Identifying Correction Rules for Auto Editing

An-Ta Huang
Department of Computer Science and Information Engineering
National Taiwan University
r97922137@ntu.edu.tw

Tsung-Ting Kuo
Department of Computer Science and Information Engineering
National Taiwan University
d97944007@csie.ntu.edu.tw

Ying-Chun Lai
School of Applied Foreign Languages
Chung-Shan Medical University
yingchun@csmu.edu.tw

Shou-de Lin
Department of Computer Science and Information Engineering
National Taiwan University
sdlin@csie.ntu.edu.tw

Abstract

This paper describes a framework to extract the effective correction rules from the sentence-aligned corpus and show a practical application: auto-editing using the found rules. The framework exploits the methodology of finding Levenshtein distance between sentences to identify the key parts of the rules and then use the editing corpus to filter, condense and refine the rules. We produce the rule candidates of such form, A => B, where A stands for the erroneous pattern and B is the correct pattern.

Our framework is language independent, therefore can be applied to other languages easily. The evaluation of the discovered rules reveals that 67.2% of the top 1500 ranked rules are annotated as correct or mostly correct by experts. Based on the rules, we create an online auto-editing system for demo on http://mslab.csie.ntu.edu.tw/~kw/new_demo.html.

Keywords: edit distance, erroneous pattern, correction rules, auto editing

Proceedings of the 22nd Conference on Computational Linguistics and Speech Processing (ROCLING 2010), Pages 251-265, Puli, Nantou,Taiwan, September 2010.

1 Introduction

Nowadays people write blogs, diaries, and reports not only in their native language but sometimes using a language they are not familiar with. During the process of writing, people sometimes make spelling mistakes, grammatical errors, or inappropriate lexical usage in writing, in particular if not using their native language. Therefore, it has become an important and practical research issue for NLP researchers on how to provide editorial assistance automatically and effectively. Especially for the second language learners, instant response to their writing indicating which part might be incorrect and then providing the auto-editing candidates would improve not only the writing itself but also the understanding of the language.

Editing is an inevitable part in learning language and can be classified into human editing and machine editing. Human editing is inefficient when the number of article is increasing and also inconvenient for people who want to edit their daily documents. Besides, human editing involves subjective opinions which are different to machine editing which depends on the objective experiment results.

Despite the growing demand of editorial assistance tools, the existing ones still have much room for improvement. Grammar checker provided by Microsoft Word has its known deficiency as being language dependent and covering only a small portion of errors without explicitly revealing the correction mechanism behind it. This paper tries to demonstrate an auto-editing system based on the correction rules mined from online editing websites.

In this paper, we focus on two research goals. First, we want to design a strategy that identifies effective rules automatically and efficiently from editing databases. Second, we want to design an auto-editing system based on the discovered rules.

Our method is language independent, therefore can be easily applied to another language. Our evaluation reveals that among the top 1500 rules the system found, 67.2% of them are regarded as correct or mostly correct.

The remainder of the paper is organized as follows: Section 2 describe the related work on detecting erroneous patterns. Section 3 lays out our methodology. Section 4 describes experiment and our demo system. Section 5 concludes our study.

2 Related Works

Previous approaches can be classified into two categories. The first category detects erroneous patterns based on rules and the second category makes use of statistical techniques.

2.1 Knowledge-Based Method

Some methods detecting erroneous patterns based on the manually created rules are proven to be effective in detecting grammar errors [1]. Lisa N. Michaud develops a system including error identification model and response generation model using

knowledge bases which cover general information about analyzing grammar structure and specific information of a user's learning history [2]. Besides, Emily M. Bender present a tutorial system based on computational grammar augmented with mal-rules for analysis, error diagnosis, and semantics-centered generation of correct forms [3]. However, the manually designed rules consume labor and time and require language experts to do that, which limits the generalization of such method and cannot be easily applied to different languages.

2.2 Statistical Techniques

As discussed in Section 2.1, rule-based methods have some apparent shortcomings. Rather than asking experts to annotate corpus, some papers propose statistical models to identify erroneous patterns. An unsupervised method to detect grammatical errors by inferring negative evidences reaches 80% precision and 20% recall but their system is only effective in recognizing certain grammatical errors and detect only about one-fifth as many errors as a human judge does [4]. Some other papers focus on detecting particular errors, such as preposition errors [5], disagreement on the quantifier and the noun [6]. Others treat the detection of erroneous sentences as a binary classification problem and propose a new feature called "Labeled Sequential Patterns (LSP)" which is compared to the other four features including two scores produced by toolkit, lexical collocation [11], and function word density. The average accuracy of LSP is 79.63% and beats the other four features. Furthermore, the existence of time word and function word in a sentence is proven to be important. While in this way, one can only know whether a sentence is correct and would not have a clue about how to correct errors [7]. Finally, some researchers treat detecting erroneous patterns as a statistical machine translation problem. They believe the erroneous sentences and the correct sentences are two different languages. However, error correction is intrinsically different from translation and there is no apparent evidence till now whether the existing machine translation techniques are suitable for such purpose [8][9].

Our work is different from the previous work in two major respects. First, we treat error detection as a pattern mining problem to extract effective rules from editing corpus. Second, we do not use any context, syntactic, or grammatical information in this paper.

3 Methodology

3.1 Overview

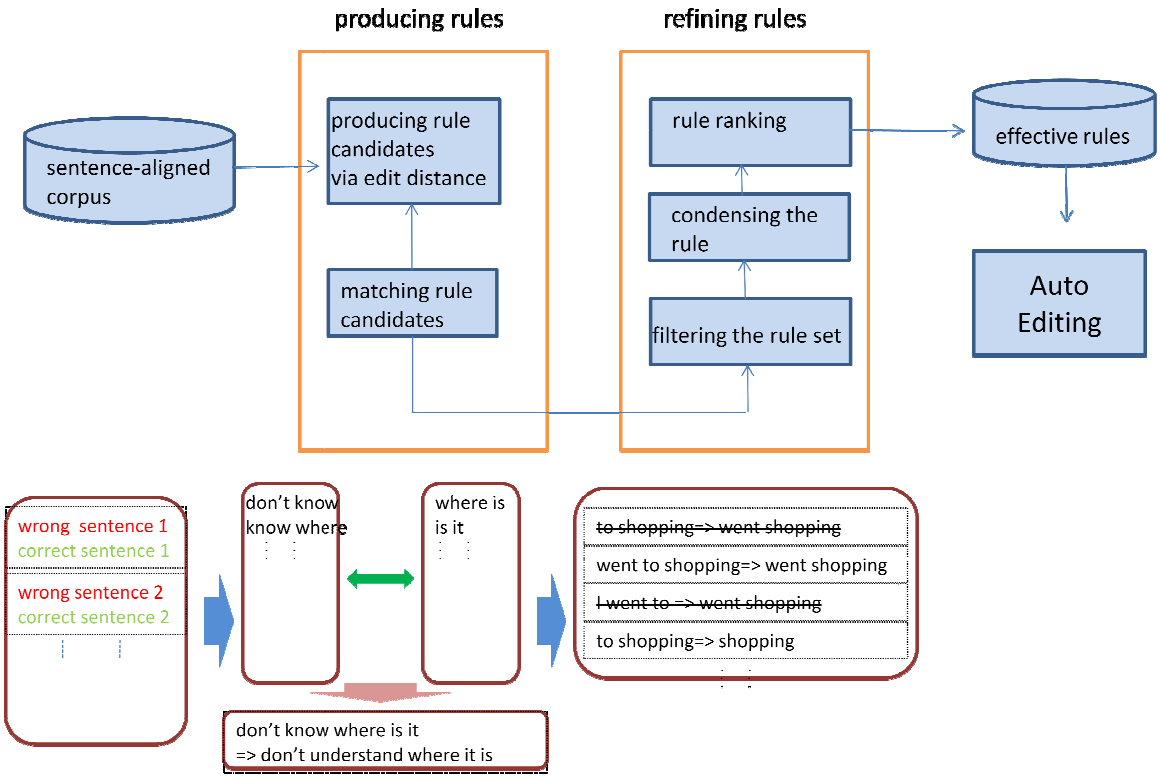

Figure 1. System Overview

Our framework consists of two parts. First it produces the rules and next it focuses on refining the rules.

3.2 Corpus Description

We retrieve 310967 parallel pairs of sentences, one erroneous sentence and one correct sentence from an online-editing website Lang-8 (http://lang-8.com/). The website allows people to write diaries in their second language and the diaries (which may contain some mistakes) would be edited by some volunteer members who are native in the corresponding language. Besides, the edited part in an article is restricted a single sentence (not cross sentences). Consequently, we can retrieve the sentence-aligned data easily from the website.

In the following chapters, we use W_i to represent the erroneous sentence of the i-th pair of sentence in the corpus and C_i represents the corresponding correct sentence to W_i. Besides, S_+ is defined as a collection of all correct sentences in the corpus, while S- is defined as a collection of all erroneous sentences.

3.3 Producing Rules

Before describing our method, we would like to provide some formal definitions for erroneous and correct patterns, rules, applying rules, and frequency of patterns:

Definition of (erroneous and correct) patterns: A pattern is a series of consecutive words (or characters) which belong to a subsequence of a sentence. An erroneous pattern represents such sequence that is believed to be wrong, and a correct pattern is the one that is believed to be correct.

Definition of rule: A rule K can be written as K_L => K_R. The left-hand side of the arrow, K_L, is the erroneous pattern and the right-hand side of the arrow, K_R, is correct pattern which K_L should be transformed to.

Definition of applying a rule to a sentence: Given a rule K : K_L => K_R, and a sentence T, if K_L exist in the T, we replace the K_L in T to K_R. Such process is considered as applying rule K to sentence T.

Definition of $fre_{S+}(K_L)$: the occurrence frequency of a pattern K_L in S_+

To discover a rule A→ B from the editing corpus, we need to first identify the plausible left and right hand side of the rule. This is by no means a trivial task, and the fact that there could be multiple choices of such rule can only make such tasks even harder. One intuitive method is to compare the word set existing in W_i and C_i and create the patterns using the difference among them. However, such the intuitive method suffers certain deficiency such as provided by the following examples.

Erroneous : "Open the light"
Correct : "Turn the light on "

Erroneous : "I with him had dinner"
Correct : "I had dinner with him"

The difference set between the first pairs are {open, turn, on} while that of the second pairs is an empty set {} since the order is not considered. It is not clear how such difference set can lead to both erroneous and correct patterns.

The approach we proposed is to exploit the procedure of calculating the word-level

Levenshtein distance which is often called editing distance [10]. The Levenshtein distance is defined as the minimum number of edits needed to transform one string into the other, with the allowable edit operations being insertion, deletion, or substitution of a single character. The *insert* operation inserts a word X in the erroneous sentence, which implies there is a word X in the correct sentence that has potential to be involved in the correct pattern K_R for a rule $K_L \rightarrow K_R$. Similarly, the *delete* operation removes one word Y from the erroneous sentence to become the correct one, and this word Y is likely to be involved in the erroneous pattern K_L. Finally, when a substitute operation is performed, the word to be replaced should appear in K_L while the replacing word shall be involved in K_R. Similarly, the edit distance between two sentences can be defined as the minimum number of allowable operations required to transform from one of them into the other, given each unit of transformation is based on words instead of characters.

Here we argue that the words operated through the editing-distance process from an erroneous to a correct sentence have higher chance to involve in the patterns for rules. For example, if we apply an editing distance approach to the following sentence pairs, multiple outputs such can be acquired such as the ones in table 1 and table 2:

Erroneous : "I still don't know where is it in the movie"

Correct : "I still don't understand where it is in the movie"

Based on the two editing-distance results shown in table 1 and 2, it is possible to obtain that the four words {it, is, know, understand} are plausible words to appear in the rule $K_L \rightarrow K_R$.

Table 1. one of the editing results for edit distance

operation	position	Involved word
Insert	6	it
delete	8	it
substitute	4	know→understand

Table 2. another editing result for edit distance

operation	position	Involved word
insert	8	Is
delete	6	Is
substitute	4	know→understand

For each pair of W_i and C_i, we can collect all the involving words after producing the Levenshtein distance. Figure 2. shows the pseudo code. We exploit a dynamic programming approach to improve its efficienscy.

Algorithm 1 edit distance

Input: Vector $V = \{(\text{erroneous sentence}, \text{correct sentence})\}$
Output: Integer $I = \text{edit distance}$

```
array E[] ⇐ erroneous sentence splited by whitespace
array C[] ⇐ correct sentence splited by whitespace
n ⇐ word counts in erroneous sentences
m ⇐ word counts in correct sentences
D ⇐ [n × m] array
if n = 0 then
    MD ⇐ m
end if
if m = 0 then
    MD ⇐ n
end if
for i = 1 to n do
    D[i][0] ⇐ i
end for
for i = 1 to m do
    D[0][i] ⇐ i
end for
for i = 1 to n do
    for j = 1 to m do
        if E[i] = C[j] then
            COST ⇐ 0
        else
            COST ⇐ 1
        end if
        D[i][j] ⇐ min(D[i-1][j]+1, D[i][j-1]+1, D[i-1][j-1]+COST)
    end for
end for
return D[n][m]
```

Algorithm 2 Rule Candidate Producing.

Input:
Set S - pairs of sentence set
Set V - word set produced by edit distance
Output: Set V - rule candidate set

```
begin
    foreach sl ∈ S do
        foreach consecutive word w ∈ sl do
            if w contains the different part then
                V.add(w)
            end if
        end for
    end for
end
return V
```

Algorithm 3 Rule Matching.

Input: Set S - rule candidate set
Output: Set R - matched rule set

```
N ⇐ threshold
begin
    foreach v1 ∈ S do
        foreach v2 ∈ S do
            if edit distance(v1, v2) ≤ N then
                String rule ⇐ form a rule, v1 => v2
                V.add(rule)
            end if
        end for
    end for
end
return R
```

Figure 2. pseudo code of producing rules

After applying the modified Levenshtein distance algorithm, it is possible to obtain a set of involving words R_i as follows.

$$R_i = \{\text{is}, \text{it}, \text{understand}, \text{know}\}$$

However, to form a reasonable pattern, the words in set Ri is not sufficient. They should be combined with other terms. Ideally K_L and K_R must consist of some words from R_i and some from the rest of the sentence. Therefore for each pair of W_i and C_i in the corpus, we retrieve consecutive word patterns in which at least one word is from R_i. For example, based on R_i, the followings are rule candidates.

Table 3. pattern candidates for forming a rule

Candidates for K_L (word length ≤ 4)	Candidates for K_R (word length ≤ 4)
don't know	*don't understand*
know where	*understand where*
where is	*where it*
is it	*it is*
it in	*is in*
still don't know	*still don't understand*
don't know where	*don't understand where*
know where is	*understand where it*
where is it	*where it is*
is it in	*it is in*
it in the	*is in the*
I still don't know	*I still don't understand*
still don't know where	*still don't understand where*
don't know where is	*don't understand where it*
know where is it	*understand where it is*
where is it in	*where it is in*
is it in the	*it is in the*
it in the movie	*is in the movie*

Next, we match each plausible candidate for K_L to each candidate for K_R to form a plausible rule. For each plausible rule, we then check its feasibility by applying it on W_i to see if the correct sentence C_i can be produced. The infeasible rules will be ignored.

Definition of feasible rule: Given a rule K : K_L => K_R, if there exists at least one erroneous sentence in corpus can be corrected using K , then K is considered as a feasible rule.

3.4 Refining Rules

So far, we have produced several rules, some makes more sense and some might not. In this section, we describe how to assess the quality of the rules.

Table 4. observation on the frequency

	pattern	fre_{S+}
erroneous	went to shopping	10
correct	went shopping	205

	pattern	fre_{S+}
Erroneous	am so exciting	0
Correct	am so excited	71

We believe the erroneous patterns K_L should not occur in the correct sentences too frequently (otherwise it shall have been replaced by the correct one K_R), therefore consider fre_{S+} as a property metric to evaluate the quality of a rule. According to the real experiment shown in Table 4, the frequency of the erroneous patterns seems to occur less frequently in the correct corpus, fre_{S+}, comparing to the correct ones.

Next, we condense the rules according to their fre_{S+}. The condensed rule is shorter than the original one and is supposed to be more general (i.e. can cover more sentences). For example, in the following sentences, the condensed rule is more general and reasonable since the subject 'I' has nothing to do with the erroneous pattern.

Erroneous : "I went to shopping and had dinner with my friend yesterday."
Correct : "I went shopping and had dinner with my friend yesterday."
Rule : "I went to shopping" => "I went shopping"
Condensed Rule : "went to shopping" => "went shopping"

To obtain the shortest possible rules for auto-editing, we propose a simple idea to check if the left hand side K_L can be condensed to a shorter one, without boosting its fre_{S+} significantly. If yes, then it implies we have found a shorter erroneous pattern which also occurs rarely in the correct corpus. For example, for the erroneous pattern "I went to shopping", table 5 shows the frequency of each possible subsequence in the correct corpus. Apparently "am worry about" is the most condensed rule without occurring more than four times in the correct corpus.

Table 5. example for condensing a rule

sentence segment	worry	worry about	am worry	am worry about	I am worry about
frequency	1446	225	206	3	1

Below is the algorithm for rule condensing. If the frequency of the condensed erroneous rule is smaller than a pre-defined frequency $N_{condense}$ which is observed from experiment,

we will accept it as a condensed erroneous pattern. We remove the same words from the K_R to produce the corresponding correct pattern. The condense process repeats until any of the word to be removed in K_L does not occur in the K_R. The pseudo code of condensing rules is shown in Figuire 3.

Algorithm 4 Rule Condensing.

Input: Set R = rule set
Output: Set V = reduced rule set

```
1: begin
2:   foreach rl ∈ R do
3:     reduced_rule ⇐ empty
4:     S ⇐ all the substrings of error pattern in rl
5:     foreach f ∈ S do
6:       if frequency(f) ≤ N_condense then
7:         reduced_rule ⇐ S
8:         BREAK
9:       end if
10:    end for
11:    remove the words which disappear in rl
12:  end for
13: end
14: return condensed_rule
```

Figure 3. pseudo code of condensing rules

The final step of the refinement is to rank the rules based on their qualities. We propose two plausible strategies to rank the rules. First, it is possible to rank the rules according to its fre_{S+} from low to high. That says, a rule is less likely to be correct something right into something wrong if its fre_{S+} is low. Second, it is possible to rank the rules according to the number of sentences in the corpus that can be applied using it. The first strategy is similar to the definition of *precision* while the second is closer to the meaning of *recall*.

4 Experiments

We retrieve 310967 pairs of English sentences from the "Lang-8" as our parallel corpus and the system finally produces 110567 rules. In the evaluation, we invite three people major in English to annotate our rules. Then we demonstrate an auto-editing system to show how such rules can be applied.

4.1 Evaluation

We rank all rules according to its fre_{S+} and four English-major people are invited to annotate the top 1500 ranked rules and we have two annotations for each rule. The labels for annotations are *correct, mostly correct, mostly wrong, wrong,* and *depends on context*. The following tables are the experiment results. Besides, there is fair agreement between the two annotations by kappa value equals to 0.49835.

Table 6. the average result of top 1500 annotated rules

	correct	mostly correct	mostly wrong	wrong	depends on context
R1~R1500	53.96%	12.96%	0.92%	4.5%	27.66%

4.2 Discussion

We also perform some manual analysis on the rules. The result is shown in Table 7 and we can know that the top 3 types of correction (67% of rules) are about spelling errors, collocation and phrase, and agreement of subject and verb. Besides, most of the incorrect rules would lead to false suggestions and 83% of rules belonging to *depend on context* category are about chunks and phrase.

Table 7. analysis on rules

I. Correct & Mostly Correct (67% of Rules)	**%**
1. spelling	60%
2. collocation and phrase (sequence of words which co-occur more often than would be expected by chance	15%
3. agreement of subject and verb	7%
4. choice of verb tense	5%
5. verb+ing forms and infinitives	2%
6. choice of the proper article	1%
7. pluralization (irregular noun)	1%
8. capitalization (use of capital letter)	1%
9. Other (use of preposition, word choice, cohesive devices, elliptical forms, punctuation, parts of speech, count and noncount nouns…etc.)	8%
II. Wrong & Mostly Wrong (0.9% of Rules)	**%**
1. suggestions of wrong corrections	97%
2. errors not to be spotted and corrected	3%
III. Depends on Context and/or Writers' Intention (32.1% of Rules)	**%**
1. correctness of the chunks/phrases	83%
2. verbal and verb tense	5 %
3. spelling (more than one possibility)	3%
4. word choice	2%
5. Others (use of preposition, conjunction, cohesive devices, parts of speech…etc.)	7%

Figure4. displays the rules histograms when we rank a rule according to the number of sentence pairs it can be applied to in the corpus.

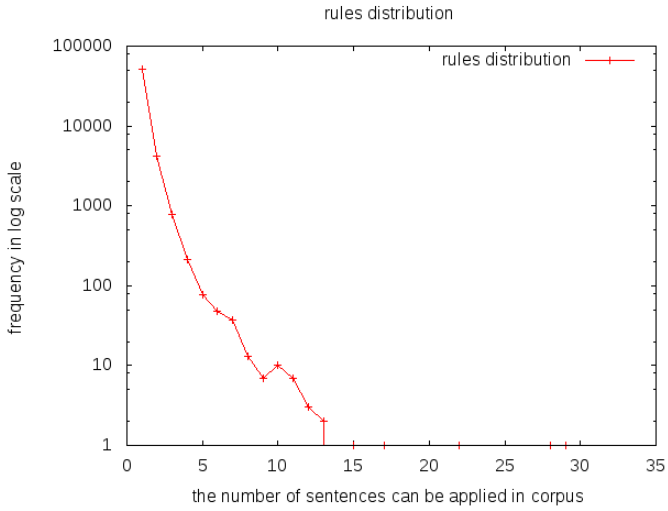

Figure 4. rule distribution

The following are some example rules discovered by our system. Such errors can hardly be corrected by Microsoft Word 2007 grammar checker.

Table 8. example rules

example rules
am worry about => am worried about
help me to study => help me study
I will appreciate it => I would appreciate it
went to shopping => went shopping
am so exciting => am so excited
waked => woke
look forward to read => look forward to reading
for read my => for reading my
The street name => The street's name
to playing with => to play with
He promised to me => He promised me
asked repeat => repeatedly asked
Have you listen to => Have you listened to
It's rains => It's raining
I ate a milk => I had milk
for the long time => for a long time
don't cooking => don't cook
will success => will succeed
don't know what happen => don't know what happened

4.3 Auto-editing System

We construct an online, real-time auto-editing system to demonstrate the usefulness of our rules to provide editorial assistance. We first try to test whether a part of the real-time typing sentence can match with the erroneous patterns. If yes (it will be marked in red), then we apply the correction rule to suggest replacing it with the correct pattern. The users can click the correct part (marked in green) to tell the system such correction is accepted. The link of our system is: http://mslab.csie.ntu.edu.tw/~kw/new_demo.html

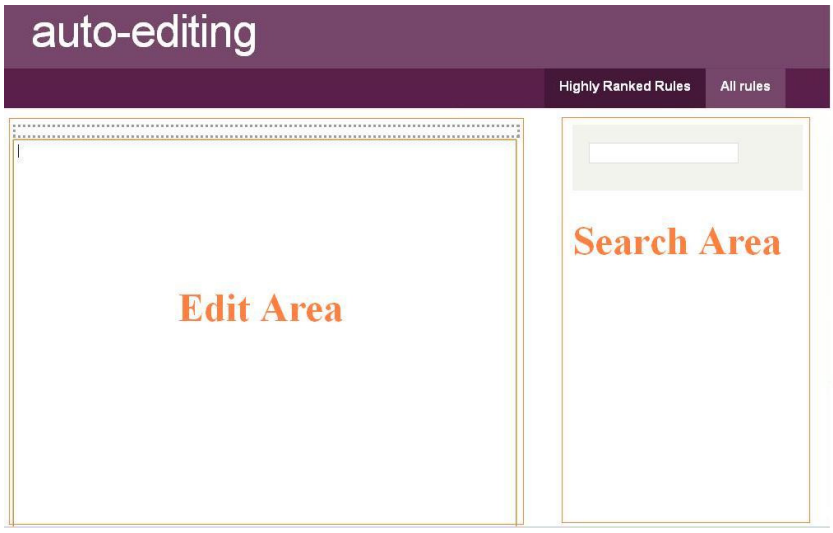

Figure 5. screenshot of demo system

The above screenshot is the entire system view. Two kinds of rules are provided. "Highly-ranked Rules" exploits only higher ranked rules and ignore lower-ranked one. "All Rules" utilizes every rule but suffers the risk of using incorrect ones.

Figure 6. screenshot of auto-editing

In the edit area, one can type sentences in English. If any of the rules is matched, the

suggested correction will appear on the above area in green. If the users agree with the corrections, they can click on the green word and the sentence will be edited accordingly.

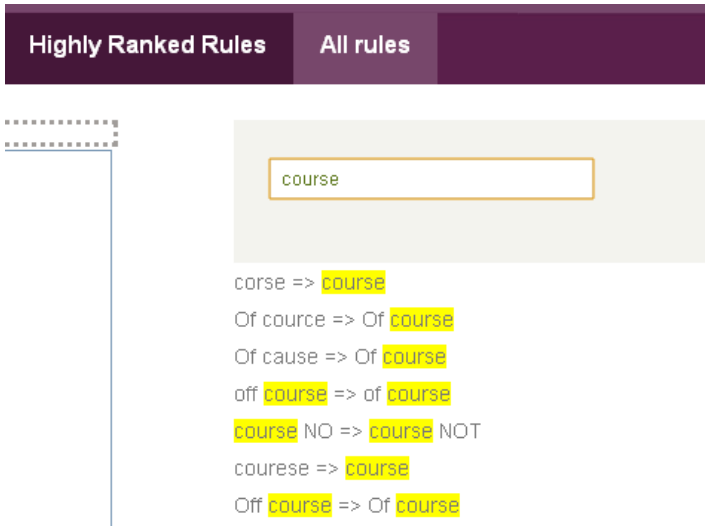

Figure 7. screenshot of keywords search in rule database

On the right hand side is a search engine for the related rules. Given a keyword (e.g. course), the system will return all rules that contains such word.

5 Conclusion and Future Work

In this work, we propose a language-universal framework that is capable of producing effective editing rules given a parallel editing corpus. The quality of rules can be assessed using one of our ranking strategies. Moreover, we have demonstrated the practical usage of the rules by constructing an Auto-Editing System to provide editorial assistance for language learners. In this paper, we produce correction rules without considering syntactic structure and POS (Part-of-Speech). In the future, we would like to make use of both of the two features. The syntactic structure retrieved by a parser can potentially make the rule more feasible since we then do not have to worry about splitting phrases into parts. On the other hand, substituting words in a rule into its POS creates more general rules that can be applied to more sentences.

6 References

[1] George E. Heidon. “Intelligent Writing Assistance.Handbook of Natural Language Processing.” Robert Dale, Hermann Moisi, and Harold Somers (ed.). Marcel Dekker.

[2] Lisa N. Michaud, Kathleen F. McCoy, and Christopher A. Pennington. “An Intelligent Tutoring System for Deaf Learners of Written English.” *In Proceeding of Fourth International ACM Conference on Assistive Technologies,* 2000

[3] Emily M. Bender, Dan Flickinger, Stephan Oepen, Annemarie Walsh, and Timothy

Baldwin. “Arboretum: Using a precision grammar for grammar checking in call.” *In Proceedings of the InSTIL/ICALL Symposium: NLP and Speech Technologies in Advanced Language Learning Systems,* 2004.

[4] Chodorow and Leacock, “An Unsupervised Method for Detecting Grammatical Errors.” *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference,* pp. 140-147, 2000.

[5] Matthieu Hermet, and Alain Désilets. “Using First and Second Language Models to Correct Preposition Errors in Second Language Authoring.” *Proceedings of the Fourth Workshop on Innovative Use of NLP for Building Educational Applications, pp. 64-72,* 2009.

[6] Chris Brocket, William B. Dolan, and Michael Gamon. “Correcting ESL errors using phrasal SMT techniques.” *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics,* 2006.

[7] Sun,G. Liu, X. Cong, G. Zhou, M. Xiong,Z. Lin, C.-Y., and Lee,J. ”Detecting Erroneous Sentences Using Automatically Mined Sequential Patterns. In Association of Computational Linguistics, 2007.

[8] Guihua Sun, Gao Cong, Xiaohua Liu, Chin-Yew Lin, and Ming Zhou. “Mining Sequential Patterns and Tree Patterns to Detect Erroneous Sentences.” *Proceedings of the 22nd national conference on Artificial intelligence - Volume 1,* 2007.

[9] Shi,Y.,andZhou,L. “Error Detection Using Linguistic Features.” *Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing,* 2005[10] V. I. Levenshtein. “Binary codes capable of correcting deletions, insertions and reversals”. Doklady Akademii Nauk SSSR **163**(4) p845-848, 1965,also Soviet Physics Doklady **10**(8) p707-710, Feb 1966

[11] Yajuan L¨ u and Ming Zhou., *“Collocation translation ac*quisition using monolingual corpora”, In Proceeding of Association for Computational Linguistics, 2004.