Journal of the Korean Society of Marine Engineering, Vol. 37, No. 7, pp. 760~770, 2013 (ISSN 1226-9549)
http://dx.doi.org/10.5916/jkosme.2013.37.7.760

# A comparison of grammatical error detection techniques for an automated english scoring system

Songwook Lee[1] · Kong Joo Lee[†]

(Received September 11, 2013 ; Revised October 1, 2013 ; Accepted October 17, 2013)

**Abstract:** Detecting grammatical errors from a text is a long-history application. In this paper, we compare the performance of two grammatical error detection techniques, which are implemented as a sub-module of an automated English scoring system. One is to use a full syntactic parser, which has not only grammatical rules but also extra-grammatical rules in order to detect syntactic errors while paring. The other one is to use a finite state machine which can identify an error covering a small range of an input. In order to compare the two approaches, grammatical errors are divided into three parts; the first one is grammatical error that can be handled by both approaches, and the second one is errors that can be handled by only a full parser, and the last one is errors that can be done only in a finite state machine. By doing this, we can figure out the strength and the weakness of each approach. The evaluation results show that a full parsing approach can detect more errors than a finite state machine can, while the accuracy of the former is lower than that of the latter. We can conclude that a full parser is suitable for detecting grammatical errors with a long distance dependency, whereas a finite state machine works well on sentences with multiple grammatical errors.

**Keywords:** grammatical errors, automated English scoring system, error detection, finite-state transducer, syntactic parser

## 1. Introduction

Grammatical error detection techniques are indispensable and extremely useful in building various applications such as automated writing scoring systems, intelligent tutoring systems for second-language learners, and conventional proofing systems **[1]-[5]**. Approaches of detecting grammatical errors in sentences can be classified into two major categories. One is a rule-based approach, and the other is a data-driven approach. A large collection of sentences which are written by target test-takers is necessary for a data-driven approach. Generally speaking, at a very early development stage, it is not easy to collect a corpus large enough to build a reliable data-driven error detection module. In addition, if a system should be able to give some feedback to test-takers about their writings, a rule-based approach is preferred to a data-driven approach. In this paper, we adopt a rule-based approach in implementation of a grammatical error detection module.

There are also several ways to implement a rule-based error detection module. The most widely known approach is to detect errors by adopting a full syntactic parser which uses hand-crafted grammar

† Corresponding Author: Dept. of Information Communications Engineering, Chungnam National University, 99 Daehak-ro, Yuseong-gu, Daejeon, 305-764, Korea, E-mail: kjoolee@cnu.ac.kr, Tel: 042-821-5662
1 Department of Computer Science and Information Engineering, Korea National University of Transportation, E-mail: leesw@ut.ac.kr, Tel: 043-841-5464

rules. A syntactic parser can analyze an input sentence by using its grammar rules, and can detect syntactic errors while parsing. Grammar rules that a syntactic parser uses include not only grammatical rules but also extra-grammatical rules. Another possible approach is to adopt a finite state technique to detect syntactic errors from an input sentence.

Finite-state techniques are used in a wide range of domains, including pattern matching, pattern recognition, speech processing, optical character recognition **[6]-[8]**. Those techniques have recently improved the computation efficiently for a wide variety of natural language processing tasks ranging from morphological analysis to phonetic and speech processing **[9]**. No matter how much finite-state techniques improve, they are clearly less powerful than full parsers armed with context-free grammars. However, if context-free parsing is not performance-effective when applied to real-world text, then an efficient text processor might make use of weaker language models, such as regular or finite-state grammars **[9]**.

In our previous research, we have implemented an automated English writing scoring system **[10]** which have a full syntactic parser in order to check grammaticality of an input sentence and detect syntactic errors from sentences, if exist. The system's target test-takers are Korean middle school students at a beginning writing skill level, so their writings not only considerably vary in quality but also are quite messy. We realized that it is not easy to build a full syntactic parser to analyze those sentences correctly.

For example, we can write extra-grammatical rules to recognize the following sentence (1) with some efforts. However, it does not seem easy to write extra-grammatical rules for the sentence (2), which is assumed to be written in Korean word-order. Even if we can write such rules, they may burden a syntactic parser with a severe ambiguity problem. Generally a full parser leaves the sentence (2) unanalyzed and thus any error is not detected from it. On the other hand, a simple regular rule for recognizing the sequence (2) is possible, and its corresponding finite state machine can detect an error from the sequence (2).

(1) I am go to home now.
(2) He her class very tall.

The accuracy of detecting errors in a full syntactic parser heavily depends on the performance of a syntactic parser. When a syntactic parser builds not a whole parse tree for an input, but a few of fragmental phrases instead, grammatical errors in a sentence may become invisible or sometimes irrelevant errors may be wrongly identified. A full syntactic parser is prone to fail to construct a complete parse tree for a messy sentence with multiple errors. Therefore a syntactic parsing approach results in a low performance in detecting grammatical errors from quite erroneous sentences.

In this paper, we compare deliberately a finite-state approach with a full parsing approach in handling grammatical errors. Both implementations are not initially designed to achieve the same task, however, since their target test-takers and writing objectives are very similar, the direct comparison between two implementations is meaningful. By doing this, we can figure out the strength and the weakness of each approach. Also we can decide which approach is superior to some tasks and which approach is not suitable for some purposes.

We first explore related studies in Section 2, and we present both a full parsing approach and a finite-state machine approach in Section 3. Grammatical errors detected by both approaches are introduced in Section 4, and evaluation results are described in Section 5, followed by the conclusion in Section 6.

## 2. Related Studies

There exist two main approaches to the problem of detecting grammatical errors from sentences. One is a rule-based approach and the other is a data-driven approach. Rule-based approaches have been shown to be useful in detecting certain kinds of syntactic errors in the learners' writing **[11]**. Data-driven approaches build statistical models to identify sentences containing errors **[12]**.

Hashemi et al. proposes a grammar checker for Swedish based on a finite state transducer **[6]**. Their approach for finding errors involves developing automata that represent two positive grammars with varying degree of detail and then subtracting the detailed one (narrow grammar) from the general one (broad grammar). They use the lexical-prefix-first method, i. e. parsing first from left margin of a phrase to the head and then extending the phrase by adding on complements. The error finder corresponds to the difference between the two grammars, broad and narrow. They can detect agreement errors in noun phrases, verb selection phenomena and local word order phenomena by using this approach. Also some attempts were done to detect missing sentence boundaries, starting with clause and verb subcategorization and trying to make use of the valency information stored in the lexicon of the system. However, they do not report its performance and coverage in detail.

The reference **[11]** proposes a data-driven approach to detecting erroneous sentences by integrating pattern discovery with supervised learning models. They 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. They propose labeled sequential patterns (LSP) to effectively characterize the features of correct and erroneous sentences. In order to automatically generate LSP, an input is tagged with POS tags. Existing frequent sequential pattern mining algorithms (e.g. **[12]**) use minimum support threshold to mine frequent sequential patterns whose support is larger than the threshold. Mining LSPs is non-trivial since its search space is exponential, although there have been a host of algorithms for mining frequent sequential patterns. Each discovered LSP forms a binary feature as the input for classification model. They evaluated the performance of their techniques with support vector machine (SVM) and Naive Bayesian (NB) classification models. An evaluation is done on the collection of English essay written by Chinese and Japanese, which is classified into correct one and erroneous one in advance. 6309 and 3742 LSPs are discovered in this research from Japanese corpus and Chinese corpus, respectively. Their approach achieves 81.5% accuracy when they use the features from LSPs as well as other additional features in distinguishing a correct sentence from an erroneous sentence.

The research **[13]** is mainly interested in detecting/correcting the most frequently occurring error types for nonnative English language learner -- preposition and article errors. As the more error-prone input of nonnative speakers often prevents parser-based analyzers from successfully identifying potential problems in nonnative writing, **[13]** use a statistical techniques in which error detection modules are based-on machine learning method. They implemented preposition error detection/correction modules, and article error detection/correction module as a form of classifier.

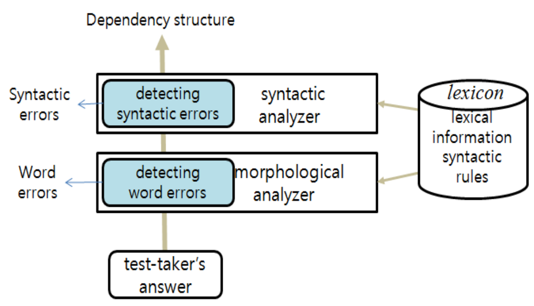

**Figure 1** The overview of automated English sentence scoring system **[10]**

| | |
|---|---|
| #RULE | NP:np0 → DET:det1 NP:np1 |
| #CONDITION | chk_dn_agreement(det1, np1, DT_AGR_ERR) |
| #CONDITION | chk_cons_vowel(det1, np1, DET_CV_ERR) |
| #ACTION | assign_np_from_det_np(np0) |

**Figure 2** An example of a syntactic rule.

The feature set for these classifier is four tokens to the left and four tokens to the right, and six part-of-speech tags to the left and to the right for the potential locations that error occurs. Since the suggested correction made by correction module is often-over-generated, a language model which is trained by using giga word corpus is adopted in order to filter out the over-generated suggestions. They used a corpus which consists of about 2,550,000 sentences in order to train the module. An evaluation is achieved on native text. They reported that accuracy of the article error detection is about 89.19% and that of the preposition error detection is about 88.95% accuracy.

In this research, we adopt a rule-based approach in detecting grammar errors from sentences. There can be various implementations for a rule-based error detection approach. We have built two different error detection modules. One is to use a full syntactic parser, and the other is to adopt a finite state machine. In this paper, our major concern is which approach is better than the other, and what advantages and disadvantages each approach has.

## 3. Grammatical Error Detection Modules

### 3.1 Grammatical error detection by a full syntactic parser

As shown in **Figure 1**, the morpho-syntactic analysis **[10]** can identify word errors as well as syntactic errors if a test-taker's answer contains errors. A syntactic analyzer is implemented to use augmented context free syntactic rules which can detect not only a grammatical sentence but also a ungrammatical sentence. The output of the morpho-syntactic analyzer is a dependency structure of an input sentence with a set of words error and syntactic errors detected from an input sentence, if they exist.

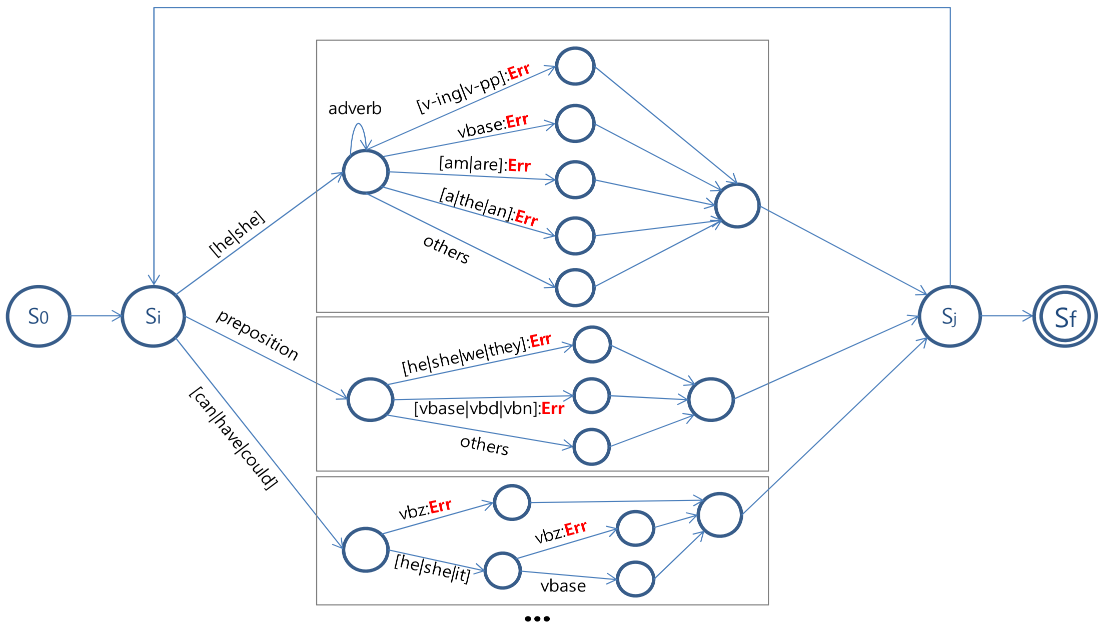

**Figure 3** An example of a part of FST to detect syntactic errors.

**Figure 2** shows an example of the augmented syntactic rule. A rule consists of a RULE field, zero or more CONDITION fields, and zero or more ACTION fields. The field CONDITION is implemented as a procedure to check the constraints on the constituents. A rule can be applied to an input sentence only when every CONDITION is satisfied. When the rule is fired, ACTIONs are activated. The ACTIONs are also implemented in the form of a procedure. Some of the CONDITION procedures are designed to handle ungrammatical inputs as well as grammatical ones. For instance, 'a flower' which is grammatical, can be analyzed by the rule depicted in Figure 2. In addition, 'a flowers', which has a number agreement error, is also covered by the same rule. The first procedure checks number agreement between the determiner and the following noun. In this case, the procedure detects an agreement error ('DN AGR ERROR'), saves the error information, and continues the analysis. The second CONDITION checks if the form of the determiner is 'an' when the following noun begins with a vowel. After checking the CONDITION procedures, ACTION procedures are activated so as to inherit the features from the head of the constituents. The total number of syntactic rules implemented in the system is 316; 235 for dealing with grammatical sentences, 26 for handling ungrammatical sentences, and 55 for covering both grammatical and ungrammatical sentences. 133 and 17 procedures are implemented for CONDITION and ACTION, respectively.

### 3.2 Finite State Transducer for detecting grammatical errors

A finite-state transducer (FST) operates on an input sentence, and generates outputs - errors detected on a input sentence. We build a FST to recognize an erroneous substring. **Figure 3** shows a simplified version of a FST in which a notation followed by ':' on an edge indicates a recognized error. (**Figure 3** is a part of FST; A real implementation is far bigger and more complex than that of **Figure 3**.) A FST consumes one word of an input string at a time and traverse from a state to another according to an input word. When a FST reaches the end of an input string, its processing ends. In **Figure 3**, edges without a label are assumed to be an epsilon-transition that can change a state without consuming any input word. For example, the sentence "Can she do this?" can be processed in this FST without an error. However, the sentence "He really make the world." is processed by the FST with detection of errors.

In order to detect syntactic errors from an input sentence by using the FST, an input sentence is first processed by a morphological analyzer. Each word is represented by triple information of [surface_string, stem, part_of_speech[1]] by virtue of a morphological analysis, and the sequence of the triple information is an input of the FST. Therefore, one of triple information or all three together can be referenced in the FST.

While one advantage of this approach is that any erroneous substring can be implemented in a FST easily, its drawback is that a FST can recognize only one error from a given substring when several errors occur together in the substring. Also, since a FST is designed to process an input in a sequential manner, it cannot detect an error which violates an agreement between two words with a long distance. A full parsing approach described in Sec 3.1, by contrast, can handle an error with a long distance dependency. However, a full parser is liable to fail in the analysis of a sentence when the sentence includes multiple syntactic errors, and furthermore it tends to generate an error-free parse tree even for a sentence with errors, if parsing succeeds.

## 4. Grammatical Errors

This section describes a kind of grammatical errors

[1] The tag set of part-of-speech used in this paper is Penn Treebank POS tag set.

dealt in this research. We have two grammatical error detection modules, one of which is a full parser, and the other is a finite state transducer. In the followings, grammatical errors that can be detected by both modules are first introduced, those by the only full parser are followed, and lastly, those by the FST only are listed. We do not handle word-level errors in this paper even though both systems can detect word-level errors too. As our test-takers are Korean middle school students, their writing skills are far from perfect and their writing is somewhat short and simple.

**Table 1** shows the grammatical errors that can be detected by both the full parsing approach and the FST approach. As you can see, they are most frequently occurring errors in writing which Korean middle school students can make. In this paper, we exclude the description of the errors that occurred less than 5 times in our corpus. The grammatical errors shown in **Table 1** can be easily detected by both approaches since they occur within a small range of a sentence.

Even though both modules can handle the same grammatical errors in **Table 1**, their actual detection coverages may be different. For example, the finite state transducer can barely detect a subject-verb agreement error when there is a big gap between a subject and its corresponding verb, while the full parser can detect it since the full parser can reduce the gap by assembling a sequence of words into phrases.

**Table 2** shows the grammatical errors which can be identified by only the full parser. On the whole, the coverage of the errors in Table 2 is broader than that of errors in **Table 3**, which are handled by the FST only. Since these errors are characterized by having a broad range in an input sentence, a full parser can detect these errors in a higher level of phrases during parsing if it is a bottom-up parser. Therefore, when a full parser fails to construct a complete tree for an input sentence, it has a slim chance to detect these errors correctly. Accordingly, a full parser should be robust in order to detect this kind of errors successfully. We found that a full parser is apt to fail to build a complete parse tree for a sentence with multiple errors, so most of the errors in **Table 2** are not easy to be identified correctly in this case.

**Table 3** shows the grammatical errors which are handled by the FST module. It is not easy to implement grammar rules to be able to handle these errors

**Table 1** The grammatical errors processed in both approaches.

| ID | kind of error / examples |
|---|---|
| PF01 | determiner-noun agreement error<br>(ex) [a sheep] [a bits] [this days] [two girl] |
| PF02 | determiner consonant/vowel form error<br>(ex) [a apple] [an piano] |
| PF03 | extra determiner error<br>(ex) This city is born [the my] grandfather.<br>There is see [the a] lot. |
| PF04 | preposition error<br>(ex) She went [to] there.<br>She looks [like] healthy.<br>It is different [with] that. |
| PF05 | modifier comparative form error<br>(ex) [more prettier] |
| PF06 | modifier superlative error<br>(ex) [most tallest] |
| PF07 | subject verb agreement error<br>(ex) [She tell] me.<br>The [earth are] big than moon. |
| PF08 | verb type error<br>(ex) Can you [come my birthday?]<br>She [looks health.]<br>He [arrived the center.]<br>I [want help] many people.<br>We [enjoy listen] music. |
| PF09 | verb form error<br>(ex) He [is speak] English.<br>Who [is call] me?<br>He [spoken] French.<br>He [can speaks] English.<br>He [has went] home. |
| PF10 | verb-be missing error<br>(ex) I [would happy] if I were a bird.<br>I [learning] English 7 years. |

**Table 2** The grammatical errors processed only in a full parser.

| ID | kind of error / examples |
|---|---|
| P01 | adverb type error<br>(ex) He is [much famous] than his father. |
| P02 | auxiliary verb type error<br>(ex) I [would better] go home. |
| P03 | noun number error<br>(ex) He is [more intelligent than any other students.] |
| P04 | verb-be complement agreement error<br>(ex) [He is the tallest students.] |
| P05 | pronoun type error<br>(ex) It is different from [he opinion.] |
| P06 | negative adverb usage error<br>(ex) I [don't like it, too.] |
| P07 | adverb position error<br>(ex) [He is more popular in Korea than any other actor.] |
| P08 | relative pronoun type error<br>(ex) I will order [the same what you eat.] |
| P09 | relative clause error<br>(ex) She is [the teacher who she came.] |
| P10 | relative pronoun missing error<br>(ex) She is [a teacher come] to my school last week. |
| P11 | conjunction missing error<br>(ex) [The prices get lower, people buy more.] |

in a full parser because these grammar rules cause much syntactic ambiguities. On the other hand, the FST can deal with these error patterns without any problems since it can be implemented without considering structures the rules fire on.

The errors F02 and PF10 have common since both deal with a sentence without a main verb. However, while the error PF10 deals with only a sentence without 'be' followed by an adjective or verb's present or past particle form, F02 deals with sentences without a main verb except 'be'.

## 5. Evaluation

### 5.1 Evaluation Setup

**Table 3** The grammatical errors processed only in the FST.

| ID | kind of error / examples |
|---|---|
| F01 | incomplete sentence (ended with a conjunction/possessive pronoun/determiner)<br>(ex) [This is because] [it is my]<br>[she is explain the] |
| F02 | sentence without main verb<br>(ex) [With me!]<br>[The palace one of the old building in Korea.] |
| F03 | redundant use of the same string<br>(ex) The woman with long hair is [very very] pretty girl. |
| F04 | using 'than' without comparative word<br>(ex) You will be swimmer [than] now.<br>[Than] you can get a tourist map. |
| F05 | redundant use of wh-word<br>(ex) I will order the same [whether what] you choose. |
| F06 | error sequence of 'pronoun+pronoun' or 'pronoun+noun'<br>(ex) [She hair] is very long.<br>[She always me] happy. |
| F07 | error sequence of 'possessive pronoun + verb'<br>(ex) What's [your eat] know?<br>[My is] her why I' don't know. |
| F08 | error sequence of 'wh-word + verb'<br>(ex) That's [why call] you.<br>I don't know she [why go] here. |
| F09 | confusing adjective and noun<br>(ex) Your good [healthy] is~<br>[foreign] asked a person~ |
| F10 | error sequence of 'determiner + verb'<br>(ex) He visited information center for [the get] a tourist map. |
| F11 | error sequence of 'be + modal verb'<br>(ex) I [am must] do it. |

In order to compare the performance between a full parsing approach and a FST approach, we make both modules to detect grammatical errors from a real corpus, and then evaluate their performances. The corpus we used in this evaluation consists of 20,234 sentences which Korean middle school students composed. Sentences consist of 6.07 words on average. Since test-takers' writing abilities vary a lot, various types of errors have been identified. We measure how many grammatical errors can be correctly detected by each module.

**Table 4** Evaluation and comparison of both approaches.

| | Full Parsing Approach | | FST approach | | the number of detected errors in common |
|---|---|---|---|---|---|
| error ID | total number of detected errors | accuracy | total number of detected errors | accuracy | |
| PF01 | 478 | 79.28% (379/478) | 420 | 90.0% (378/420) | 280 |
| PF02 | 78 | 100% | 78 | 100% | 78 |
| PF03 | 99 | 83.84% (83/99) | 79 | 100% | 79 |
| PF04 | 188 | 64.36% (121/188) | 176 | 77.84% (137/176) | 92 |
| PF05 | 203 | 98.52% (200/203) | 159 | 97.48% (155/159) | 152 |
| PF06 | 8 | 100% | 12 | 100% | 4 |
| PF07 | 1569 | 80.37% (1261/1569) | 1213 | 88.46% (1073/1213) | 879 |
| PF08 | 636 | 86.01% (547/636) | 536 | 88.06% (472/536) | 352 |
| PF09 | 1940 | 93.3% (1810/1940) | 1569 | 93.63% (1469/1569) | 1003 |
| PF10 | 257 | 70.43% (181/257) | 376 | 89.36% (336/376) | 101 |
| TOTAL | 5456 | 85.56% (4668/5456) | 4618 | 90.71% (4189/4618) | |

## 5.2 Evaluation Results

**Table 4** shows the comparison of the performance between the full parsing approach and the FST approach for the errors PF01~10. One of the apparent results is that the full parsing approach has more number of detected errors than the FST approach has. It is because the full parser can identify a grammar errors with a broader coverage while the FST can detect an error with smaller coverage in a sentence. The errors PF02 and PF06 have 100% accuracies as they can be easily identified regardless of which approach is used to detect errors. As you easily guess, the most frequently occurring error is a verb-form error (PF09), which has a smaller range of an input comparing to other grammatical errors. In the following, we list some of mis-identified errors in order to help you to understand what kind of problems can happen in this case.

Most of problems in case of the error PF07 (subj-verb agreement error) in the full parser are due to incompleteness of a syntactic parser and the vulnerability of a parser to input errors. Since the examples (4) and (5) have some syntactic errors, a syntactic parser cannot generate a full complete parse tree covering the whole sentence, instead, generates a few chunks. In case of (4), one of the chunks is [VP [NP will/noun] [VP go/verb]], and this chunk causes a subj-verb agreement error. The parser cannot make a phrase for the string 'She always give' in case of the example (5), so the subj-verb agreement error is not identified. Both problems do not appear in the FST approach since the FST does not care about building a whole parse tree. However, the FST detects a sub-verb agreement error for the example (6) since it cannot recognize a complex phrase ('to read books') as a subject and considers only a small range of the input.

<u>PF01: determiner-noun agreement error</u>

- the full parser incorrectly identifies the error:

(1) It is dangerous [that children] plays soccer in the street.

- the FST wrongly identifies the error:

(2) I saw [many fall] leaves.

(3) You tonight [four home] ok.

<u>PF07: subject verb agreement error</u>

- the full parser incorrectly identifies the error:

(4) I will go to the swimming next day.
  (parsing failure) →
  chunk [will/noun go/verb] causes the error

(5) She always give a happy to me.
  (parsing failure) → NO agreement error occurred

- the FST wrongly identifies the error:

(6) To read [books is] important in the weeks.

**Table 5** shows the accuracies of detecting errors mentioned in **Table 2**. Since the current version of the full parser is implemented to detect the restricted types of the errors P01 (adverb type error), P02 (auxiliary verb type error) and P03(noun number error), their recalls are low, but their accuracies are 100% in this evaluation. Comparing to other errors, the errors P08, P09, and P10 have low accuracy because they can be correctly identified only if the parser succeeded to analyze a whole sentence. The full parser detects the error P10 (relative pronoun missing error) a lot, but the half of them are evaluated the wrong detection, due to the incompleteness of the parser's performance. The following shows some problems of the full parser.

P04: verb-be complement agreement error
- the full parser incorrectly identifies the error:
(7) The Earth is the bigger than the Moon.
(parsing failure) →
chunk [The Earth is the bigger] causes the error

P10: relative pronoun missing error
- the full parser incorrectly identifies the error:
(8) I am tommorow go to swim.
→ [tommorrow/unknown that(missing) go/v] causes the error
(9) He speak English even Frence.
→ [English/n that(missing) even/v Frence/unknown-noun] causes the error
(10) Recetly the world have experienced many changes.
→ [the world/n that(missing) have experienced] (Due to the misspelling 'Recently', the whole sentence is analyzed as a NP(noun phrase) rather than a S(sentence).

**Table 6** shows the accuracies of detecting errors mentioned in Table 3. There are two main reasons why many F02 (sentence without main verb) errors are detected in the corpus. One is that there are a lot of incomplete sentences because test-takers' writing ability is so low that they do not know how to startand how to finish an English sentence. The examples of those sentences are "She true", "I born in city in Seoul.", "I yesterday", "This apple better than", etc. The other reason is due to an additional unnecessary punctuation within a sentence. The following sentence (11) has an unnecessary punctuation mark in the middle of the sentence, so the FST considers it as a separated two sentences, and the second sentence is evaluated without a main verb. The sentences (12) and (13) show the examples that the FST detects correctly the error F10 (the sequence of 'determiner + verb').

**Table 5** Accuracies of the full parsing approach.

| | Full Parsing Approach | |
|---|---|---|
| error ID | total number of detected errors | accuracy |
| P01 | 73 | 100% (73/73) |
| P02 | 11 | 100% (11/11) |
| P03 | 41 | 100% (41/41) |
| P04 | 97 | 61.86% (60/97) |
| P05 | 61 | 73.77% (45/61) |
| P06 | 35 | 97.14% (34/35) |
| P07 | 43 | 83.72% (36/43) |
| P08 | 27 | 59.25% (16/27) |
| P09 | 5 | 20% (1/5) |
| P10 | 296 | 47.97% (142/296) |
| P11 | 76 | 73.68% (56/76) |
| TOTAL | 765 | 67.32 (515/765) |

**Table 6** Accuracies of the FST approach.

| | FST Approach | |
|---|---|---|
| error ID | total number of detected errors | accuracy |
| F01 | 170 | 100% (170/170) |
| F02 | 550 | 83.64% (460/550) |
| F03 | 43 | 100% |
| F04 | 29 | 100% |
| F05 | 5 | 100% |
| F06 | 161 | 94.41% (152/161) |
| F07 | 23 | 100% |
| F08 | 8 | 100% |
| F09 | 108 | 92.59% (100/108) |
| F10 | 62 | 87.09% (54/62) |
| F11 | 14 | 100% |
| TOTAL | 1173 | 90.19% (1058/1173) |

**Table 7** Overall comparison between the full parsing approach and the FST approach.

| | Full parsing Approach | | FST approach | |
|---|---|---|---|---|
| | total number of errors | accuracy | total number of errors | accuracy |
| COMMON errors (PF01~PF10) | 5456 | 85.56% (4668/5456) | 4618 | 90.71% (4189/4618) |
| Full Parser's errors (P01~P11) | 765 | 67.32% (515/765) | | |
| FST's errors (F01~F11) | | | 1173 | 90.19% (1058/1173) |
| TOTAL | 6221 | 83.31% (5183/6221) | 5791 | 90.61% (5247/5791) |

F02: sentence without main verb
- In case the FST wrongly identifies the error:

(11) He can speak. Not only English but also French.

F10: error sequence of 'determiner + verb'
- In case the FST correctly identifies the error:

(12) young people are enjoy [the sing].
← A test-taker is confusing 'song' with 'sing'.

(13) Who call you [the tell] phone?
← A test-taker's intention is 'the telephone'.

**Table 7** summarizes the overall comparison between the two approaches. The full parsing approach can identify more errors than the FST approach can. However, the overall accuracy of the FST approach is higher than that of the full parsing approach. As long as we can decide when to use which approach automatically, combining the two approaches in the error detection can achieve the best performance.

## 4. Conclusion

In this paper, we have compared two approaches in detecting grammatical errors in an automated English writing scoring system. One is the full parser to analyze an input sentence as well as to detect grammatical errors from the sentence. That is to say, the full parser has grammatical rules as well as extra-grammatical rules. The other is the finite state transducer, which can detect only grammatical errors not decide if a whole input sentence is analyzable or not.

We conclude that a finite state approach is better than a full parsing approach in case that test-takers' writing abilities are at beginning level so their writing has multiple errors. However, when we want to detect an error with a long-distance range, a full parsing approach is necessary. The direction of the future study is how to combine the two approaches in the error detection.

## References

[1] Yongmei Shi and Lina Zhou. “Error detection using linguistic features.” Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing, pp. 41-48 2005.

[2] George E. Heidorn, Intelligent Writing Assistance, Handbook of Natural Language Processing. Robert Dale, Hermann Moisl and Harold Somers (Eds), CRC Press, 2000.

[3] Lisa N. Michaud, Kathleen F. McCoy, and Christopher A. Pennington. “An intelligent tutoring system for deaf learners of written english.” Proceedings of the Fourth International ACM Conference on Assistive Technologies, pp. 92-100, 2000.

[4] Emily M. Bender, Dan Flickinger, Stephan Oepen, Annemarie Walsh, and Timothy Baldwin. “Arboretum: Using a precision grammar for grammar checking in call.” Proceedings of InSTIL/ICALL Symposium on Computer Assisted Learning., pp. 83-86, 2004.

[5] Wael H. Gomaa and Aly A. Fahmy, “Short answer grading using string similarity and corpus-based similarity,” International Journal of Advanced Computer Science and Application, vol. 3, no. 11, pp. 115-121, 2012.

[6] Sylvana Sofkova Hashemi, Robin Cooper, and Robert Andersson, “Positive grammar checking: A finite state approach,” Computational Linguistics and Intelligent Text Processing Lecture Notes in Computer Science, vol. 2588, pp. 635-646, 2003.

[7] Kong Joo Lee, “Compositional rules of Korean auxiliary predicates for sentiment analysis,” Journal of the Korean Society of Marine Engineering, vol. 37, no. 3, pp. 291-299, 2013.

[8] Kong Joo Lee, Songwook Lee, and Jee Eun Kim, “A bidirectional Korean-Japanese statistical machine translation system by using MOSES,” Journal of the Korean Society of Marine Engineering, vol. 36, no. 5, pp. 683-693, 2012.

[9] Emmanuel Roche and Yves Schabes, Finite-State Language Processing, Cambridge, The MIT Press, 1997.

[10] Kong Joo Lee, Yong-Seok Choi, and Jee Eun Kim, “Building an automated English sentence evaluation system for students learning English as a second language,” Computer Speech & Language, vol. 25, no. 2, pp. 246-260, 2011.

[11] Guihua Sun, Xiaohua Liu, Gao Cong, Ming Zhou, Zhonguang Xiong, John Lee, and Chin-Yew Lin, “Detecting erroneous sentences using automatically mined sequential patterns,” Processings of the 45th Annual Meeting of the Association of Computational Linguistics, pp. 81–88, 2007.

[12] Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, and Helen Pinto, “Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth,” Proceedings of Proceedings of International Conference on Data Engineering, pp. 215-224, 2001.

[13] Michael Gamon, Claudia Leacock, Chris Brocket, W. B. Dolan, Jianfeng Gao, Dmitriy Belenko, and Alexander Klementiev, “Using statistical techniques and web search to correct ESL errors,” Journal of The Computer Assisted Language Instruction Consortium, vol. 26, no. 3, pp. 491-511, 2009.