EdIt: A Broad-Coverage Grammar Checker Using Pattern Grammar

Chung-Chi Huang Mei-Hua Chen

Institute of Information Systems and Applications, National Tsing Hua University, HsinChu, Taiwan, R.O.C. 300

Shih-Ting Huang Jason S. Chang

Department of Computer Science, National Tsing Hua University, HsinChu, Taiwan, R.O.C. 300

{u901571,chen.meihua,koromiko1104,Jason.jschang}@gmail.com

Abstract

We introduce a new method for learning to detect grammatical errors in learner's writing and provide suggestions. The method involves parsing a reference corpus and inferring grammar patterns in the form of a sequence of content words, function words, and parts-of-speech (e.g., "*play ~ role in* Ving" and "*look forward to* Ving"). At runtime, the given passage submitted by the learner is matched using an extended Levenshtein algorithm against the set of pattern rules in order to detect errors and provide suggestions. We present a prototype implementation of the proposed method, EdIt, that can handle a broad range of errors. Promising results are illustrated with three common types of errors in non-native writing.

1 Introduction

Recently, an increasing number of research has targeted language learners' need in editorial assistance including detecting and correcting grammar and usage errors in texts written in a second language. For example, Microsoft Research has developed the ESL Assistant, which provides such a service to ESL and EFL learners.

Much of the research in this area depends on hand-crafted rules and focuses on certain error types. Very little research provides a general framework for detecting and correcting all types of errors. However, in the sentences of ESL writing, there may be more than one errors and one error may affect the performance of handling other errors. Erroneous sentences could be more efficiently identified and corrected if a grammar checker handles all errors at once, using a set of pattern rules that reflect the predominant usage of the English language.

Consider the sentences, "*He play an important roles to close this deals.*" and "*He looks forward to hear you.*" The first sentence contains inaccurate word forms (i.e., *play, roles,* and *deals*), and rare usage (i.e., "*role to close*"), while the second sentence use the incorrect verb form of "*hear*". Good responses to these writing errors might be (a) Use "*played*" instead of "play." (b) Use "*role*" instead of "*roles*", (c) Use "*in closing*" instead of "*to close*" (d) Use "*to hearing*" instead of "*to hear*", and (e) insert "*from*" between "*hear*" and "*you*." These suggestions can be offered by learning the patterns rules related to "**play ~ role**" and "**look forward**" based on analysis of ngrams and collocations in a very large-scale reference corpus. With corpus statistics, we could learn the needed phraseological tendency in the form of pattern rules such as "**play ~ role in V-ing**) and "**look forward to V-ing**." The use of such pattern rules is in line with the recent theory of Pattern Grammar put forward by Hunston and Francis (2000).

We present a system, EdIt, that automatically learns to provide suggestions for rare/wrong usages in non-native writing. Example EdIt responses to a

text are shown in Figure 1. EdIt has retrieved the related pattern grammar of some ngram and collocation sequences given the input (e.g., "*play ~ role in* V-ing[1]", and "*look forward to* V-ing"). EdIt learns these patterns during pattern extraction process by syntactically analyzing a collection of well-formed, published texts.

At run-time, EdIt first processes the input passages in the article (e.g., "*He play an important roles to close* ") submitted by the L2 learner. And EdIt tag the passage with part of speech information, and compares the tagged sentence against the pattern rules anchored at certain collocations (e.g., "*play ~ role*" and "*look forward*"). Finally, EdIt finds the minimum-edit-cost patterns matching the passages using an extended Levenshtein's algorithm (Levenshtein, 1966). The system then highlights the edits and displays the pattern rules as suggestions for correction. In our prototype, EdIt returns the preferred word form and preposition usages to the user directly (see Figure 1); alternatively, the actual surface words (e.g., "closing" and "deal") could be provided.

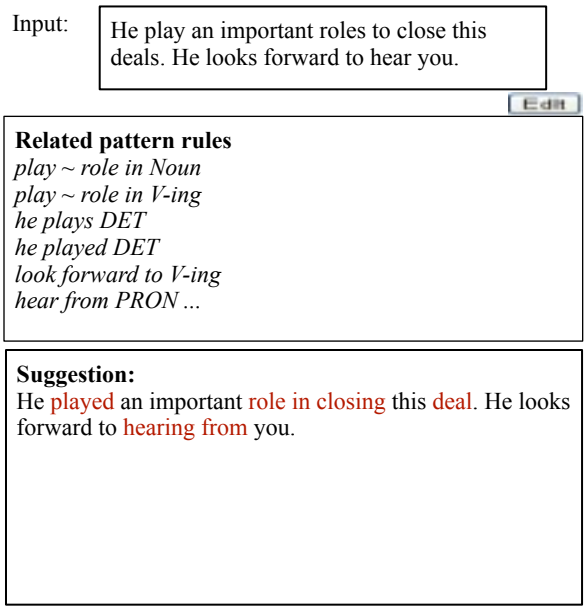

Figure 1. Example responses to the non-native writing.

2 Related Work

Grammar checking has been an area of active research. Many methods, rule-oriented or data-driven, have been proposed to tackle the problem of detecting and correcting incorrect grammatical and usage errors in learner texts. It is at times no easy to distinguish these errors. But Fraser and Hodson (1978) shows the distinction between these two kinds of errors.

For some specific error types (e.g., article and preposition error), a number of interesting rule-based systems have been proposed. For example, Uria et al. (2009) and Lee et al. (2009) leverage heuristic rules for detecting Basque determiner and Korean particle errors, respectively. Gamon et al. (2009) bases some of the modules in ESL Assistant on rules derived from manually inspecting learner data. Our pattern rules, however, are automatically derived from readily available well-formed data, but nevertheless very helpful for correcting errors in non-native writing.

More recently, statistical approaches to developing grammar checkers have prevailed. Among unsupervised checkers, Chodorow and Leacock (2000) exploits negative evidence from edited textual corpora achieving high precision but low recall, while Tsao and Wible (2009) uses general corpus only. Additionally, Hermet et al. (2008) and Gamon and Leacock (2010) both use Web as a corpus to detect errors in non-native writing. On the other hand, supervised models, typically treating error detection/correction as a classification problem, may train on well-formed texts as in the methods by De Felice and Pulman (2008) and Tetreault et al. (2010), or with additional learner texts as in the method proposed by Brockett et al. (2006). Sun et al. (2007) describes a method for constructing a supervised detection system trained on raw well-formed and learner texts without error annotation.

Recent work has been done on incorporating word class information into grammar checkers. For example, Chodorow and Leacock (2000) exploit bigrams and trigrams of function words and part-of-speech (PoS) tags, while Sun et al. (2007) use labeled sequential patterns of function, time expression, and part-of-speech tags. In an approach similar to our work, Tsao and Wible (2009) use a combined ngrams of words forms, lemmas, and part-of-speech tags for research into constructional phenomena. The main differences are that we anchored each pattern rule in lexical collocation so as to avoid deriving rules that is may have two

[1] In the pattern rules, we translate the part-of-speech tag to labels that are commonly used in learner dictionaries. For instance, we use *V-ing* for the tag VBG denoting the progressive verb form, and *Pron* and *Pron$* denotes a pronoun and a possessive pronoun respectively.

consecutive part-of-speech tags (e.g, "*V Pron$ socks off"*). The pattern rules we have derived are more specific and can be effectively used in detecting and correcting errors.

In contrast to the previous research, we introduce a broad-coverage grammar checker that accommodates edits such as substitution, insertion and deletion, as well as replacing word forms or prepositions using pattern rules automatically derived from very large-scale corpora of well-formed texts.

3 The EdIt System

Using supervised training on a learner corpus is not very feasible due to the limited availability of large-scale annotated non-native writing. Existing systems trained on learner data tend to offer high precision but low recall. Broad coverage grammar checkers may be developed using readily available large-scale corpora. To detect and correct errors in non-native writing, a promising approach is to automatically extract lexico-syntactical pattern rules that are expected to distinguish correct and in correct sentences.

3.1 Problem Statement

We focus on correcting grammatical and usage errors by exploiting pattern rules of specific collocation (elastic or rigid such as "*play ~ rule*" or "*look forward*"). For simplification, we assume that there is no spelling errors. EdIt provides suggestions to common writing errors[2] of the following correlated with essay scores[3].

(1) wrong word form
  (A) singular determiner preceding plural noun
  (B) wrong verb form: concerning modal verbs (e.g., "would said"), subject-verb agreement, auxiliary (e.g., "should have tell the truth"), gerund and infinitive usage (e.g., "look forward to see you" and "in an attempt to helping you")

(2) wrong preposition (or infinitive-to)
  (A) wrong preposition (e.g., "to depends of it")
  (B) wrong preposition and verb form (e.g., "to play an important role to close this deal")

(3) transitivity errors
  (A) transitive verb (e.g., "to discuss about the matter" and "to affect to his decision")
  (B) intransitive verb (e.g., "to listens the music")

The system is designed to find pattern rules related to the errors and return suggestionst. We now formally state the problem that we are addressing.

Problem Statement: We are given a reference corpus *C* and a non-native passage *T*. Our goal is to detect grammatical and usage errors in *T* and provide suggestions for correction. For this, we *extract* a set of *pattern rules*, $u_1,\ldots, u_m$ from *C* such that the rules reflect the predominant usage and are likely to distinguish most errors in non-native writing.

In the rest of this section, we describe our solution to this problem. First, we define a strategy for identifying predominant phraseology of frequent ngrams and collocations in Section 3.2. Afer that, we show how EdIt proposes grammar correctionsedits to non-native writing at run-time in Section 3.3.

3.2 Deriving Pattern Rules

We attempt to derive patterns (e.g., "*play ~ role in V-ing*") from *C* expected to represent the immediate context of collocations (e.g., "*play ~ role*" or "*look forward*"). Our derivation process consists of the following four-stage:

Stage 1. Lemmatizing, POS Tagging and Phrase chunking. In the first stage, we lemmatize and tag sentences in *C*. Lemmatization and POS tagging both help to produce more general pattern rules from ngrams or collocations. The based phrases are used to extract collocations.

Stage 2. Ngrams and Collocations. In the second stage of the training process, we calculate ngrams and collocations in *C*, and pass the frequent ngrams and collocations to Stage 4.

We employ a number of steps to acquire statistically significant collocations--determining the pair of head words in adjacent base phrases, calculating their pair-wise mutual information values, and filtering out candidates with low MI values.

Stage 3. onstructing Inverted Files. In the third stage in the training procedure, we build up inverted files for the lemmas in *C* for quick access in Stage 4. For each word lemma we store surface words, POS tags, pointers to sentences with base phrases marked.

[2] See (Nicholls, 1999) for common errors.

[3] See (Leacock and Chodorow, 2003) and (Burstein et al., 2004) for correlation.

```
procedure GrammarChecking(T,PatternGrammarBank)
(1) Suggestions=""//candidate suggestions
(2) sentences=sentenceSplitting(T)
    for each sentence in sentences
(3)  userProposedUsages=extractUsage(sentence)
     for each userUsage in userProposedUsages
(4)   patGram=findPatternGrammar(userUsage.lexemes,
                          PatternGrammarBank)
(5)   minEditedCost=SystemMax; minEditedSug=""
      for each pattern in patGram
(6)     cost=extendedLevenshtein(userUsage,pattern)
        if cost<minEditedCost
(7)       minEditedCost=cost; minEditedSug=pattern
      if minEditedCost>0
(8)     append (userUsage,minEditedSug) to Suggestions
(9) Return Suggestions
```

Figure 2. Grammar suggestion/correction at run-time

```
procedure extendedLevenshtein(userUsage,pattern)
(1) allocate and initialize costArray
    for i in range(len(userUsage))
      for j in range(len(pattern))
        if equal(userUsage[i],pattern[j]) //substitution
(2a)     substiCost=costArray[i-1,j-1]+0
        elseif sameWordGroup(userUsage[i],pattern[j])
(2b)     substiCost=costArray[i-1,j-1]+0.5
(2c)    else substiCost=costArray[i-1,j-1]+1
        if equal(userUsage[i+1],pattern[j+1]) //deletion
(3a)     delCost=costArray[i-1,j]+smallCost
(3b)    else delCost=costArray[i-1,j]+1
        if equal(userUsage[i+1],pattern[j+1]) //insertion
(4a)     insCost=costArray[i,j-1]+smallCost
(4b)    else insCost=costArray[i,j-1]+1
(5)     costArray[i,j]=min(substiCost,delCost,insCost)
(6) Return costArray[len(userUsage),len(pattern)]
```

Figure 3. Algorithm for identifying errors

Stage 4. Deriving pattern rules. In the fourth and final stage, we use the method described in a previous work (Chen et al., 2011) and use the inverted files to find all sentences containing a give word and collocation. Words surrounding a collocation are identified and generalized based on their corresponding POS tags. These sentences are then transformed into a set of n-gram of words and POS tags, which are subsequently counted and ranked to produce pattern rules with high frequencies.

3.3 Run-Time Error Correction

Once the patterns rules are derived from a corpus of well-formed texts, EdIt utilizes them to check grammaticality and provide suggestions for a given text via the procedure in Figure 2.

In Step (1) of the procedure, we initiate a set *Suggestions* to collect grammar suggestions to the user text *T* according to the bank of pattern grammar *PatternGrammarBank*. Since EdIt system focuses on grammar checking at sentence level, *T* is heuristically split (Step (2)).

For each *sentence*, we extract ngram and POS tag sequences *userUsage* in *T*. For the example of "He play an important roles. He looks forword to hear you", we extract ngram such as *he V DET, play an JJ NNS, play ~ roles to V, this NNS, look forward to VB,* and *hear Pron.*

For each *userUsage*, we first access the pattern rules related to the word and collocation within (e.g., play-role patterns for "*play ~ role to close*") Step (4). And then we compare *userUsage* against these rules (from Step (5) to (7)). We use the extended Levenshtein's algorithm shown in Figure 3 to compare *userUsage* and pattern rules.

If only partial matches are found for *userUsage*, that could mean we have found a potential errors. We use *minEditedCost* and *minEditedSug* to contrain the patterns rules found for error suggestions (Step (5)). In the following, we describe how to find minimal-distance edits.

In Step (1) of the algorithm in Figure 3 we allocate and initialize *costArray* to gather the dynamic programming based cost to transform *userUsage* into a specific contextual rule *pattern*. Afterwards, the algorithm defines the cost of performing substitution (Step (2)), deletion (Step (3)) and insertion (Step (4)) at i-indexed *userUsage* and j-indexed *pattern*. If the entries *userUsage*[*i*] and *pattern*[*j*] are equal literally (e.g., "VB" and "VB") or grammatically (e.g., "DT" and "Pron$"), no edit is needed, hence, no cost (Step (2a)). On the other hand, since learners tend to select wrong word form and preposition, we set a lower cost for substitution among different word forms of the same lemma or lemmas with the same POS tag (e.g., *replacing V with V-ing* or *replacing to with in*". In addition to the conventional deletion and insertion (Step (3b) and (4b) respectively), we look ahead to the elements *userUsage*[*i*+1] and *pattern*[*j*+1] considering the fact that "with or without preposition" and "transitive or intransitive verb" often puzzles EFL learners (Step (3a) and (4a)). Only a small edit cost is counted if the next elements in *userUsage* and *Pattern* are "equal". In Step (6) the extended Levenshtein's algorithm returns the minimum edit cost of revising *userUsage* using *pattern*.

Once we obtain the costs to transform the *userUsage* into a similar, frequent pattern rules, we propose the minimum-cost rules as suggestions for

correction (e.g., "*play ~ role in V-ing*" for revising "*play ~ role to V*") (Step (8) in Figure 2), if its minimum edit cost is greater than zero. Otherwise, the usage is considered valid. Finally, the *Suggestions* accumulated for *T* are returned to users (Step (9)). Example input and editorial suggestions returned to the user are shown in Figure 1. Note that pattern rules involved flexible collocations are designed to take care of long distance dependencies that might be always possible to cover with limited ngram (for *n* less than 6). In addition, the long patter rules can be useful even when it is not clear whether there is an error when looking at a very narrow context. For example, "hear" can be either be transitive or intransitive depending on context. In the context of "look forward to" and person noun object, it is should be intransitive and require the preposition "from" as suggested in the results provided by EdIt (see Figure 1).

In existing grammar checkers, there are typically many modules examining different types of errors and different module may have different priority and conflict with one another. Let us note that this general framework for error detection and correction is an original contribution of our work. In addition, we incorporate probabilities conditioned on word positions in order to weigh edit costs. For example, the conditional probability of *V* to immediately follow "look forward to" is virtually 0, while the probability of *V-ing* to do so is approximates 0.3. Those probabilistic values are used to weigh different edits.

4 Experimental Results

In this section, we first present the experimental setting in EdIt (Section 4.1). Since our goal is to provide to learners a means to efficient broad-coverage grammar checking, EdIt is web-based and the acquisition of the pattern grammar in use is offline. Then, we illustrate three common types of errors, scores correlated, EdIt[4] capable of handling.

4.1 Experimental Setting

We used British National Corpus (BNC) as our underlying general corpus *C*. It is a 100 million British English word collection from a wide range of sources. We exploited GENIA tagger to obtain the lemmas, PoS tags and shallow parsing results of *C*'s sentences, which were all used in constructing inverted files and used as examples for GRASP to infer lexicalized pattern grammar.

Inspired by (Chen et al., 2011) indicating EFL learners tend to choose incorrect prepositions and following word forms following a VN collocation, and (Gamon and Leacock, 2010) showing fixed-length and fixed-window lexical items are the best evidence for correction, we equipped EdIt with pattern grammar rules consisting of fixed-length (from one- to five-gram) lexical sequences or VN collocations and their fixed-window usages (e.g., "IN(*in*) VBG" after "play ~ role", for window 2).

4.2 Results

We examined three types of errors and the mixture of them for our correction system (see Table 1). In this table, results of ESL Assistant are shown for comparison, and grammatical suggestions are underscored. As suggested, lexical and PoS information in learner texts is useful for a grammar checker, pattern grammar EdIt uses is easily accessible and effective in both grammaticality and usage check, and a weighted extension to Levenshtein's algorithm in EdIt accommodates substitution, deletion and insertion edits to learners' frequent mistakes in writing.

5 Future Work and Summary

Many avenues exist for future research and improvement. For example, we could augment pattern grammar with lexemes' PoS information in that the contexts of a word of different PoS tags vary. Take *discuss* for instance. The present tense verb *discuss* is often followed by determiners and nouns while the passive is by the preposition *in* as in "… is discussed in Chapter one." Additionally, an interesting direction to explore is enriching pattern grammar with semantic role labels (Chen et al., 2011) for simple semantic check.

In summary, we have introduced a method for correcting errors in learner text based on its lexical and PoS evidence. We have implemented the method and shown that the pattern grammar and extended Levenshtein algorithm in this method are promising in grammar checking. Concerning EdIt's broad coverage over different error types, simplicity in design, and short response time, we plan to evaluate it more fully: with or without conditional probability using majority voting or not.

[4] At http://140.114.214.80/theSite/EdIt_demo2/

Erroneous sentence	EdIt suggestion	ESL Assistant suggestion
Incorrect word form		
… a sunny days …	a sunny N	a sunny day
every days, I …	every N	every day
I would said to …	would V	would say
he play a …	he V-ed	none
… should have tell the truth	should have V-en	should have to tell
… look forward to see you	look forward to V-ing	none
… in an attempt to seeing you	an attempt to V	none
… be able to solved this problem	able to V	none
Incorrect preposition		
he plays an important role to close …	play ~ role in	none
he has a vital effect at her.	have ~ effect on	effect on her
it has an effect on reducing …	have ~ effect of V-ing	none
… depend of the scholarship	depend on	depend on
Confusion between intransitive and transitive verb		
he listens the music.	missing "to" after "listens"	missing "to" after "listens"
it affects to his decision.	unnecessary "to"	unnecessary "to"
I understand about the situation.	unnecessary "about"	unnecessary "about"
we would like to discuss about this matter.	unnecessary "about"	unnecessary "about"
Mixed		
she play an important roles to close this deals.	she V-ed; an Adj N; play ~ role in V-ing; this N	play an important role; close this deal
I look forward to hear you.	look forward to V-ing; missing "from" after "hear"	none

Table 1. Three common score-related error types and their examples with suggestions from EdIt and ESL Assistant.

References

C. Brockett, W. Dolan, and M. Gamon. 2006. Correcting ESL errors using phrasal SMT techniques. In *Proceedings of the ACL*.

J. Burstein, M. Chodorow, and C. Leacock. 2004. Automated essay evaluation: the *criterion online writing service*. *AI Magazine*, 25(3):27-36.

M. H. Chen, C. C. Huang, S. T. Huang, H. C. Liou, and J. S. Chang. 2011. A cross-lingual pattern retrieval framework. In *Proceedings of the CICLing*.

M. Chodorow and C. Leacock. 2000. An unsupervised method for detecting grammatical errors. In *Proceedings of the NAACL*, pages 140-147.

R. De Felice and S. Pulman. 2008. A classifer-based approach to preposition and determiner error correction in L2 English. In *COLING*.

I. S. Fraser and L. M. Hodson. 1978. Twenty-one kicks at the grammar horse. *English Journal*.

M. Gamon, C. Leacock, C. Brockett, W. B. Bolan, J. F. Gao, D. Belenko, and A. Klementiev. Using statistical techniques and web search to correct ESL errors. *CALICO*, 26(3): 491-511.

M. Gamon and C. Leacock. 2010. Search right and thou shalt find … using web queries for learner error detection. In *Proceedings of the NAACL*.

M. Hermet, A. Desilets, S. Szpakowicz. 2008. Using the web as a linguistic resource to automatically correct lexico-syntatic errors. In *LREC*, pages 874-878.

S. Hunston and G. Francis. 2000. Pattern grammar: a corpus-driven approach to the lexical grammar of English.

C. M. Lee, S. J. Eom, and M. Dickinson. 2009. Toward analyzing Korean learner particles. In *CALICO*.

V. I. Levenshtein. 1966. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10:707-710.

C. Leacock and M. Chodorow. 2003. Automated grammatical error detection.

D. Nicholls. 1999. The Cambridge Learner Corpus – error coding and analysis for writing dictionaries and other books for English Learners.

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

J. Tetreault, J. Foster, and M. Chodorow. 2010. Using parse features for prepositions selection and error detection. In *Proceedings of the ACL*, pages 353-358.

N. L. Tsao and D. Wible. 2009. A method for unsupervised broad-coverage lexical error detection and correction. In *NAACL Workshop*, pages 51-54.