POSTECH Grammatical Error Correction System in the CoNLL-2014 Shared Task

Kyusong Lee, Gary Geunbae Lee
Department of Computer Science and Engineering
Pohang University of Science and Technology
Pohang, Korea
{Kyusonglee,gblee}@postech.ac.kr

Abstract

This paper describes the POSTECH grammatical error correction system. Various methods are proposed to correct errors such as rule-based, probability n-gram vector approaches and router-based approach. Google N-gram count corpus is used mainly as the correction resource. Correction candidates are extracted from NUCLE training data and each candidate is evaluated with development data to extract high precision rules and n-gram frames. Out of 13 participating teams, our system is ranked 4th on both the original and revised annotation.

1 Introduction

Automatic grammar error correction (GEC) is widely used by learners of English as a second language (ESL) in written tasks. Many methods have been proposed to correct grammatical errors; these include methods based on rules (Naber, 2003), on statistical machine translation (Brockett et al., 2006), on machine learning, and on n-grams (Alam et al., 2006). Early research (Han et al., 2006; De Felice, 2008; Knight & Chander, 1994; Nagata et al., 2006) on error correction for non-native text was based on well-formed corpora.

Most recent work (Cahill et al., 2013; Rozovskaya & Roth, 2011; Wu & Ng, 2013) has used machine learning methods that rely on a GE-tagged corpus such as NUCLE, Japanese English Learner corpus, and Cambridge Learner Corpus (Dahlmeier et al., 2013; Izumi et al., 2005; Nicholls, 2003), because well-formed and GE-tagged approaches are closely related to each other, can be synergistically combined. Therefore, research using both types of data has also been conducted (Dahlmeier & Ng, 2011). Moreover, a meta-classification method using several GE-tagged corpora and a native corpus has been proposed to correct the grammatical errors (Seo et al., 2012). A meta-classifier approach has been proposed to combine a language model and error-specific classification for correction of article and preposition errors (Gamon, 2010). Web-scale well-formed corpora have been successfully applied to grammar error correction tasks instead of using error-tagged data (Bergsma et al., 2009; Gamon et al., 2009; Hermet et al., 2008). Especially in the CoNLL-2013 grammar error correction shared task, many of the high-ranked teams (Kao et al., 2013; Mark & Roth, 2013; Xing et al., 2013) exploited the Google Web-1T n-gram corpus. The major advantage of using these web-scale corpora is that extremely large quantities of data are publicly available at no additional costs; thus fewer data sparseness problems arise compared to previous approaches based on error-tagged corpora.

We also use the Google Web-1T n-gram corpus. We extract the candidate pairs (original erroneous text and its correction) from NUCLE training data. We use a router to choose the best frame to compare the n-gram score difference between the original and replacement in a given candidate pair.

The intuition of our grammar error correction method is the following: First, if the uni-gram count is less than some threshold, we assume that the word is erroneous. Second, if the replacement word n-gram has more frequent than the original word n-gram, it presents strong evidence for correction. Third, depending on the candidate pair, tailored n-gram frames help to correct errors accurately. Fourth, only high precision method and rules are applied. If correction precision on a candidate pair is less than 30% in development data,

Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task, pages 65–73,
Baltimore, Maryland, 26-27 July 2014.

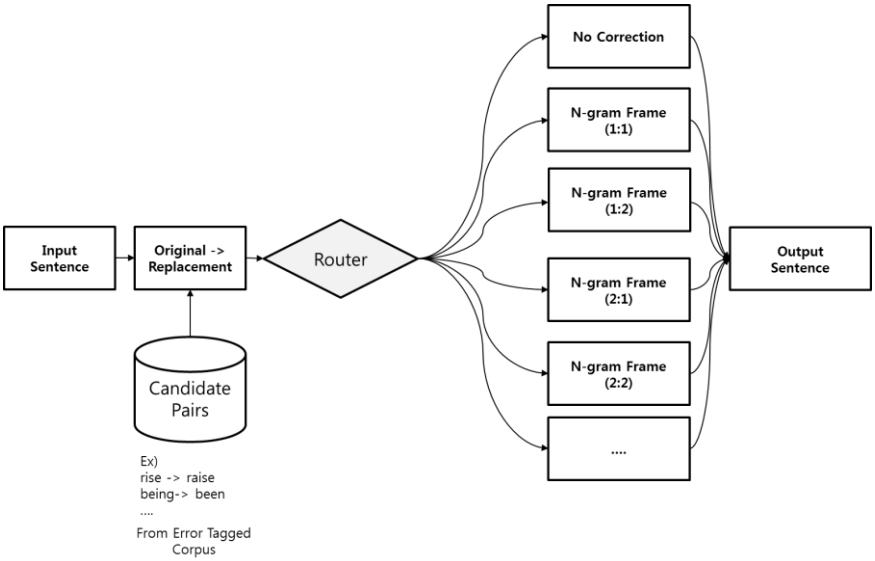

Figure 1. Overall Process of Router-based Correction

we do not make a correction for the candidate pair at runtime.

In the CoNLL-Shared Task, objectives were presented yearly. In 2012, the objective was to correct article and preposition errors; in 2013, it was to correct article, preposition, noun number, verb form, and subject-verb agreement errors. This year, the objective is to correct all errors. Thus, our method should also correct preprocessing and spelling errors. Detailed description of the shared task set up, data set, and evaluation about the CoNLL-2014 Shared Task is explained in (Ng et al., 2014)

2 Data and Recourse

The Google Web-1T corpus contains 10^{12} words of running text and the counts for all 10^9 five-word sequences that appear > 40 times (Brants & Franz, 2006). We used the NUS Corpus of Learner English (NUCLE) training data to extract the candidate pairs and CoNLL-2013 Official Shard Task test data as development data. We used the Stanford parser (De Marneffe & Manning, 2008) to extract part-of-speech, dependency, and constituency trees.

3 Method

3.1 Overall Process

We correct the errors in the following order:
Tokenizing → spelling error correction → punctuation error correction → N-gram Vector Approach for Noun number (Nn) → Router-based Correction (Deletion Correction → Insertion Correction → Replacement) for various error types → Rule-based method for verb errors. Between each pair of step, we parse, tag, and tokenize again using the Stanford parser because the previous correction affects parsing, tagging, and tokenizing results.

3.2 Preprocessing

Because the correction task is no longer restricted to five error types, tokenizing and spelling error correction have become critical for error correction. To detect tokenizing error such as "civilizations.It", a re-tokenzing process is necessary. If a word contains a comma, punctuation (e.g., ',' or '.') and the word count in Google n-gram is less than some threshold (here, 1000), we tokenize the word, e.g., as "civilizations . It". We also correct spelling errors by referring to the Google n-gram word count. If the word uni-gram count is less than a threshold (here, 60000) and the part-of-speech (POS) tag is not NNP or NNPS, we assume that the word has o ne or more errors. The threshold is set based on the development set. We use the Enchant Python Library to correct the spelling errors[1]. However, using only one best result is not very accurate. Thus, among the best results in the Enchant Python Library, we select the one best word, i.e. that word with the highest frequency in the Google n-gram corpus. Using NUCLE training data, rules are constructed for comma, punctuation, and other errors (Table 3).

[1] http://abisource.com/projects/enchant/

3.3 Candidate Generation

Selecting appropriate correction candidates is critical for the precision of the method. In article and noun number correction, the number of candidates is small: 'a','an','the' in article correction, 'plural' or 'singular' in noun number correction. However, the number of correction candidates can be unlimited in wrong collocation/idiom errors. Reducing the number of candidates is important in the grammar error correction task.

Nn Correction Candidate: noun number correction has just one replacement candidate. If the word is plural, its correction candidate is singular, and vice versa. The language tool[2] can perform these changes.

Other Correction Candidate: for corrections other than noun number, candidates are selected from the GE-tagged corpus. A total of 4206 pairs were extracted. We use the notation of candidate pair (o→r), which links the original word (o) and its correction candidate (r). In the deletion correction step, we determine whether or not the word should be deleted. In the insertion correction step, we select the insertion position in a sentence as a space between two words. If o is ∅, insertion correction is required; if r is ∅, the pair deletion correction is required. We use the Stanford constituency parser (De Marneffe & Manning, 2008) to extract a noun phrase; if it does not contain a determiner or article, we insert one in front of the noun phrase; if the noun in the noun phrase is singular, 'the', 'a', and, 'an' are selected an insertion candidates; if the noun is plural, only 'the' is selected as an insertion candidate. We only apply insertion correction at ArtOrDet, comma errors, and preposition; we skip insertion correction for other error types because selecting an insertion position is difficult and if every position is selected as insertion position, precision decrease.

4 N-gram Approach

We used the following notation.

N(o)	n-gram vector in original sentence
N(r)	n-gram vector in replacement sentence
$n(o)_i$	i th element in N(o)
$n(r)_i$	i th element in N(r)
N[i:j]	n-gram vector from i th element to j th element

[2] http://www.languagetool.org

Web-scale data have also been used successfully in many other research areas, such as lexical disambiguation (Bergsma et al., 2009). Most NLP systems resolve ambiguities with the help of a large corpus of text, e.g.:

- The system tried to decide {among, between} the two confusable words.

Disambiguation accuracy increases with the size of the corpus. Many systems incorporate the web count into their selection process. For the above example, a typical web-based system would query a search engine with the sequences "decide among the" and "decide between the" and select the candidate that returns the most hits. Unfortunately, this approach would fail when disambiguation requires additional context. Bergsma (2009) suggested using the context of samples of various lengths and positions. For example, from the above the example sentence, the following 5-gram patterns can be extracted:

- system tried to decide {among, between}
- tried to decide {among, between} the
- to decide {among, between} the two
- decide {among, between} the two confusable
- {among, between} the two confusable words

Similarly, four 4-gram patterns, three 3-gram patterns and two 2-gram patterns are extracted by spanning the target. A score for each pattern is calculated by summing the log-counts. This method was successfully applied in lexical disambiguation. Web-scale data were used with the count information specified as features. Kao et al. (2013) used a "moving window (MW)" :

$$MW_{i,k}(\mathrm{w}) = \{w_{i-j}, \ldots, w_{i-j+(k-1)}, j = 0, k-1\} \quad (1)$$

where i denotes the position of the word, k the window size and w the original or replacement word at position i. The window size is set to 2 to 5 words. MW is the same concept as the SUMLM:

$$S_{i,k}(w) = \sum_{ngram \in MW_k(w)} count(ngram) \quad (2)$$

Both approaches apply the sum of all MWs in (1). Our approach is based on the MW method. The difference is that instead of summing all the MWs, we consider only one best MW which is referred to here as a frame. The following sentences

demonstrate the case when the following words are the crucial features to correct errors:

- I will do it (in→at) home.
- We need (an→∅) equipment to solve problems.

However, following sentences demonstrate the case when preceding words is the crucial feature to correct errors:

- One (are→is) deemed to death at a later stage .
- But data that (shows→show) the rising of life expectancies

We investigated which frame is the best based on the development set, then router is trained to decide on the frame depending on the candidate pair.

4.1 Router-based N-gram Correction

A frame is a sequence of words around the target position. A frame is divided into a preceding frame and a following frame. The target position can be either a position of a target word (Figure 2a) or a position in which a candidate word is judged to be necessary (Figure 2b). Once the size (i.e., number of words) of frames is chosen, several forms of frames (n; m) with different sizes of preceding (n) and following (m) words are possible.

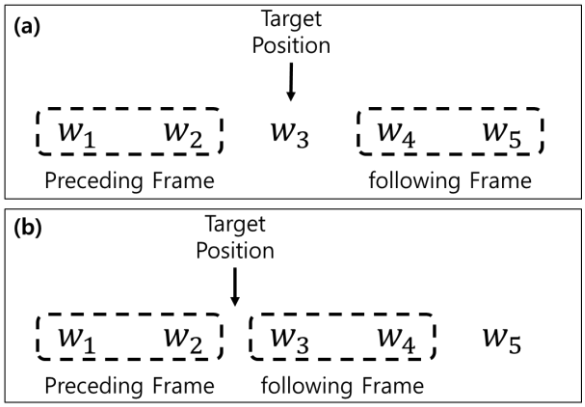

Figure 2. Frame for n-gram

The router is designed to take care of two stages (training, run-time) error correction. During training, the router selects the best frame for each candidate pair. By testing each candidate pair with each frame in the development data; the frame with the best precision is selected as the best frame among (1;1), (1;2), (1;3), (2;1),(2:2), etc.

At the end of the training stage, the router has a list of pairs (x) which matches the best frame (y) associated with it (Table 1) as a result of comparing each candidate pair with one in the development corpus.

During runtime, the router assigns each candidate pair to the best frame to produce the output sentence (Figure 1). For example, for a sentence "This ability is not seen 40 years **back** where the technology advances were not as good as now ." the candidate pair for correction (back→ ago) is suggested. The best frame assigned by the router for this pair (1;1), which is "years back where". The best candidate frame for this is "year ago where". At this point, we query the count of "years back where" and "years ago where" from the Google N-gram Count Corpus; these counts are 46 and 1815 respectively. Because the count of "years ago where" is greater than that of "years back where", the former is selected as the correct form. As a result, the sentence "This ability is not seen 40 years **back** where the technology advances were not as good as now." is corrected to "This ability is not seen 40 years **ago** where the technology advances were not as good as now."

Table 1. Example of Trained Router

x (o→r)	y
(another→other)	(1;3)
(less→fewer)	(1;3)
(rise→raise)	(1;2)
(back→ago)	(1;1)
(could→can)	(2;1)
(well→good)	(2;1)
(near→∅)	No correction

Some words are allowed to have multiple best frames; in all the best frames, if a candidate word sequence is more frequent than an original word sequence in the Google count, then correction is made. The multiple frames are also trained from the development data set.

4.2 Probability n-gram Vector

We use the probability n-gram Vector approach to correct Nn. Most errors are corrected using the router-based method; however, training the router for every noun is difficult because the number of nouns is extremely large. Moreover, for noun number, we found that rather than considering one direction or one frame of n-gram, every direction of n-gram should be considered for better performance such as forward, backward, and two-way. Thus, the probability n-gram vector algorithm is applied only in the noun number error correction. We propose the probability n-gram vector method to correct grammatical errors to consider both directions, forward and backward. In a forward n-gram, the probability of each word is estimated

depending on the preceding word. On the other hand, in a backward n-gram the probability of each word is estimated depending on the following words. When the probability of a candidate word is higher than original word, we replace the original with the candidate word in the correction step.

Probability n-gram vectors are generated from the original word and a candidate word (Figure 3). Rather than using a single sequence of n-gram probability, we apply contexts of various lengths and positions. We applied the probability information using the Google n-gram count information as in the following equation:

$$P(w_i|w_{i-2},w_{i-1}) = \frac{C(w_{i-2},w_{i-1}w_i)}{C(w_{i-2},w_{i-1})}$$

Moreover, rather than calculating one word's probability given n words such as $P(w_i|w_{i-1},w_{i-2},w_{i-3})$, our model calculates the probability of m words given an n word sequence. The following is an example 4-gram with forward probability:

- m = 3, n = 1 $P(w_{i-2},w_{i-1}w_i|w_{i-3})$
- m = 2, n = 2 $P(w_{i-1},w_i|w_{i-3},w_{i-2})$
- m = 1, n = 3 $P(w_i|w_{i-3},w_{i-2},w_{i-1})$.

We construct a 40-dimensional probability vector with forward and backward probabilities considering of twenty 5-grams, twelve 4-grams, six 3-gram, and two 2-gram. Additionally, the elements of the n-gram vector are detailed in Table 2.

Table 2: The elements of n-gram vector

5-GRAM
$n_0 = P(w_i\|w_{i+1}w_{i+2}w_{i+3}w_{i+4})$ backward
$n_1 = P(w_i\|w_{i-4}w_{i-3}w_{i-2}w_{i-1})$ forward
$n_2 = P(w_iw_{i+1}\|w_{i+2}w_{i+3}w_{i+4})$ backward
……..
4-GRAM
$n_{20} = P(w_i\|w_{i+1}w_{i+2}w_{i+3})$ backward
$n_{21} = P(w_i\|w_{i-3}w_{i-2}w_{i-1})$ forward
……..
3-GRAM
$n_{32} = P(w_i\|w_{i+1}w_{i+2})$ backward
$n_{33} = P(w_i\|w_{i-2}w_{i-1})$ forward
$n_{34} = P(w_iw_{i+1}\|w_{i+2})$ backward
$n_{35} = P(w_{i-1}w_i\|w_{i-2})$ forward
$n_{36} = P(w_{i-1}w_i\|w_{i+1})$ backward
$n_{37} = P(w_iw_{i+1}\|w_{i-1})$ forward
2-GRAM
$n_{38} = P(w_i\|w_{i+1})$ backward
$n_{39} = P(w_i\|w_{i-1})$ forward

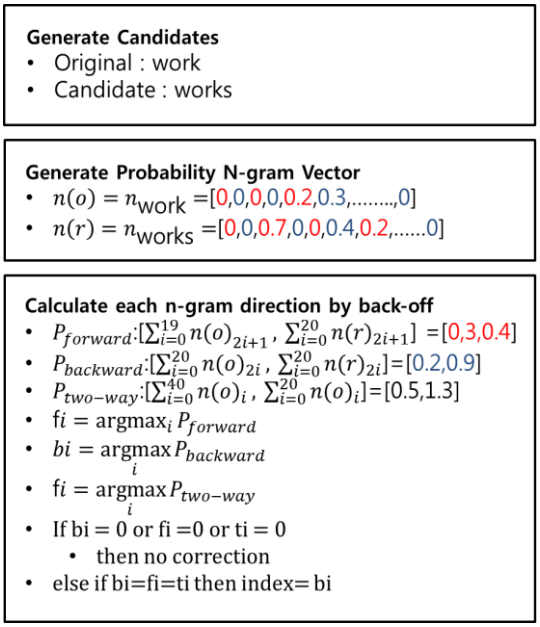

Figure 3. Overall process of Nn Correction

Back-Off Model: A high-order n-gram is more effective than a low-order n-gram. Thus, we applied back-off methods (Katz, 1987) to assign higher priority to higher order probabilities. If all elements in 5-gram vectors are 0 for both the original and candidate sentence, which means $\sum_{i=0}^{19}\{n(o)_i + n(r)_i\} = 0$, we consider 4-gram vectors ($N_{[20:31]}$). If 4-gram vectors are 0, we consider 3-gram vectors. Moreover, when the proposed method calculates each of the forward, backward and two-way probabilities, the back-off method is used to get each score.

Correction: Here, we explain the process of error correction using n-gram vectors. First, we generate Nn error candidates. Second, we construct the n-gram probability vector for each candidate. The back-off method is applied in N(o)+N(r), The vector contains various directions and ranges of probabilities of words given a sample sentence. We then calculate forward n-gram score by summing even elements in the vector. We calculate the backward n-gram by summing odd elements in Table 2. Next, the two-way n-gram is calculated by summing all elements for both directions n-gram. If forward, backward, and two-way n-grams have higher probabilities for the candidate word, we select the candidate as corrected word (Figure 3).

Algorithm Rule1-Comma

```
1:  function rule1( tok_sent, tok_pos)
2:    for i ← 0 ... len(tok_sent) do
3:      if tok_sent[i] in [ However', 'Therefore', 'Thus'] and not tok_sent[i + 1] == ',' then
4:        tok_sent[i]= tok_sent[i] + ' ,'
```

Algorithm Rule2-preposition

```
1:  function rule2( tok_sent, tok_pos)
2:    for i ← 0 ... len(tok_sent) do
3:      if tok_sent[i] = 'according' and not tok_sent [i+1] = 'to'
4:        tok_sent [i+1] = 'to '+ tok_sent [i+1]
```

Algorithm Rule3-Subject Verb Agreement

```
1:  function rule3( tok_sent, tok_pos)
2:    for i ← 0 ... len(tok_sent) do
3:      if tok_sent[i] is 'which'
4:        if tok_pos[i − 1] == 'NNS' and tok_pos[i + 1] == 'VBZ' then
5:          tok_sent[i + 1]= changeWordForm (tok_sent[i + 1], 'VBP')
6:        else if tok_pos[i − 1] == 'NNS' and tok_pos[i + 1] == 'NNS' then
7:          tok_sent[i + 1]= =changeWordForm(tok_sent[i + 1], 'VBP')
8:        else if tok_pos[i − 1] == 'NN' and tok_pos[i + 1]== 'are' then
9:          tok_sent[i + 1]= = is
10:       else if tok_pos[i − 1] == 'NN' and tok_pos[i + 1] in ['VBP','VB','NN'] then
11:         tok_sent[i + 1]= = makePlural(tok_sent[i + 1])
```

Algorithm Rule4-Subject Verb Agreement

```
1:  function rule4( tok_sent, tok_pos)
2:    for i ← 0 ... len(tok_sent) do
3:      if not ( tok_pos[i]is 'VBZ' and ['NN','this','it','one','VBG'] in tok_pos[i − 5: i]) then
4:        tok_cand ←changeWordForm( tok_word[i], 'VBP')
5:      else if not ( tok_pos[i]is 'VBP' and ['I','we','they','and'] in tok_sent[i − 5: i]) then
6:        tok_cand ←changeWordForm( tok_word[i], 'VBZ')
7:      else if not ( tok_pos[i]is 'NN' and ['be','ing'] in tok_sent[i − 5: i]) then
8:        tok_cand ←changeWordForm( tok_word[i], 'VBN')
9:      original = ngramCount( tok_sent), candidate =ngramCount(tok_cand)
10:     If original < candidate then
11:       Return tok_cand
```

Table 3. Examples of Rules

5 Verb Correction (Rule-based)

There are several types of verb errors in non-native text such as verb tense, verb modal, missing verb, verb form, and subject-verb-agreement (SVA). Among these errors, we attempt to correct SVA errors using rule-based methods (Table 3). In non-native text, parsing and tagging errors are inevitable, and it may cause false alarm. Thus, instead of dependency parsing to find subject and verb, we consider the preceding five words because erroneous sentences often contain dependency errors. Moreover, in erroneous sentences, POS tagging accuracy is lower than native text. Thus, NN and VB are misclassified, as are VBZ and NNS. A rule is used that encodes the relevant linguistic knowledge that these words or POSs should not occur in the five positions preceding the VBZ: 'NN', 'this', 'it' ,'one', 'VBG'. Moreover, words that preceded and follow 'which' should agree in verb form, as indicated in Rule3 and Rule4.

6 Experiment

The CoNLL-2014 training data consist of 1,397 articles together with gold-standard annotation.

Table 4. Performance on each error type

	Original annotation			Revised annotation		
	Precision	Recall	F0.5	Precision	Recall	F0.5
N-gram (Nn)	31.0	6.55	17.75	42.28	9.0	24.31
Rule (Verb)	28.95	1.12	4.86	31.17	1.29	5.52
Rule (Mec)	49.34	5.47	18.94	52.16	6.17	20.93
Router (Others)	28.11	12.49	22.49	35.29	15.45	28.08
All	**34.51**	**21.73**	**30.88**	**41.28**	**25.59**	**36.77**

The documents are a subset of the NUS Corpus of Learner English (NUCLE). We use the Max-Match (M2) scorer provided by the CoNLL-2014 Shared Task. The M2 scorer works by using the set that maximally matches the set of gold-standard edits specified by the annotator as being equal to the set of system edits that are automatically computed and used in scoring (Dahlmeier & Ng, 2012). The official evaluation metric is F0.5, weighting precision twice as much as recall. We achieve F0.5 of 30.88; precision of 34.51; recall of 21.73 in the original annotation (Table 4). After original official annotations announced by organizers (i.e., only based on the annotations of the two annotators), another set of annotations is offered based on including the additional answers proposed by the 3 teams (CAMB, CUUI, UMC). The improvement gap between the original annotation and the revised annotation of our team (POST) is 5.89%. We obtain the highest improvement rate except for the 3 proposed teams (Figure 4), F0.5 of 36.77; precision of 41.28; recall of 25.59 in the revised annotation. Our system achieves the 4th highest scores of 13 participating teams based on both the original and revised annotations. To analyze the scores of each of the error types and modules, we apply the method of n-gram vector (Nn), rule-based (Verb, Mec), and router-based (others) separately in both the original and the revised annotation of all error types. We achieve high precision by rules at the Mec which indicates punctuation, capitalization, spelling, and typos errors. Additionally, the Nn type has the highest improvement gap between the original and revised annotation (17% → 24.31 of F0.5). In order for our team to improve the high precision in the rule-based approach, we tested potential rules on the development data and kept a rule only if its precision on that data set was 30% or greater. When we trained router, the same strategy was conducted. If a frame could not achieve 30% precision, we assigned the candidate pair as "no correction" in the router. These constraints achieve precision of 30 % in most error types.

7 Discussion

Although preposition errors are frequently committed in non-native text, we mostly skip the correction of preposition error. This is because assigning prepositions correctly is extremely difficult, because (1) the preposition used can vary (e.g., Canada: 'on the weekend' vs. Britain 'at the weekend'); (2) in a given location, more than one preposition may be possible, and the choice affects the meaning (e.g., 'on the wall', vs. 'at the wall'). Verb errors can consist of many multi-

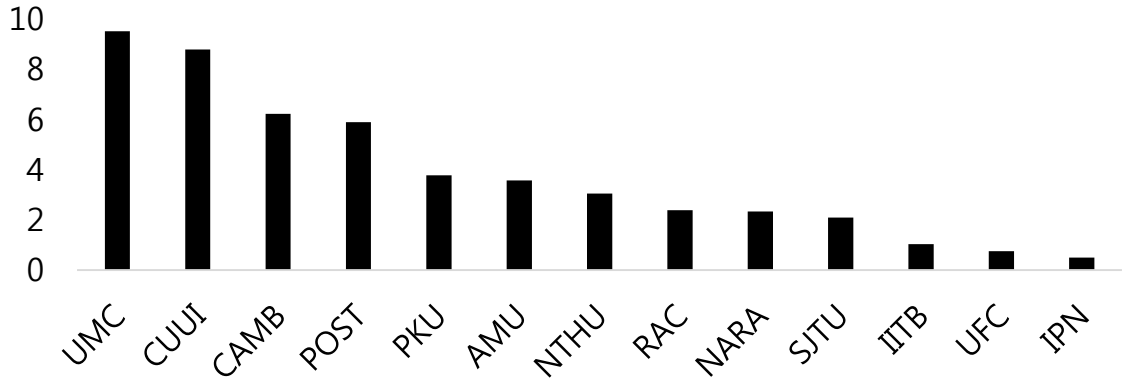

Figure 4. Improvement gap between the original annotation and revised annotation of each team

word errors due to errors of usages of passive and active voice. (e.g. release→be released). Our current system cannot correct these multi-words errors, for three reasons. First, if the original example consists of one word and the optimal replacement consists of two words, n-gram scores cannot be applied easily to compare probabilities between them. Second, the n-gram approach also fails if the distance between subject and verb is more than 5. Third, multiply dependent errors are critical for verb error correction. For example, noun number, determiner, and subject verb agreement are often dependent upon each other: e.g. "And once this happens, privacy does not exist any more and people's (life→lives) (is→are) under great threaten." The correction order will be important when all error type must be corrected simultaneously.

Grammar error correction is a challenging problem. In CoNNL-2013, more than half of the related teams obtained F-score < 10.0. This low performance in the grammar error correction can be explained by several reasons, which indicate the present limitations of grammar correction systems.

Among a total of 4206 pairs, we only use small amount of candidate pairs, 215 pairs are used for candidate pairs. The other 3991 pairs are discarded in the router training step because these pairs cannot be corrected by the n-gram approach. Various classification methods and statistical machine translation based methods will be investigated in the router-based approach to find the tailored methods for the given word. A demonstration and progress of our grammar error correction system is available to the public[3].

8 Conclusion

We have described the POSTECH grammatical error correction system. We use the Google N-gram count corpus to detect spelling errors, punctuation, and comma errors. A rule-based method is used to correct verb, punctuation, comma errors and preposition errors. The Google corpus is also used for an n-gram vector approach and a router-based approaches. Currently we use the router to select the best frame. In the future, we will train a router to select the best method among classification, n-gram approach, statistical machine translation-based method and pattern matching approaches. A machine learning method will be used to train the router with various features.

Acknowledgements

This research was supported by the MSIP(The Ministry of Science, ICT and Future Planning), Korea and Microsoft Research, under IT/SW Creative research program supervised by the NIPA(National IT Industry Promotion Agency) (NIPA-2013- H0503-13-1006) and this research was supported by the Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology(2010-0019523).

References

Han, Na-Rae, Chodorow, Martin, & Leacock, Claudia. (2006). Detecting errors in English article usage by non-native speakers.

Alam, Md Jahangir, UzZaman, Naushad, & Khan, Mumit. (2006). N-gram based statistical grammar checker for Bangla and English.

Bergsma, Shane, Lin, Dekang, & Goebel, Randy. (2009). *Web-Scale N-gram Models for Lexical Disambiguation*. Paper presented at the IJCAI.

Brants, Thorsten, & Franz, Alex. (2006). The Google Web 1T 5-gram corpus version 1.1. *LDC2006T13*.

Brockett, Chris, Dolan, William B, & Gamon, Michael. (2006). *Correcting ESL errors using phrasal SMT techniques.* Paper presented at the Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics.

Cahill, Aoife, Madnani, Nitin, Tetreault, Joel, & Napolitano, Diane. (2013). *Robust Systems for Preposition Error Correction Using Wikipedia Revisions.* Paper presented at the Proceedings of NAACL-HLT.

Dahlmeier, Daniel, & Ng, Hwee Tou. (2011). *Grammatical error correction with alternating structure optimization.* Paper presented at the Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1.

Dahlmeier, Daniel, & Ng, Hwee Tou. (2012). *Better evaluation for grammatical error correction.* Paper presented at the Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.

[3] http://isoft.postech.ac.kr/grammar

Dahlmeier, Daniel, Ng, Hwee Tou, & Wu, Siew Mei. (2013). *Building a large annotated corpus of learner English: The NUS corpus of learner English*. Paper presented at the Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications.

De Felice, Rachele. (2008). *Automatic error detection in non-native English.* University of Oxford.

De Marneffe, Marie-Catherine, & Manning, Christopher D. (2008). *The Stanford typed dependencies representation.* Paper presented at the Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation.

Gamon, Michael. (2010). *Using mostly native data to correct errors in learners' writing: a meta-classifier approach.* Paper presented at the Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics.

Gamon, Michael, Leacock, Claudia, Brockett, Chris, Dolan, William B, Gao, Jianfeng, Belenko, Dmitriy, & Klementiev, Alexandre. (2009). Using statistical techniques and web search to correct ESL errors. *Calico Journal, 26*(3), 491-511.

Hermet, Matthieu, Désilets, Alain, & Szpakowicz, Stan. (2008). Using the web as a linguistic resource to automatically correct lexico-syntactic errors.

Izumi, Emi, Uchimoto, Kiyotaka, & Isahara, Hitoshi. (2005). *Error annotation for corpus of Japanese learner English.* Paper presented at the Proceedings of the Sixth International Workshop on Linguistically Interpreted Corpora.

Kao, Ting-Hui, Chang, Yu-Wei, Chiu, Hsun-Wen, & Yen, Tzu-Hsi. (2013). CoNLL-2013 Shared Task: Grammatical Error Correction NTHU System Description. *CoNLL-2013, 20*.

Katz, Slava. (1987). Estimation of probabilities from sparse data for the language model component of a speech recognizer. *Acoustics, Speech and Signal Processing, IEEE Transactions on, 35*(3), 400-401.

Knight, Kevin, & Chander, Ishwar. (1994). *Automated postediting of documents.* Paper presented at the AAAI.

Mark, Alla Rozovskaya Kai-Wei Chang, & Roth, Sammons Dan. (2013). The University of Illinois System in the CoNLL-2013 Shared Task. *CoNLL-2013*, *51*, 13.

Naber, Daniel. (2003). A rule-based style and grammar checker. Diploma Thesis

Nagata, Ryo, Morihiro, Koichiro, Kawai, Atsuo, & Isu, Naoki. (2006). *A feedback-augmented method for detecting errors in the writing of learners of English.* Paper presented at the Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics.

Ng, Hwee Tou , Wu, Siew Mei , Briscoe, Ted , Hadiwinoto, Christian , Susanto, Raymond Hendy, & Bryant, Christopher (2014). *The CoNLL-2014 Shared Task on Grammatical Error Correction.* Paper presented at the the Eighteenth Conference on Computational Natural Language Learning: Shared Task (CoNLL-2014 Shared Task), Baltimore, Maryland, USA.

Nicholls, Diane. (2003). *The Cambridge Learner Corpus: Error coding and analysis for lexicography and ELT.* Paper presented at the Proceedings of the Corpus Linguistics 2003 conference.

Rozovskaya, Alla, & Roth, Dan. (2011). Algorithm selection and model adaptation for ESL correction tasks. *Urbana, 51*, 61801.

Seo, Hongsuck, Lee, Jonghoon, Kim, Seokhwan, Lee, Kyusong, Kang, Sechun, & Lee, Gary Geunbae. (2012). *A meta learning approach to grammatical error correction.* Paper presented at the Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2.

Wu, Yuanbin, & Ng, Hwee Tou. (2013). *Grammatical error correction using integer linear programming.* Paper presented at the Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics.

Xing, Junwen, Wang, Longyue, Wong, Derek F, Chao, Lidia S, & Zeng, Xiaodong. (2013). UM-Checker: A Hybrid System for English Grammatical Error Cor-rection. *CoNLL-2013*, 34.

Yannakoudakis, Helen, Briscoe, Ted, & Medlock, Ben. (2011). *A New Dataset and Method for Automatically Grading ESOL Texts.* Paper presented at the ACL.