.A.*):

$$Rating\ Agreement(bias) = \frac{\sum_{i=1}^{N} 1\{|hs_i - ps_i| \le bias\}}{N}, \tag{14}$$

where N is the size of the testing set; hs_i and ps_i are the human score and predicted score of the i^{th} essay, respectively; and *bias* can be set to an acceptable value according to various tests. In the experiments, we set *bias* to 1 according to the scoring criteria of the MHK test. The indicator function is $\mathbf{1}\{TRUE\} = 1$ and $\mathbf{1}\{FALSE\} = 0$.

(ii) *Fairness coefficient:* Fairness coefficient looks into each class. From the human score distribution in Figure 13, we see that the majority lies in the score of 3.5. If the classifier scores all essays 3.5, *R.A.*(1) can rise up to 83.5%, which apparently violates the fairness of the test. Some poorly written essays may get higher scores than they should have, whereas some excellent ones may get lower scores. Therefore, we design a criterion named fairness coefficient (*F.C.*):

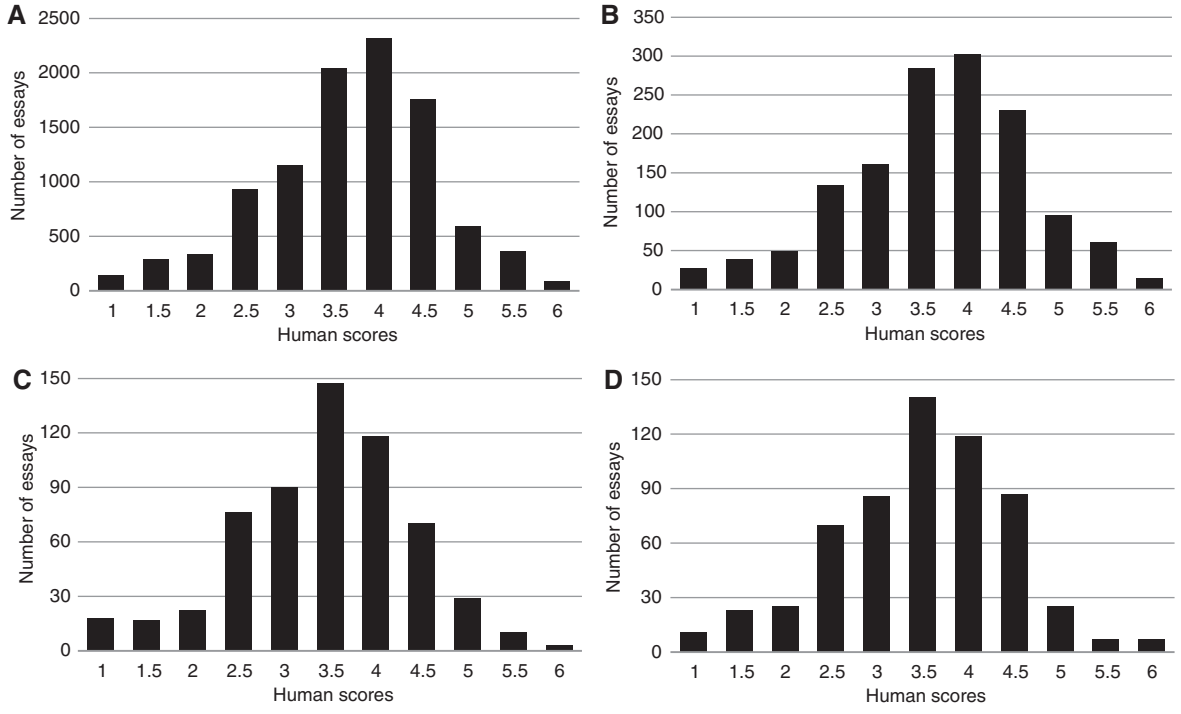

Figure 13: The Human Score Distributions of Training Set 1 (A), Training Set 2 (B), the Validation set (C), and the Testing Set (D).

$$Fairness\ Coefficient(bias) = \frac{\sum_{i=1}^{C}\left(log\left(\frac{n_i}{N}\right)\cdot R.A(bias)_i\right)}{\sum_{i=1}^{C} log\left(\frac{n_i}{N}\right)}, \tag{15}$$

where C is the number of classes, N is the size of the validation or testing set, n_i is the size of class i, and $R.A.(bias)_i$ is the Rating Agreement of class i with the *bias*. The reason for using this coefficient is because low- and high-score essays weigh more but have less number than ordinary essays in the sampling set. We increase their weights by using the logarithm of their proportions, focusing on the minorities.

(iii) *Quadratic weighted kappa:* Quadratic weighted kappa κ takes chance agreement into account because there exists the risk that a system just uses a baseline classifier (which will be explained later) and guesses all the scores by pure chance. To calculate this value, we can construct a confusion matrix, showing the human scores and the predicted scores.

(iv) *Spearman's correlation:* As in evaluations of other scoring systems, we calculate Spearman's correlation (*Spearman*).

(v) *Pearson's correlation:* For the sake of completeness, we introduce the last criterion, Pearson correlation (*Pearson*).

6.3 Experimental Results with Different Models

6.3.1 Genuine CLSI

As the language model has impact on the results, we perform Genuine CLSI with each language model. Literature, Mixture, and News produced collection word sets of sizes of 3720, 3724, and 3722, respectively. Table 1 lists the scoring details of Genuine CLSI with the language model *News* (LM: News). We omit the scoring details of other two language models due to space limitation. For each model, we use various combinations of Punishment Rules (PR1, PR2, and PR3) and Merging Rules showed in Formulas 6 and 7 (Plus and Log). Lines 3–8 in Table 1 means the numbers of essays recognized under different conditions. From Table 1, we see that Genuine CLSI successfully recognizes low-score essays.

As for the performance criteria *R.A.* and *F.C.*, we show the experimental results in Table 2. When choosing an appropriate model, Rating Agreement is the first criterion to be considered, while Fairness Coefficient is the second one. Therefore, we select the combination of Mixture corpus, Punishment Rule 2, and Merging Rule Log as the best combination for Genuine CLSI because the Rating Agreement of this combination reaches the highest.

6.3.2 Modified CLSI

For Modified CLSI, we follow the same settings as for Genuine CLSI to choose the best combination. Literature, Mixture, and News produce collection word sets of size 3711, 3734, and 3774, respectively. Table 3 lists

Table 1: Scoring Details of Genuine CLSI (LM: News).

Human score	**1**	**1.5**	**2**	**2.5**	**3**	**3.5**	**4**	**4.5**	**5**	**5.5**	**6**
Validation set	18	17	22	76	90	147	118	70	29	10	3
PR1 + Log	9	7	10	63	83	146	113	69	23	1	0
PR1 + Plus	13	7	10	58	78	144	112	69	21	1	0
PR2 + Log	13	8	10	59	78	144	113	69	21	1	0
PR2 + Plus	13	8	10	59	78	144	112	69	21	1	0
PR3 + Log	13	9	11	61	80	144	113	69	21	2	0
PR3 + Plus	13	8	10	59	78	144	113	69	21	1	0

Bold values mean the best results.

Table 2: Results of Genuine CLSI.

	Literature		Mixture		News	
	***R.A.*(1) (%)**	***F.C.*(1)**	***R.A.*(1) (%)**	***F.C.*(1)**	***R.A.*(1) (%)**	***F.C.*(1)**
PR 1 + Log	86.50	0.512884	85.17	0.523624	**87.33**	0.503083
PR1 + Plus	85.83	0.487807	86.67	0.504911	85.50	0.512351
PR2 + Log	**87.17**	0.521066	**87.50**	0.520829	84.33	0.520196
PR2 + Plus	85.83	0.505677	86.67	0.504682	85.83	0.519765
PR3 + Log	87.00	0.514504	86.00	0.521734	87.17	0.547303
PR3 + Plus	86.17	0.507489	86.33	0.503171	86.00	0.520196

Bold values mean the best results.

Table 3: Scoring Details of Modified CLSI (LM: News).

Human score	**1**	**1.5**	**2**	**2.5**	**3**	**3.5**	**4**	**4.5**	**5**	**5.5**	**6**
Validation set	18	17	22	76	90	147	118	70	29	10	3
PR1 + Log	11	9	14	63	79	143	113	66	22	2	0
PR1 + Plus	9	8	11	64	85	145	113	68	22	6	0
PR2 + Log	10	9	14	63	79	143	114	66	22	2	0
PR2 + Plus	9	8	12	64	85	145	113	68	22	6	0
PR3 + Log	11	9	14	64	79	143	113	66	21	1	0
PR3 + Plus	9	8	12	64	85	145	113	68	21	6	0

Bold values mean the best results.

the scoring details of Modified CLSI with the language model *News*. We omit the scoring details of other two language models due to space limitation. From Table 3, we see that Modified CLSI can successfully recognize low-score essays.

Table 4 shows the results of Modified CLSI. From Table 4, we see that when we use the corpus News, Punishment Rule 2, and Merging Rule Log, both *R.A.*(1) and *F.C.*(1) are the best.

6.3.3 Conventional LSI

We run conventional LSI on the same training and validation sets to choose the best language model. Similarly, we list the scoring details in Table 5. From Table 5, it is obvious that LSI fails to recognize the low- and high-score essays, from point 1 to 2 and from 5.5 to 6, respectively, which is unacceptable because it causes local overrating and underrating problems.

Table 6 shows the results of conventional LSI. From Table 6, we see that LSI performs best when using the News corpus to train the model. Therefore, in further experiments, we choose *News* for it.

Table 4: Results of Modified CLSI.

	Literature		Mixture		News	
	***R.A.*(1) (%)**	***F.C.*(1)**	***R.A.*(1) (%)**	***F.C.*(1)**	***R.A.*(1) (%)**	***F.C.*(1)**
PR1 + Log	87.00	0.525937	**87.83**	0.527405	87.00	0.550349
PR1 + Plus	87.67	0.568560	**87.83**	0.517870	88.50	0.576078
PR2 + Log	87.17	0.516931	**87.83**	0.533746	**88.67**	**0.580783**
PR2 + Plus	**88.00**	0.569614	87.33	0.536656	87.00	0.544682
PR3 + Log	**88.00**	0.567975	85.83	0.548933	86.83	0.535112
PR3 + Plus	**88.00**	0.569614	87.33	0.536656	88.50	0.577512

Bold values mean the best results.

Table 5: Scoring Details of Conventional LSI.

Human score	**1**	**1.5**	**2**	**2.5**	**3**	**3.5**	**4**	**4.5**	**5**	**5.5**	**6**
Validation set	18	17	22	76	90	147	118	70	29	10	3
Literature	0	0	0	65	87	147	118	70	22	0	0
Mixture	0	0	0	62	89	147	118	70	22	0	0
News	0	0	1	65	88	147	118	70	22	0	0

Bold values mean the best results.

Table 6: Results of Conventional LSI.

	***R.A.*(1) (%)**	***F.C.*(1)**
Literature	84.83	0.346885
Mixture	84.67	0.345652
News	**85.17**	0.352250

Bold value mean the best results.

6.3.4 Sensitivity to Dimension

In previous experiments, dimension K in truncated SVD is set to an empirical value 100. This value is important because it will produce much noise if it is too high, or lose information if too low. In order to test the sensitivities of conventional LSI, Genuine CLSI, and Modified CLSI to this value, we run experiments from $K = 25$–200 at intervals of 25, and plot them in Figure 14.

In Figure 14, we see that for $R.A.(1)$, the performance of conventional LSI decreases sharply as K increases, whereas those of CLSIs decrease more smoothly and are better than the conventional method. For $F.C.(1)$, the performance of conventional LSI fluctuates continually, whereas those of CLSIs remain relatively stable and are better, too. Hence, we conclude that CLSIs are more stable considering the dimension K. For the other three criteria, κ, $\rho(Spearman)$, and $\rho(Pearson)$, we see clearly that CLSIs stay relatively stable, but conventional LSI fluctuates and is worse than CLSIs.

These experiments justify that CLSIs are less sensitive to K, meaning that K performs well no matter how much latent semantics is required, which is of great importance, because it is often hard to find the appropriate dimension K, and less K means less memory usage.

6.4 Further Experiments on Selected Models

6.4.1 Comparisons Using Selected Models

Previously, we use various combinations and corpora to choose the best models for Genuine CLSI, Modified CLSI, and LSI, respectively. In order to explicitly demonstrate that CLSI outperforms LSI for ACES, we compare CLSIs and LSI along with other methods. Table 7 lists the scoring details, and Table 8 shows comparisons of different methods.

In Tables 7 and 8, three additional methods are added. The first one is a Baseline Classifier, which counts the most frequent class in the training set and guesses all essays' scores in the testing set. In our experiments, as the most frequent class in the training set is point 4, all the essays in the testing set are given 4. The extremely low κ, $\rho(Spearman)$, and $\rho(Pearson)$ justify that this classifier is not applicable, although it has relatively close $F.C.(1)$ to the LSI method.

The second one is RLSI, which is also an LSI-based method and has been applied to ACES, but its scoring perspective is different from CLSI: RLSI is from the topic perspective, whereas CLSI is from the language proficiency and content perspectives.

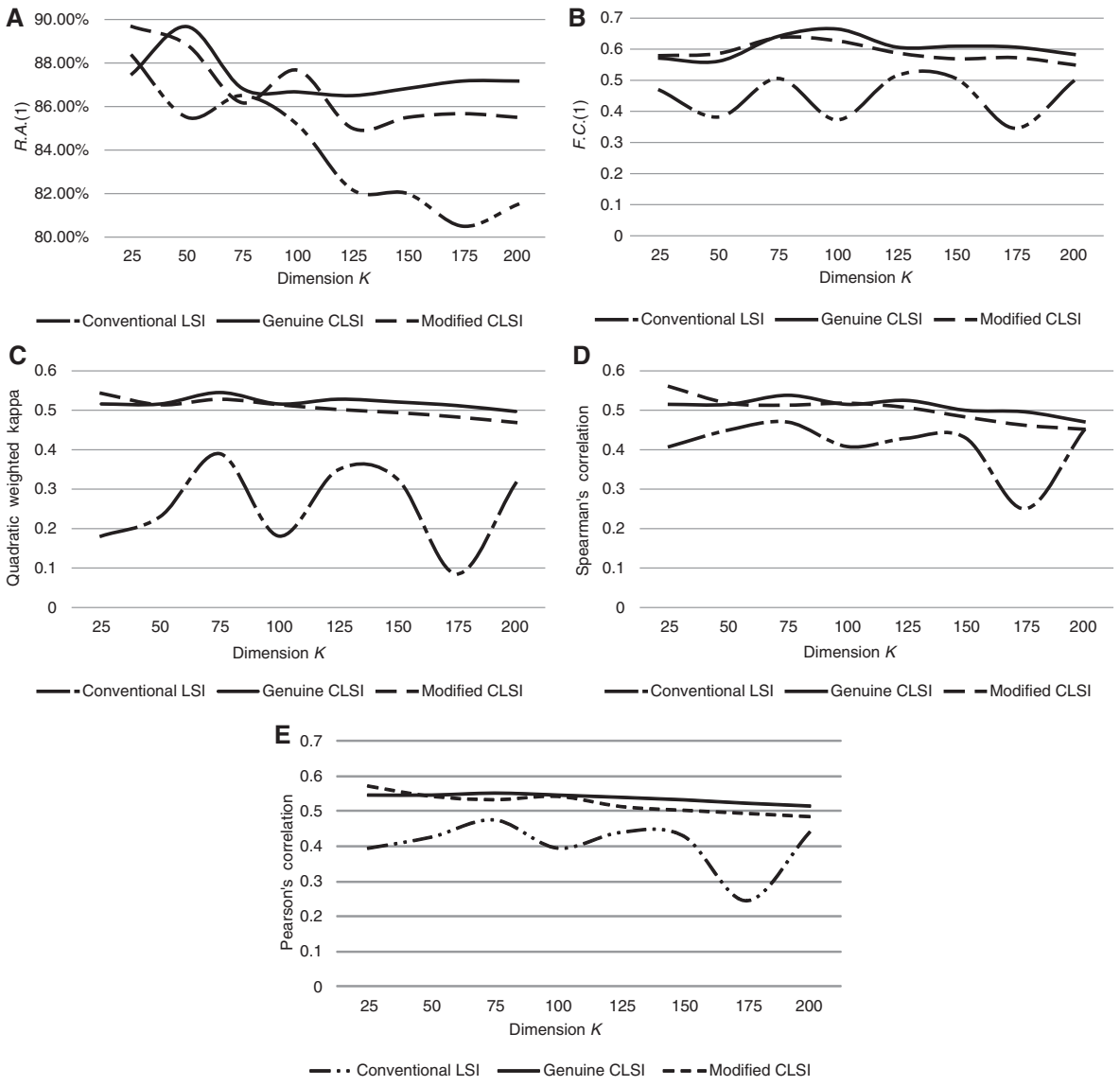

Figure 14: Comparisons of Conventional LSI, Genuine CLSI, and Modified CLSI with Different Dimensions (*K*).
(A) Rating Agreement; (B) Fairness Coefficient; (C) Quadratic Weighted Kappa; (D) Spearman's Correlation; (E) Pearson's Correlation.

Table 7: Scoring Details of Different Methods.

Human score	**1**	**1.5**	**2**	**2.5**	**3**	**3.5**	**4**	**4.5**	**5**	**5.5**	**6**
Testing set	11	23	25	70	86	140	119	87	25	7	7
Genuine CLSI	8	8	11	64	83	138	117	85	23	1	0
Modified CLSI	6	9	12	63	81	137	114	84	24	3	0
Conventional LSI	0	1	1	56	85	140	118	87	25	0	0
RLSI	2	3	9	50	67	137	111	87	24	4	0
LDA	1	3	1	32	80	139	117	87	23	0	0

Bold values mean the best results.

The third one is LDA (http://www.cs.princeton.edu/223Cblei/lda-c/index.html), which also scores essays from the topic perspective. Though this method is a probabilistic one and not based on LSI, we also include it for the sake of completeness.

Table 8: Comparisons of Different Methods.

	R.A.(1) (%)	*F.C.*(1)	κ	ρ(*Spearman*)	ρ(*Pearson*)
Genuine CLSI	**89.67**	0.561517	**0.516**	0.515	**0.5460**
Modified CLSI	88.83	**0.586551**	0.514	**0.518**	0.5423
Conventional LSI	85.50	0.381993	0.231	0.450	0.4265
RLSI	82.33	0.499882	0.318	0.386	0.3941
LDA	80.50	0.428769	0.308	0.465	0.4367
Baseline classifier	76.17	0.320264	0	0	0

Bold values mean the best results.

From Table 8, we see that both CLSI methods outperform other methods in rating agreement and fairness coefficient. More specifically, Genuine CLSI performs better than Modified CLSI in rating agreement slightly, and lower in Fairness Coefficient slightly. Therefore, we conclude that both versions of CLSI perform well and can be applied to ACES.

6.4.2 Scoring Deviation

In order to further demonstrate the performance of CLSI, we define scoring deviation as

$$Scoring\ Deviation(lower, upper) = \frac{\sum_{n=1}^{N} 1\{lower < |hs_i - ps_i| \leqslant upper\}}{N}, \tag{16}$$

where *lower* and *upper* are the lower bound and upper bound of a given interval, respectively.

We calculate scoring deviation and show comparisons in Table 9. We see that conventional LSI has a high deviation because there are >2% essays whose differences between human scores and predicted scores are 2.5. Both RLSI and LDA present unacceptable high deviations, because there are still 0.83% and 0.17% essays with the differences higher than 2.5. By sharp contrast, the corresponding deviations of both Genuine CLSI and Modified CLSI are much lower. These results further confirm that CLSI is superior to other methods.

7 Conclusions and Future Work

In this paper, we have proposed a new LSI method based on context information that can be applied to ACES, and developed its two versions: Genuine CLSI and Modified CLSI. Using Genuine CLSI, one can extract context information directly without segmenting the essays in advance. Detection and correction for erroneous characters is integrated to the processing in Modified CLSI, which can execute in the text preprocessing stage without extra computation. Both methods can effectively address the local overrating and underrating problem caused by conventional LSI, and score essays from the perspective of language fluency and content. Moreover, Modified CLSI can also be used to help student learn languages.

Table 9: Comparisons of Scoring Deviation.

	Genuine CLSI	Modified CLSI	Conventional LSI	RLSI	LDA
(1, 1.5]	5.00%	6.33%	7.17%	9.50%	10.33%
(1.5, 2]	4.17%	3.83%	5.17%	5.00%	5.50%
(2, 2.5]	0.67%	0.17%	2.17%	2.67%	3.17%
(2.5, 3]	0.00%	0.00%	0.00%	0.50%	0.50%
Overall	9.84%	10.33%	14.51%	17.67%	19.05%

Although the local overrating and underrating problems have been alleviated to some extent, from the experimental results we see that these problems are not solved completely, leading to a Fairness Coefficient that is still not high. Therefore, we point out two ways to address these problems using CLSI. One is to reinforce context information so that ACES can distinguish poor, ordinary, and excellent essays better. The other is to add other deeper features to assist ACES. Moreover, deep learning is also a potentially powerful way to this field.

Acknowledgments: This work is supported by the Beijing Higher Education Young Elite Teacher Project, under grant YETP-0768; the Fundamental Research Funds for the Central Universities, under grant YX2014-18; and the National Natural Science Foundation of China, under grant 61472369.

Bibliography

[1] Y. Attali and J. Burstein, Automated essay scoring with E-rater v.2.0, *JTLA* **4** (2006), 3–30.
[2] D. M. Blei, A. Y. Ng and M. I. Jordan, Latent Dirichlet allocation, *J. Mach. Learn. Res.* **3** (2003), 993–1022.
[3] J. Burstein, M. Chodorow and C. Leacock, Criterion^{SM} online essay evaluation: an application for automated evaluation of student essays, in: *Proceedings of the 15th Annual Conference on Innovative Applications of Artificial Intelligence*, pp. 3–10, 2003.
[4] L. Cai, X. Peng, D. Ke and J. Zhao, Research of the feature for automated essay scoring system for Chinese proficiency test for minorities, in: *Proceedings of the 5th Youth Workshop of Computational Linguistics (YWCL)*, 2010.
[5] Y. Cao and C. Yang, Automated Chinese essay scoring with latent semantic analysis, *ER* **3** (2007), 63–71.
[6] C. Chang and C. Lin, LIBSVM: a library for support vector machines, *ACM T. Intel. Syst. Tec.* **2** (2011), 27:1–27:27, Software available at http://www.csie.ntu.edu.tw/223Ccjlin/libsvm. Accessed February, 2014.
[7] S. F. Chen and J. Goodman, An empirical study of smoothing techniques for language modeling, *Comput. Speech Lang.* **13** (1999), 359–394.
[8] C. Cortes and V. Vapnik, Support-vector networks, *Mach. Learn.* **20** (1995), 273–297.
[9] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer and R. Harshman, Indexing by latent semantic analysis, *J. Am. Soc. Inform. Sci.* **41** (1990), 391–407.
[10] S. Dikli, An overview of automated scoring of essays, *JTLA* **5** (2006), 1–34.
[11] S. Elliot, IntelliMetric: from here to validity, in: *Automated Essay Scoring: A Cross-disciplinary Perspective*, pp. 71–86, Lawrence Erlbaum Associates, Inc., Mahwah, NJ, 2003.
[12] 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: *Proceedings of 12th International Conference on Document Analysis and Recognition*, pp. 763–767, 2013.
[13] S. Hao, Y. Xu, D. Ke, K. Su and H. Peng, SCESS: A WFSA-based automated simplified Chinese essay scoring system with incremental latent semantic analysis, *Nat. Lang. Eng.*, **22** (2014), 291–319.
[14] S. Hao, Y. Xu, H. Peng, K. Su and D. Ke, Automated Chinese essay scoring from topic perspective using regularized latent semantic indexing, in: *Proceedings of the 22nd International Conference on Pattern Recognition*, pp. 3092–3097, 2014.
[15] M. M. Islam and A. S. M. L. Hoque, Automated essay scoring using generalized latent semantic analysis, *J. Comput.* **7** (2012), 616–626.
[16] D. Ke, X. Peng, Z. Zhao, Z. Chen and S. Wang, Word-level-based automated Chinese essay scoring method, in: *Proceedings of National Conference on Man-Machine Speech Communication*, pp. 57–59, 2011.
[17] V. Klema and A. Laub, The singular value decomposition: its computation and some applications, *IEEE T. Automat. Cont.* **25** (1980), 164–176.
[18] T. K. Landauer, D. Laham and P. W. Foltz, The intelligent essay assessor: putting knowledge to the test, in: *The Association of Test Publishers Computer-Based Testing: Emerging Technologies and Opportunities for Diverse Applications Conference*, 2001.
[19] T. K. Landauer, D. Laham and P. W. Foltz, Automated essay scoring: a cross-disciplinary perspective, in: *Automated Scoring and Annotation of Essays with the Intelligent Essay Assessor*, pp. 87–112, Lawrence Erlbaum Associates, Inc., Mahwah, NJ, 2003.
[20] Y. Li, *Automated Essay Scoring for Testing Chinese as a Second Language*, PhD thesis, 2006.
[21] C. F. V. Loan, Generalizing the singular value decomposition, *SIAM J. Numer. Anal.* **13** (1976), 76–83.
[22] M. Mohri, Weighted finite-state transducer algorithms: an overview, in: *Formal Languages and Applications*, vol. 148, pp. 551–564, 2004.

[23] P. Nakov, A. Popova and P. Mateev, Weight functions impact on LSA performance, in: *Proceedings of EuroConference RANLP*, pp. 187–193, 2001.
[24] E. B. Page, Computer grading of student prose, using modern concepts and software, *J. Exp. Educ.* **62** (1994), 127–142.
[25] H. Peng, The minorities-oriented Chinese level test, *CE* **10** (2005), 57–59.
[26] X. Peng, D. Ke, Z. Zhao, Z. Chen and B. Xu, Automated Chinese essay scoring based on word scores, *JCIP* **2** (2012), 102–108.
[27] C. Ramineni, C. S. Trapani, D. M. Williamson, T. Davey and B. Bridgeman, Evaluation of the e-rater scoring engine for the TOEFL independent and integrated prompts, *ETS Research Report* (2012).
[28] L. M. Rudner and T. Liang, Automated essay scoring using Bayes' theorem, *JTLA* **1** (2002), 3–21.
[29] M. Shermis and J. Burstein, *Automated essay scoring: a cross-disciplinary perspective*, Psychology Press, Brighton and Hove, UK, 2003.
[30] A. Stolcke, SRILM – an extensible language modeling toolkit, in: *Proceedings of the International Conference on Spoken Language Processing*, pp. 901–904, 2002.
[31] Vantage Learning, *About IntelliMetric*, Vantage Learning, Newtown, PA, 2001.
[32] Vantage Learning, *A true score study of 11th grade student writing responses using IntelliMetric Version 9.0*, Vantage Learning, Newtown, PA, 2003.
[33] K. Wang, C. Zong and K. Su, Integrating generative and discriminative character-based models for Chinese word segmentation, *ACM Trans. Asian Lang. Inform. Process.* **31** (2012).
[34] Q. Wang, J. Xu, H. Li and N. Craswell, Regularized latent semantic indexing: a new approach to large-scale topic modeling, *ACM T. Inform. Syst.* **31** (2013).
[35] H. Yannakoudakis, T. Briscoe and B. Medlock, A new dataset and method for automatically grading ESOL texts, in: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pp. 180–189, 2011.
[36] Y. Zhao, Application of latent semantic analysis in auto-grading system, *JYU (Natural Science)* **37** (2011), 345–348.