Mining Sequential Patterns and Tree Patterns to Detect Erroneous Sentences

Guihua Sun * **Gao Cong** **Xiaohua Liu** **Chin-Yew Lin** **Ming Zhou**
Chongqing University Microsoft Research Asia
sunguihua5018@163.com {gaocong, xiaoliu, cyl, mingzhou}@microsoft.com

Abstract

An important application area of detecting erroneous sentences is to provide feedback for writers of English as a Second Language. This problem is difficult since both erroneous and correct sentences are diversified. In this paper, we propose a novel approach to identifying erroneous sentences. We first mine labeled tree patterns and sequential patterns to characterize both erroneous and correct sentences. Then the discovered patterns are utilized in two ways to distinguish correct sentences from erroneous sentences: (1) the patterns are transformed into sentence features for existing classification models, e.g., SVM; (2) the patterns are used to build a rule-based classification model. Experimental results show that both techniques are promising while the second technique outperforms the first approach. Moreover, the classification model in the second proposal is easy to understand, and we can provide intuitive explanation for classification results.

Introduction

For writers of English as a Second Language (ESL), tools capable of automatically detecting errors in their writing are useful and desirable. Moreover, detecting erroneous sentences is also useful in controlling the quality of parallel bilingual sentences mined from the Web, and evaluating machine translation results (Corston-Oliver, Gamon, & Brockett 2001; Gamon, Aue, & Smets 2005). Unfortunately, this problem is hard, despite its importance, since ESL writers of different first-language backgrounds and skill levels may make various errors which are difficult to characterize. There has been little progress in the area over the last decade (Brockett, Dolan, & Gamon 2006).

The common mistakes (Yukio et al., 2001; Gui and Yang, 2003) made by ESL learners include spelling, lexical collocation, sentence structure, tense, agreement, verb formation, wrong Part-Of-Speech (POS), article usage, etc. Research into error detection for ESL writers remains largely focused on certain kinds of grammar errors, including tense, agreement, verb formation, article usage, etc. There has been little work on detecting general grammatical errors (but not some pre-defined types of errors) with the exception of commercial grammar checker tools. However commercial tools are usually not geared to meet the needs of ESL writers. For example, the Grammar checker provided by Microsoft Word is designed primarily with native writers in mind and often fails to identify the mistakes made by ESL writers, e.g. Microsoft Word cannot detect the sentence structure error in the sentence "*Only if my teacher has given permission, I am allowed to enter this room.*"

Some methods of detecting erroneous sentences are based on manual rules. These methods (Heidorn 2000; Michaud, McCoy, & Pennington 2000; Bender *et al.* 2004) have been shown to be effective in detecting certain kinds of grammatical errors in the writing of English learners. However, it could be expensive to write rules manually. Linguistic experts are needed to write rules of high quality; Also, it is difficult to produce and maintain a large number of non-conflicting rules to cover a wide range of grammatical errors for ESL writers with different first-language backgrounds. Worse still, it is hard to write rules for some grammatical errors, for example, detecting errors concerning the articles and singular plural usage (Nagata *et al.* 2006).

Instead of using hand-crafted rules, statistical approaches (Chodorow & Leacock 2000; Izumi *et al.* 2003; Brockett, Dolan, & Gamon 2006; Nagata *et al.* 2006) build statistical models to identify sentences containing errors. However, existing statistical approaches focus on some pre-defined errors and the reported results are not attractive. Moreover, these approaches, e.g., (Izumi *et al.* 2003; Brockett, Dolan, & Gamon 2006) usually need errors to be specified and tagged in the training sentences, which requires expert help to be recruited.

In this paper we propose a novel approach to identifying erroneous sentences. The basic idea of the approach is outlined as follows. We first generate two datasets, POS tags of sentences and parse trees of sentences, by pre-processing each sentence. We then mine *labeled tree* from parse trees and *labeled sequential patterns* from POS tags, respectively, for both erroneous sentences and correct sentences. The discovered patterns are used in two fashions to build classification models. (1) They are used as features for existing learning models; and (2) They are used to build a pattern based classifier. The problem of using sequential patterns and tree patterns for classification remains largely unexplored while there are a host of studies on leveraging association rules for classification, e.g. (Liu, Hsu, & Ma 1998).

We make the following main contributions in this paper.

- We propose two approaches to employing *labeled tree patterns* (LTPs) and *labeled sequential patterns*(LSPs) to build learning models for error detection: (1) patterns are converted into features for existing classification models, including SVM and NB; (2) we develop a new pattern-based classification (PBC) model.
- We define the most significant k (to be explained) *labeled tree patterns* and *labeled sequential patterns* for each training sentence, and develop algorithms for min-

*Work done while the author was a visiting student at MSRA

ing them to build classification models.

- We evaluate our methods on two datasets consisting of sentences written by Japanese and Chinese, respectively. Experimental results show that our proposed techniques are promising. Our method outperforms Microsoft Word03 and ALEK (Chodorow & Leacock 2000) from Educational Testing Service (ETS) on the data that we tested. Experimental results also show that pattern based classification method outperforms NB and SVM using our discovered patterns as features. Moreover, the pattern based classification method can provide intuitive explanation for classification results while it is hard to derive explanation for SVM results, and thus can provide feedback on the specific errors in erroneous sentences.

The rest of this paper is organized as follows. The next section discusses related work. Section 3 presents the proposed technique. We evaluate our proposed technique in Section 4. Section 5 concludes this paper and discusses future work.

Related Work

Research on detecting erroneous sentences can be classified into two categories. The first category makes use of handcrafted rules, e.g., template rules (Heidorn 2000) and malrules in context-free grammars (Michaud, McCoy, & Pennington 2000; Bender *et al.* 2004). As discussed in Section 1, manual rule based methods have some shortcomings.

The second category uses statistical techniques to detect erroneous sentences. An unsupervised method (Chodorow & Leacock 2000) is employed to detect grammatical errors by inferring negative evidence from TOEFL administrated by ETS. The method (Izumi *et al.* 2003) aims to detect omission-type and replacement-type errors. Transformation-based leaning is employed in (Shi & Zhou 2005) to learn rules to detect errors for speech recognition outputs. They also require specifying error tags that can tell the specific errors and their corrections in the training corpus. The phrasal Statistical Machine Translation (SMT) technique is employed to identify and correct writing errors (Brockett, Dolan, & Gamon 2006). This method must collect a large number of parallel corpora (pairs of erroneous sentences and their corrections) and performance depends on SMT techniques that are not yet mature. The work in (Nagata *et al.* 2006) focuses on a type of error, namely mass vs. count nouns. In contrast to these statistical methods, our technique needs neither errors tagged nor parallel corpora, and is not limited to a specific type of grammatical error.

There are also studies on automatic essay scoring at document-level. For example, E-rater (Burstein *et al.* 1998), developed by the ETS, and Intelligent Essay Assessor (Foltz, Laham, & Landauer 1999). The evaluation criteria for documents are different from those for sentences. A document is evaluated mainly by its organization, topic, diversity of vocabulary, and grammar while a sentence is done by grammar and lexical choice.

In our preliminary work (Sun *et al.* 2007), we investigate using labeled sequential patterns mined from POS tags of sentences as features for SVM to detect erroneous sentences; our results show that labeled sequential patterns are more effective than other features, e.g. lexical collocation, syntactic score etc. This paper extends the work (Sun *et al.* 2007) in three aspects: 1) we use both labeled tree patterns and sequential patterns to build classification models to detect erroneous sentences. 2) we impose constraints when mining sequential patterns, which enable us to mine more meaningful and robust patterns for classification. 3) we build pattern-based classification models.

There is little work making use of sequential patterns and tree patterns for classification. To our knowledge, sequential patterns are used to identify comparative sentences (Jindal & Liu 2006) and tree patterns are employed to classify XML data (Zaki & Aggarwal 2003). Our pattern-based classification approach differs from the two previous approaches in that 1) we use both tree patterns and sequential patterns; 2)classification methods are different. Out pattenbased classification method is inspired by the RCBT classification method (Cong *et al.* 2005) using top-k rule groups to classify microarray data.

Proposed Technique

This section first gives our problem statement and then presents our proposed technique to build learning models.

Problem Statement

Given a set of training data containing correct and erroneous sentences, we study the problem of identifying erroneous/correct sentences. Unlike some previous work, our technique requires neither that the erroneous sentences are tagged with detailed errors, nor that the training data consist of parallel pairs of sentences (an error sentence and its correction). The erroneous sentence contains a wide range of grammatical errors. We do not consider spelling errors in this paper.

We address the problem by building classification models. The main challenge is to automatically extract representative features for both correct and erroneous sentences to build effective classification models. We illustrate the challenge with an example. Consider an erroneous sentence, "*If* Maggie *will* go to supermarket, she *will* buy a bag for you." It is difficult for previous methods using statistical techniques to capture such an error. For example, N-gram language model is considered to be effective in writing evaluation (Burstein *et al.* 1998; Corston-Oliver, Gamon, & Brockett 2001). However, it becomes very expensive if $N > 3$ and N-grams only consider continuous sequence of words, which is unable to detect the above error *"if...will...will"*.

We propose *labeled tree patterns* (LTPs) and *labeled sequential patterns* (LSPs) to effectively characterize the features of correct and erroneous sentences. We next illustrate the two kinds of patterns with examples.

Example 1: Consider an erroneous sentence, "*A beautiful dog.*" A labeled tree pattern (LTP) (S(NPB)(.)) $\Rightarrow E$ (E denotes Erroneous) is contained by its syntax tree [1],

[1] In this paper we use nested brackets to represent tree structure.

(S(NPB(DT)(JJ)(NN))(.)). Consider another erroneous sentence, "*He visit here yesterday.*" Two LSPs, <he, VB> $\Rightarrow E$ and <VB, yesterday> $\Rightarrow E$, where VB is POS tag of base form verb, are contained by the sentence. □

Mining Labeled Tree Patterns

Labeled Tree Patterns (LTPs): Each sentence can be parsed into a syntactic tree using the Collins' Parser Toolkit (Collins 1997). For example, sentence "Jake hit the ball" will be parsed into S(NP)(VP(VB)(NP(DT)(NN))). We denote each syntactic tree as $t = (V, E, \preceq)$, where V is the set of labeled nodes, E the set of edges and $\preceq$ the set of sibling relations. All the syntactic trees form the dataset T. We say that a tree $t_1 = (V_1, E_1, \preceq_1)$ is *contained* in tree $t_2 = (V_2, E_2, \preceq_2)$ if there exists a matching function ψ: $t_1 \rightarrow t_2$ satisfying the following conditions for any v, v_1, $v_2 \in t_1.V_1$: 1) ψ preserves the parent relation, i.e., $(v_1, v_2) \in E_1$ iff $(\psi(v_1), \psi(v_2)) \in E_2$; 2) ψ preserves the sibling relation, i.e., $(v_1, v_2) \in \preceq_1$ iff $(\psi(v_1), \psi(v_2)) \in \preceq_2$; 3) ψ preserves the labels, i.e., the label of v in t_1 is the same with the label of $\psi(v)$ in t_2. We also say that t_2 contains t_1. It is denoted by $t_1 \sqsubseteq t_2$. The *support* of tree t, denoted by $\texttt{sup}(t)$, is the fraction of trees in database T containing t.

A labeled tree pattern tp is in the form of $\texttt{t} \Rightarrow \texttt{c}$, where $\texttt{t}$ is an ordered tree and $\texttt{c}$ is a class label. We say that LTP tp_1 is contained by tp_2, denoted by $tp_1 \sqsubseteq tp_2$, if $tp_1.\texttt{t} \sqsubseteq tp_2.\texttt{t}$ and $tp_1.\texttt{c} = tp_2.\texttt{c}$. The *support* of LTP tp is the joint probability of $tp.\texttt{t}$ and $tp.\texttt{c}$, i.e. the fraction of the trees in T containing tp. The *confidence*, denoted by $\texttt{conf}(tp)$, of LSP p is the probability of tp being true. Formally, $\texttt{conf}(tp) = \frac{\texttt{sup}(tp)}{\texttt{sup}(tp.\texttt{t})}$. Support is to measure the generality of tp and confidence is a statement of predictive ability of tp. The higher the confidence of a pattern is, the better it can distinguish between correct sentences and erroneous sentences.

We say that a LTP tp_1 is more **significant** than tp_2 if $(\texttt{conf}(tp_1) > \texttt{conf}(tp_2)) \vee (\texttt{sup}(tp_1) > \texttt{sup}(tp_2) \wedge \texttt{conf}(tp_1) = \texttt{conf}(tp_2))$. Given a user-specified minimum support threshold $minsup$, the top-k LTPs for each tree t in T, denoted by $\texttt{LTP}(k, t)$, is the *most significant* k LTPs in which $\texttt{sup}(tp) \geq minsup$, $tp.\texttt{t} \sqsubseteq t$ and there exists no LTP tp' such that $tp' \notin \texttt{LTP}(k, t)$ and tp' is more significant than any tp in $\texttt{LTP}(k, t)$. For brevity, we will use the abbreviation TkLTPs to refer to the top k LTPs for each tree.

Example 2: Consider a tree database containing three tuples t_1 = (S(NP(DT)(JJ)(NNP))(VP(VBD)(PP)(.)), C), t_2 = (S(NP(NNP))(VP(VBD)(NP)(PP)((IN)(NPB)(.))), C) and t_3 = (S(NP(PRP)(PRP)(NN)(.)), E). Each tuple is a pair of tree and label, where label E denotes erroneous sentence and label C denotes correct sentence. The corresponding trees of the three tuples are shown in Fig 1. One example LTP is tp_1, S(NP(NNP))(VP(VBD)(PP)) $\Rightarrow C$ with support 66.7% and confidence 100%, which is contained in tuples t_1 and t_2. As another example, LTP tp_2, S(NP)$\Rightarrow C$ with support 66.7% and confidence 66.7%. LTP tp_1 is a better indication of class C than tp_2. LTP tp_1 is a top-1 LTP for tuples t_1 and t_2. S(NP(DT)) $\Rightarrow C$ is not a top-1 LTP for tuple t_1 since its support is 33.3% and confidence 100%, which is less significant than tp_1. For tuple t_3 S(NP(PRP)) $\Rightarrow E$ is a top-1 LTP with support 33.3% and confidence 100%. □

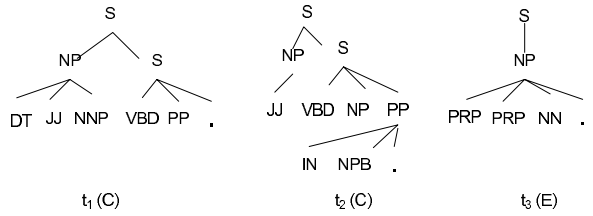

Figure 1: Running Example of Parse Trees

Algorithm MineLTP(T, r, $minsup$, k)
Input: Database T, tree r, minimum support $minsup$ and k.
*/*Initially, T is the tree database, r is empty tree/**
Output: The TkLTPs, denoted by $\texttt{LTP}(k, t)$, for each tree t in T.

1. for each tree r_i generated by extending r with i, where i will be a rightmost occurrence node of r_i, we count the supports of r_i in both erroneous and correct sequences, denoted by $\texttt{sup}_E(r_i)$ and $\texttt{sup}_C(r_i)$, respectively;
2. for each r_i, */*E: Erroneous, C: Correct/**
   for each t, $r_i \sqsubseteq t$, if LTP $r_i \Rightarrow E$ (or $r_i \Rightarrow C$) is more significant than LTPs in $\texttt{LTP}(k, t)$,
   update $\texttt{LTP}(k, t)$ with the new LTP;
   Let $mconf_E$ (resp. $mconf_C$) be the minimum confidence for all $\texttt{LTP}(k, t)$, $r_i \sqsubseteq t$ and t is labeled with E (resp. C);
   if $\texttt{sup}_E(r_i)$ (resp. $\texttt{sup}_C(r_i)$) is larger than $minsup$ and further extension could generate LTPs with confidence larger than $mconf_E$ (resp. $mconf_C$),
   let D_{r_i} be the set of trees in D containing r_i;
   call $\texttt{MineLTP}(D_{r_i}, r_i, minsup, k)$;

Figure 2: The MineLTP Algorithm

Mining top-k LTPs of each tree (TkLTPs): One naive way of mining TkLTPs is to first find the set P of patterns satisfying a user specified minimum support $minsup$ using the previous algorithms for mining frequent subtrees, e.g., (Zaki & Aggarwal 2003), then for each tree t in database find the set of patterns contained in t from P, and return the most significant k patterns for each t. This solution is effective, but inefficient. Instead of mining all the patterns satisfying $minsup$ and then finding TkLTPs for each tree, we directly mine the final set TkLTPs for each tree. The idea is to make use of TkLTPs to generate dynamic minimum confidences to prune search space although we do not have user-specified confidence threshold. We maintain the dynamic minimum confidence by tracking the lowest confidence among the current TkLTPs for each training tree during the mining process. The framework of our algorithm is similar to that of algorithm (Zaki & Aggarwal 2003): it first discovers tree patterns of size 1 (containing 1 node), then extends them to generate tree patterns of size 2 (containing 2 nodes), and so on; this process continues until further extension will not generate TkLTPs. Two effective pruning strategies are integrated into the framework: 1) if a LTP does not satisfy *minimum support* constraint, any extension of this LTP cannot have the *minimum support*; 2) if further extensions cannot generate patterns with higher minimum confidence, we can stop. We outline the algorithm $\texttt{MineLSP}$ in Fig. 2. Details are ignored due to space limitation.

Mining *Labeled Sequential Patterns* (LSPs)

We next introduce Labeled Sequential Patterns (LSPs). The features captured by LTPs and LSPs for erroneous (or correct) sentences may be similar. LSPs and LTPs, however, can also be mutual complementary. We illustrate this using several erroneous sentences. 1) Erroneous sentences that can be captured by LSPs but not LTPs. a) Sentence "*He visit here yesterday.*"contain LSP <VB, yesterday> $\Rightarrow E$; b) Sentence "*He was so effect that many person know he.*" contains <so, NN, that>$\Rightarrow E$; c) Sentence"*although he likes it, but he can't buy it.*" contains "<although, but>"$\Rightarrow E$. 2) Erroneous sentences that can be captured by LTPs by not LSPs: a) Sentence "*He gives me very impress.*"contains LTP (NP(JJ)(VB)) $\Rightarrow E$; b) Sentence "*Will you want to buy them.*"contains LTP (SINV(VBZ)(NPB)(.))$\Rightarrow E$.

Labeled Sequential Patterns (LSPs): Let I be a set of items and L be a set of class labels. Let D be a sequence database in which each tuple is composed of a list of items in I and a class label in L. We define that a sequence $s_1 =< a_1, ..., a_m >$ is *contained* in a sequence $s_2 =< b_1, ..., b_n >$, denoted by $s_1 \preceq s_2$, if there exist integers $i_1, ...i_m$ such that $1 \leq i_1 < i_2 < ... < i_m \leq n$ and $a_j = b_{i_j}$ for all $j \in 1, ..., m$. Note that it is not required that s_1 appears continuously in s_2. The *support* of sequence s, denoted by $\texttt{sup}(s)$, is the fraction of tuples in database D containing s.

A labeled sequential pattern p is in the form of $\texttt{s} \Rightarrow \texttt{c}$, where $\texttt{s}$ is a sequence and $\texttt{c}$ is a class label. We say that LSP p_1 is contained by p_2, denoted by $p_1 \sqsubseteq p_2$, if $p_1.\texttt{s} \sqsubseteq p_2.\texttt{s}$ and $p_1.c = p_2.c$. Following the definitions of *support*, *confidence*, *significant* and *top-k* for LTPs, we can derive the corresponding definitions for LSPs. Due to space limitation, we do not give them.

Generating Sequence Database: We apply Part-Of-Speech (POS) tagger to tag each training sentence while keeping function words[2] and time words[3]. After the processing, each sentence together with its label becomes a database tuple. The function words and POS tags play important roles in both grammars and sentence structures. In addition, the time words are key clues in detecting errors of tense usage. We use the method in (Sun *et al.* 2007) to process each sentence. For example, after the processing, the sentence "*In the past, John was kind to his sister*" is converted into "*In the past, NNP was JJ to his NN*", where the words "*in*", "*the*", "*was*", "*to*" and "*his*" are function words, the word "*past*" is time word, and "*NNP*", "*JJ*", and "*NN*" are POS tags.

Mining T*k*LSPs with Constraints: The length of the discovered LSPs is flexible and they can be composed of contiguous or distant words/tags. To reduce generating spurious patterns, we also employ distance constraints to reduce noisy LSPs. More specifically, the "containment" relationship between two sequences is refined as follows: if two adjacent items in a sequence s_1 are function word and POS tag, in addition to the conditions in the previous definition of "s_1 is contained in s_2", the function word and POS tags in s_2 needs to adjacent. Note that the distance between other items is not limited, e.g. distance between function words since two distant function words may represent meaningful sentence structure. By following the algorithm in Fig 2, one can derive an algorithm for mining T*k*LSPs.

[2]http://www.marlodge.supanet.com/museum/funcword.html

[3]http://www.wjh.harvard.edu/%7Einquirer/Time%40.html

Classification models

On one hand, the discovered LTPs and LSPs can be used as binary features to represent the erroneous/correct sentences, in which any learning algorithm can be used. In this paper, support vector machine (SVM) and Naive Bayesian (NB) classification models are employed for the purpose.

On the other hand, we also propose a pattern based classification (PBC) method using the discovered T*k*LTPs and T*k*LSPs. We build a main classifier using the set of most significant patterns for each training sentence. We then build *k* -1 standby classifiers using the other LTPs and LSPs to classify sentences that cannot be handled by the main classifier.

Building Classifier: We next compute the set of combined top-k patterns from T*k*LTPs and T*k*LSPs. Let TP_j denote the set of LTPs that appear as a top-j LTP in at least one of the training sentences, and SP_j denote the set of LSPs that appear as a top-j LSP in at least one of the training sentences. According to the definition of *significant*, we compute the set, P_1, of combined top-1 patterns for each training sentence from the set TP_1 of top-1 LTPs and the set SP_1 of top-1 LSPs. Let TSP_2 be $((TP_1 \cup SP_1) \setminus P_1) \cup TP_2 \cup SP_2$. We generate the top-$j$ ($j = 2, ..., k$) from TSP_j, where TSP_j = $(TSP_{j-1} \setminus P_{j-1}) \cup TP_j \cup SP_j$, j = 3, ..., k.

We will thus have k sets of patterns P_1,...,P_k. The k sets of patterns are used to build k classifiers CL_1,...,CL_k with CL_j being built from P_j. We call CL_1 the main classifier and CL_2,...,CL_k standby classifiers. Besides main and standby classifiers, we set a default class if none of them can classify a sentence. This default class is set as the majority class of the remaining training data.

Prediction: Given a test data t, we will go through CL_1 to CL_k in that order to see if t can be handled by any of these classifiers. The first classifier that has matching rules for t will determine its class. If the test data cannot be handled by any of the classifiers, then the default class will be used for the prediction.

PBC matches a test sentence with all patterns of an individual classifier (the main classifier or individual standby classifiers) and makes a decision by aggregating voting scores. We use a voting score for a pattern p with class label c_i by considering both confidence and support as follows:

$$S(p) = \texttt{conf}(p) * \texttt{sup}(p) * |D|/d_{c_i},$$

where d_{c_i} is the number of training sentences of the class c_i and $|D|$ is the total number of training sentences. Note that $0 \leq S(p) \leq 1$. By summing up the scores of all patterns in each class c_i, we get a score $S^{c_i}_{norm}$ for normalization purpose. Given a test sentence t, we suppose that t satisfies the following m_i patterns of class c_i: $p(t)_1, ..., p(t)_{m_i}$. The classification score of class c_i for test t is calculated as:

$$Score(t)^{c_i} = (\sum_{i=1}^{m_i} S(p(t)_i))/S^{c_i}_{norm}.$$

We make a prediction for test t with the highest classification score.

Dataset	Type	Source	Number
JC	(+)	the Japan Times newspaper and Model English Essay	16,857
	(-)	HEL (Hiroshima English Learners' Corpus) and JLE (Japanese Learners of English Corpus)	17,301
CC	(+)	the 21st Century newspaper	3,200
	(-)	CLEC (Chinese Learner Error Corpus)	3,199

Table 1: Corpora ((+): correct; (-): erroneous)

Experimental Evaluation

Experimental Setup

Classification Methods: One method is the pattern based classifier (PBC). The other two classification models are SVM[4] and NB. For SVM, we vary the linear kernel and polynomial kernel and report the better results.

Data: We collect two datasets from different domains, Japanese Corpus (JC) and Chinese Corpus (CC). Table 1 gives the details of our corpora. Note that our data does not consist of parallel pairs of sentences (one error sentence and its correction). The erroneous sentences include a wide range of grammatical errors, but not spelling errors.

We mine T*k*LTPs and T*k*LSPs for each sentence. We empirically set *minimum support* at 0.1% and k at 3.

Metrics: We report the overall accuracy. To show the classification performance for both classes, we also report the `precision`, `recall`, and `F-score` for correct and erroneous sentences, respectively.

Experimental Results

The Effectiveness of Classification Methods: The experiment is to evaluate three classification methods, pattern-based classification (PBC), SVM and NB. The discovered S*k*LTPs, S*k*LSPs, and their combination are used as features, respectively. The experimental results are obtained through 10-fold cross-validation, and are given in Table 2. We can see that the performance of PBC is consistently better than NB and is comparable with SVM. When both S*k*LTPs and S*k*LSPs are employed, PBC performs better than SVM.

In addition, the complexity together with the distance model of SVM is much more complicated than PBC classifier and it is hard to derive understandable explanation of any decision made by SVM. This limits the practical use of SVM in erroneous/correct sentences. In contrast, the PBC classifier is very intuitive and easy to understand. The discovered LTPs and LSPs themselves are intuitive and meaningful, and characterize significant linguistic features. We discovered 18,367 LTPs in JC data and 12,186 LTPs in CC data; we discovered 26,309 LSPs in JC data and 16,295 LSPs in CC data.

We next give some examples of discovered LTPs. 1) Erroneous labels: a) (VP(VB)(ADJP)(NP) (verb phrase containing verb, adjective phrase and noun phrase) with *support* 0.178% and *confidence* 94.12%, and b) (S(NPB(ADJP(JJR)))) (sentence containing a base noun phrase containing adjective phrase containing comparative adjective) with *support* 0.243% and *confidence* 100%; 2) Correct labels: a) (S(NP(NNP))(VP(VBD)(ADVP))) with *support* 0.434% and *confidence* 100%, and b) (VP(VBG)(NPB(DT)(JJ)(NN))) with *support* 0.815% and *confidence* 85.71%.

We also give some examples of discovered LSPs. 1) Erroneous labels: a) <*a*, NNS> (a + plural noun) with *support* 0.39% and *confidence* 85.71%, b) <*to*, VBD> (to + past tense format) with *support* 0.11% and *confidence* 84.21%, and c) <the, more, the, JJ> (the more + the + base form of adjective) with *support* 0.19% and *confidence* 0.93%; 2) Correct labels: a) <NN, VBZ> (singular or mass noun + the 3^{rd} person singular present format) with *support* 2.29% and *confidence* 75.23%, b) <have, VBN, since> (have + past participle format + since) with *support* 0.11% and *confidence* 85.71%, and c) <*one*, *of*, NNS> (one of + plural noun) with *support* 0.25% and *confidence* 98.91%.

Comparing with other Methods: It is difficult to find benchmark methods to compare with our technique because, as discussed in Section 2, existing methods often require error tagged corpora or parallel corpora, or focus on a specific type of errors. In this paper, we compare our technique with the grammar checker of Microsoft Word03 and the ALEK (Chodorow & Leacock 2000) method used by ETS. ALEK is used to detect inappropriate usage of specific vocabulary words. Note that we do not consider spelling errors. Due to space limitation, we only report the `precision`, `recall`, `F-score` for erroneous sentences, and the overall accuracy.

As can be seen from Table 3, our method outperforms the other two methods in terms of overall accuracy, F-score, and recall, while the three methods achieve comparable precision [5]. We realize that the grammar checker of Word is a general tool and the performance of ALEK (Chodorow & Leacock 2000) could be improved if larger training data is used. We find that Word and ALEK usually cannot find sentence structure errors, e.g., "*The more you listen to English, the easy it becomes*." contains the discovered LSP <*the, more, the, JJ*> ⇒ *Error*.

Cross-domain Results: To see the performance of our method on cross-domain data from writers of different first-language backgrounds, we use the JC dataset (resp. CC dataset) for training while the CC dataset (resp. JC dataset) is used as test data. The results are shown in Table 4. This experiment shows that the performance is not as good as that on data of the same first-language background, although it still outperforms Word and ALEK. The reason is that the mistakes made by Japanese and Chinese are different, thus the learning model trained on one data does not fit very well on the other data. Moreover, we also merge the JC dataset and CC dataset, and conduct 10-fold cross-validation test on the merged dataset. The result is similar to the results on Dataset CC in Table 2.

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

[5] Similar to most classifiers, PBC can easily make a compromise between precision and recall by setting a threshold.

Dataset	Feature	Method	A(%)	(-)F(%)	(-)R(%)	(-)P(%)	(+)F(%)	(+)R(%)	(+)P(%)
CC	T*k*LTP	PBC	74.45	74.53	75.72	73.37	74.37	73.21	75.56
		NB	70.71	71.28	72.18	70.4	70.11	69.21	71.03
		SVM	73.12	68.57	58.21	83.45	76.51	88,27	67.52
	T*k*LSP	PBC	78.86	79.12	81.15	77.20	78.59	76.64	80.66
		NB	74.92	74.01	71.57	76.68	75.75	78.25	73.44
		SVM	77.84	79.30	85.86	73.68	76.16	70.01	83.50
	T*k*LTP+T*k*LSP	PBC	80.44	79.47	76.68	82.47	81.33	84.11	78.72
		NB	75.01	74.59	74.00	75.18	75.43	76.01	74.86
		SVM	78.55	77.33	74.12	80.84	79.64	82.87	76.66
JC	T*k*LTP	PBC	72.24	72.67	74.76	70.69	71.79	69.78	73.93
		NB	70.94	73.58	81.54	67.04	67.72	60.50	76.90
		SVM	73.16	76.92	90.34	66.97	67.97	56.35	85.62
	T*k*LSP	PBC	75.12	73.83	70.83	77.11	76.30	79.35	73.48
		NB	72.39	77.36	95.26	65.13	64.61	49.94	91.49
		SVM	76.43	74.75	69.66	80.65	77.91	83.23	73.22
	T*k*LTP+T*k*LSP	PBC	77.21	79.03	86.03	73.09	75.04	83.08	68.42
		NB	73.35	77.10	90.41	67.21	68.13	56.55	85.68
		SVM	76.66	75.97	72.67	76.59	77.30	80.77	74.12

Table 2: The Experimental Results of three Classification Methods (A: overall accuracy; (-): erroneous sentences; (+): correct sentences; F: F-score; R: recall; P: precision)

Dataset	Method	A	(-)F	(-)R	(-)P
CC	PBC	80.44	79.47	76.68	82.47
	Word	58.47	32.02	19.81	84.22
	ALEK	55.21	22.83	13.42	76.36
JC	PBC	77.21	79.03	86.03	73.09
	Word	58.87	33.67	21.03	84.73
	ALEK	54.69	20.33	11.67	78.95

Table 3: The Comparison Results

Dataset	Feature	A	(-)F	(-)R	(-)P
CC(Train)+ JC(Test)	T*k*LTP	69.08	77.84	88.69	69.35
	T*k*LSP	74	80.28	86.43	74.95
	T*k*LSP+T*k*LTP	75.20	76.58	80.75	72.83
JC(Train)+ CC(Test)	T*k*LTP	65.32	70.76	87.67	59.32
	T*k*LSP	68.82	65.87	59.73	73.43
	T*k*LSP+T*k*LTP	69.55	67.02	61.36	73.82

Table 4: The Cross-domain Results of PBC

Conclusions and Future Work

This paper proposed a new approach to identifying erroneous/correct sentences. The approach mined T*k*LTPs and T*k*LSPs for each training sentence and used the discovered patterns to build classification models. Empirical evaluation on diverse data demonstrated the effectiveness of our techniques.

In the future, we plan to work on providing suggested correction feedback for ESL learners. The discovered LTPs and LSPs indicate the specific errors in the erroneous sentence. This makes it possible for us to mine their corrections from parallel corpora, and thus to suggest corrections.

References

Bender, E. M.; Flickinger, D.; Oepen, S.; Walsh, A.; and Baldwin, T. 2004. Arboretum: Using a precision grammar for grammmar checking in call. In *Proc. InSTIL/ICALL Symposium on Computer Assisted Learning*.

Brockett, C.; Dolan, W.; and Gamon, M. 2006. Correcting esl errors using phrasal smt techniques. In *ACL*.

Burstein, J.; Kukich, K.; Wolff, S.; Lu, C.; Chodorow, M.; Braden-Harder, L.; and Harris, M. D. 1998. Automated scoring using a hybrid feature identification technique. In *Proc. ACL*.

Chodorow, M., and Leacock, C. 2000. An unsupervised method for detecting grammatical errors. In *NAACL*.

Collins, M. 1997. Three generative, lexicalised models for statistical parsing. In *ACL*.

Cong, G.; Tan, K.-L.; Tung, A. K.; and Xu, X. 2005. Mining top-k covering rule groups for gene expression data. In *SIGMOD*.

Corston-Oliver, S.; Gamon, M.; and Brockett, C. 2001. A machine learning approach to the automatic evaluation of machine translation. In *Proc. ACL*.

Foltz, P.; Laham, D.; and Landauer, T. 1999. Automated essay scoring: Application to educational technology. In *EdMedia '99*.

Gamon, M.; Aue, A.; and Smets, M. 2005. Sentence-level mt evaluation without reference translations: Beyond language modeling. In *Proc. EAMT*.

Heidorn, G. E. 2000. *Intelligent Writing Assistance*. Handbook of Natural Language Processing. Robert Dale, Hermann Moisi and Harold Somers (ed.). Marcel Dekker.

Izumi, E.; Uchimoto, K.; Saiga, T.; Supnithi, T.; and Isahara, H. 2003. Automatic error detection in the japanese learners' english spoken data. In *Proc. ACL*.

Jindal, N., and Liu, B. 2006. Identifying comparative sentences in text documents. In *SIGIR*.

Liu, B.; Hsu, W.; and Ma, Y. 1998. Integrating classification and association rule mining. In *KDD*.

Michaud, L. N.; McCoy, K. F.; and Pennington, C. A. 2000. An intelligent tutoring system for deaf learners of written english. In *Proc. ACM Conference on Assistive Technologies*.

Nagata, R.; Kawai, A.; Morihiro, K.; and Isu, N. 2006. A feedback-augmented method for detecting errors in the writing of learners of english. In *Proc. ACL*.

Shi, Y., and Zhou, L. 2005. Error detection using linguistic features. In *HLT/EMNLP*.

Sun, G.; Liu, X.; Cong, G.; Zhou, M.; Xiong, Z.; Lin, C.-Y.; and Lee, J. 2007. Detecting erroneous sentences using automatically mined sequential patterns. In *ACL*.

Zaki, M. J., and Aggarwal, C. C. 2003. Xrules: an effective structural classifier for xml data. In *KDD*.