# Sentence-level Rewriting Detection

**Fan Zhang**
University of Pittsburgh
Pittsburgh, PA, 15260
zhangfan@cs.pitt.edu

**Diane Litman**
University of Pittsburgh
Pittsburgh, PA, 15260
litman@cs.pitt.edu

## Abstract

Writers usually need iterations of revisions and edits during their writings. To better understand the process of rewriting, we need to know what has changed between the revisions. Prior work mainly focuses on detecting corrections within sentences, which is at the level of words or phrases. This paper proposes to detect revision changes at the sentence level. Looking at revisions at a higher level allows us to have a different understanding of the revision process. This paper also proposes an approach to automatically detect sentence revision changes. The proposed approach shows high accuracy in an evaluation using first and final draft essays from an undergraduate writing class.

## 1 Introduction

Rewriting is considered to be an important process during writing. However, conducting successful rewriting is not an easy task, especially for novice writers. Instructors work hard on providing suggestions for rewriting (Wells et al., 2013), but usually such advice is quite general. We need to understand the changes between revisions better to provide more specific and helpful advice.

There has already been work on detecting corrections in sentence revisions (Xue and Hwa, 2014; Swanson and Yamangil, 2012; Heilman and Smith, 2010; Rozovskaya and Roth, 2010). However, these works mainly focus on detecting changes at the level of words or phrases. According to Faigley's definition of revision change (Faigley and Witte, 1981), these works could help the identification of *Surface Changes* (changes that do not add or remove information to the original text). However, *Text Changes* (changes that add or remove information) will be more difficult to identify if we only look at revisions within sentences. According to Hashemi and Schunn (2014), when instructors were presented a comparison of differences between papers derived from words, they felt the information regarding changes between revisions was overwhelming.

This paper proposes to look at the changes between revisions at the level of sentences. Comparing to detecting changes at the word level, detecting changes at the sentence level contains less information, but still keeps enough information to understand the authors' intention behind their modifications to the text. The sentence level edits could then be grouped and classified into different types of changes. The long-term goal of this project is to allow us to be able to identify both *Text Changes* and *Surface Changes* automatically. Students, teachers, and researchers could then perform analysis on the different types of changes and have a better understanding of the rewriting process. As a preliminary work, this paper explores steps toward this goal: First, automatically generate the description of changes based on four primitives: *Add*, *Delete*, *Modify*, *Keep*; Second, merge the primitives that come from the same purpose.

## 2 Related work

Hashemi and Schunn (2014) presented a tool to help professors summarize students' changes across papers before and after peer review. They first split the original documents into sentences and then built on the output of Compare Suite (CompareSuite, 2014) to count and highlight changes in different colors. Figure 1 shows a screenshot of their work. As we can see, the modifications to the text are misinterpreted. Line 66 in the final draft should correspond to line 55 and line 56 in the first draft, while line 67 and line 68 should be a split of line 57 in the first draft. However, line 67 is aligned to line 56 wrongly in their work. This wrong alignment caused many misrecognized modifications. According to Hashemi, the instructors who use the system think that the overwhelming information of changes make the system less useful. We hypothesize that since their work is based on analysis at the word level, although their approach might work for identifying differences within one sentence, it makes mistakes when sentence analysis is the primary concern.

Our work avoids the above problem by detecting differences at the sentence level. Sentence alignment is the first step of our method; further inferences about revision changes are then based on the alignments generated. We borrow ideas from the research on sentence alignment for monolingual corpora. Existing research usually focuses on the alignment from the text to its summarization or its simplification (Jing, 2002; Barzilay and Elhadad, 2003; Bott and Saggion, 2011). Barzilay and Elhadad (2003) treat sentence alignment as a classification task. The paragraphs are clustered into groups, and a binary classifier is trained to decide whether two sentences should be aligned or not. Nelken (2006) further improves the performance by using TF*IDF score instead of word overlap and also utilizing global optimization to take sentence order information into consideration. We argue that summarization could be considered as a special form of revision and adapted Nelken's approach to our approach.

Edit sequences are then inferred based on the results of sentence alignment. Fragments of edits that come from the same purpose will then be merged. Related work to our method is sentence clustering (Shen et al., 2011; Wang et al., 2009). While sentence clustering is trying to find and cluster sentences similar to each other, our work is to find a cluster of sentences in one document that is similar to one sentence in the other document after merging.

## 3 Sentence-level changes across revisions

### 3.1 Primitives for sentence-level changes

Previous work in educational revision analysis (Faigley and Witte, 1981; Connor and Asenavage, 1994) categorized revision changes to be either surface changes or text-based changes. With both categories, six kinds of changes were defined as shown in Table 1.

| Code | Explanation |
|---|---|
| Addition | Adding a word or phrase |
| Deletion | Omitting a word or phrase |
| Substitutions | exchange words with synonyms |
| Permutation | rearrange of words or phrases |
| Distribution | one segment divided into two |
| Consolidation | combine two segments into one |

Table 1: Code Definition by L.Faigley and S.Witte

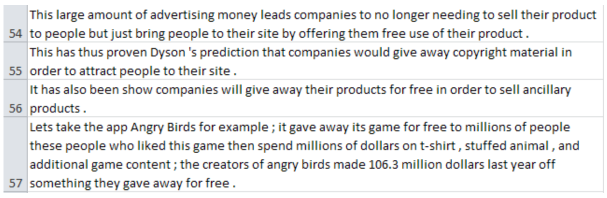
Different from Faigley's definition, we define only 4 primitives for our first step of edit sequence generation: *Add*, *Delete*, *Modify* and *Keep*. This definition is similar to Bronner's work (Bronner and Monz, 2012). We choose this definition because these 4 primitives only correspond to one sentence at a time. *Add*, *Delete*, *Modify* indicates that the writer has added/deleted/modified a sentence. *Keep* means the original sentence is not modified. We believe *Permutation*, *Distribution* and *Consolidation* as defined by Faigley could be described with these four primitives, which could be recognized in the later merge step.

### 3.2 Data and annotation

The corpus we choose consists of paired first and final drafts of short papers written by undergraduates in a course "Social Implications of Computing Technology". Students are required to write papers on one topic and then revise their own papers. The revisions are guided by other students' feedback based on a grading rubric, using a web-based peer review system. Students first submitted their original paper into the system, and then were randomly assigned to review and comment others' work according to the writing rubric. The authors would receive the others' anonymous comments, and then could choose to revise their work based on others' comments as well as their own insights obtained by reviewing other papers.

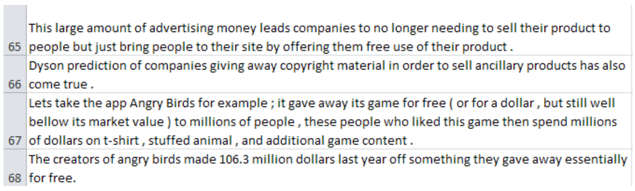
The papers in the corpus contain two topics. In the first topic, the students discussed the role that Big Data played in Obama's presidential campaign. This topic contains 11 pairs of first and final drafts of short papers. We name this **C1**. The other topic, named **C2**, talks about intellectual property and contains 10 pairs of paper drafts. The students involved in these two topics are from the same class. Students make more modifications to their papers in **C2**. More details can be seen in Table 2.

Our revision change detection approach contains three steps: sentence alignment, edit sequence generation and merge of edit sequences. Thus we annotated for these three steps.



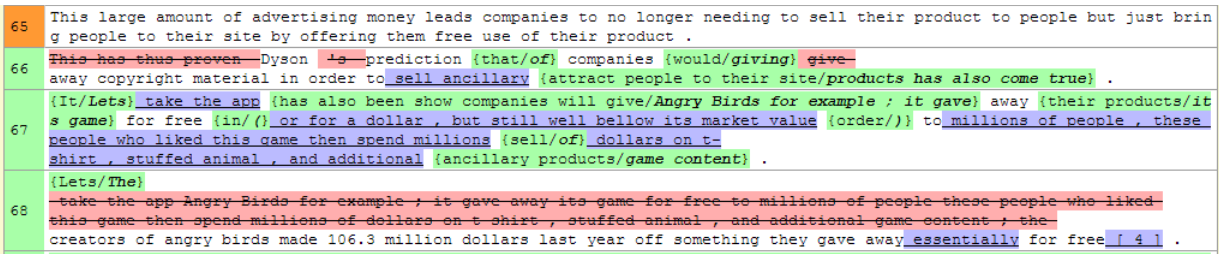
| | |
|---|---|
| 54 | This large amount of advertising money leads companies to no longer needing to sell their product to people but just bring people to their site by offering them free use of their product . |
| 55 | This has thus proven Dyson 's prediction that companies would give away copyright material in order to attract people to their site . |
| 56 | It has also been show companies will give away their products for free in order to sell ancillary products . |
| 57 | Lets take the app Angry Birds for example ; it gave away its game for free to millions of people these people who liked this game then spend millions of dollars on t-shirt , stuffed animal , and additional game content ; the creators of angry birds made 106.3 million dollars last year off something they gave away for free . |

(a) first draft



| | |
|---|---|
| 65 | This large amount of advertising money leads companies to no longer needing to sell their product to people but just bring people to their site by offering them free use of their product . |
| 66 | Dyson prediction of companies giving away copyright material in order to sell ancillary products has also come true . |
| 67 | Lets take the app Angry Birds for example ; it gave away its game for free ( or for a dollar , but still well bellow its market value ) to millions of people , these people who liked this game then spend millions of dollars on t-shirt , stuffed animal , and additional game content . |
| 68 | The creators of angry birds made 106.3 million dollars last year off something they gave away essentially for free. |

(b) final draft



| | |
|---|---|
| 65 | This large amount of advertising money leads companies to no longer needing to sell their product to people but just brin g people to their site by offering them free use of their product . |
| 66 | ~~This has thus proven~~ Dyson ~~'s~~ prediction {that/*of*} companies {would/*giving*} ~~give~~ away copyright material in order to sell ancillary {attract people to their site/*products has also come true*} . |
| 67 | {It/*Lets*} take the app {has also been show companies will give/*Angry Birds for example ; it gave*} away {their products/*its game*} for free {in/*(*} or for a dollar , but still well bellow its market value {order/*)*} to millions of people , these people who liked this game then spend millions {sell/*of*} dollars on t-shirt , stuffed animal , and additional {ancillary products/*game content*} . |
| 68 | {Lets/*The*} ~~take the app Angry Birds for example ; it gave away its game for free to millions of people these people who liked this game then spend millions of dollars on t shirt , stuffed animal , and additional game content ; the~~ creators of angry birds made 106.3 million dollars last year off something they gave away essentially for free [ 4 ] . |

(c) Revision detection using Hashemi's approach

Figure 1: Fragments of a paper in corpus **C2** discussing intellectual property, (c) is Hashemi's work, green for recognized modifications, blue for insertions and red for deletion

For sentence alignment, each sentence in the final draft is assigned the index of its aligned sentence in the original draft. If a sentence is newly added, it will be annotated as ADD. Sentence alignment is not necessarily one-to-one. It can also be one-to-many (*Consolidation*) and many-to-one (*Distribution*). Table 3 shows a fragment of the annotation for the text shown in Figure 1.

For edit sequences, the annotators do the annotation based on the initial draft. For the same fragment in Table 3, the annotated sequence is: *Keep*, *Modify*, *Delete*, *Modify*, *Add*[1].

For edit sequence merging, we further annotate *Consolidation* and *Distribution* based on the edit sequences. In our example, 66 consolidates 55 and 56, while 57 distributes to 67 and 68.

| | **pairs** | **#D1** | **#D2** | **Avg1** | **Avg2** |
|---|---|---|---|---|---|
| C1 | 11 | 761 | 791 | 22.5 | 22.7 |
| C2 | 10 | 645 | 733 | 24.7 | 24.5 |

Table 2: Detailed information of corpora. #D1 and #D2 are the number of sentences in the first and final draft, Avg1 and Avg2 are the average number of words in one sentence in the first and final draft

As a preliminary work, we only have one annotator doing all the annotations. But for the annotation of sentence alignments, we have two annotators annotating on one pair of papers. The paper contains 76 sentences, and the annotators only disagree in one sentence. The kappa is 0.794 [2], which suggests that the annotation is reliable based on our annotation scheme.

## 4 Automatic detection of revision changes

The detection of revision changes contains three parts: sentence alignment, edit sequence generation and edit sequence merging. The first two parts generate edit sequences detected at the sentence level, while the third part groups edit sequences and classifies them into different types of changes. Currently the third step only covers the identification of *Consolidation* and *Distribution*.

| Sentence Index (Final) | 65 | 66 | 67 | 68 |
|---|---|---|---|---|
| Sentence Index (First) | 54 | 55,56 | 57 | 57 |

Table 3: An example of alignment annotation

**Sentence alignment** We adapted Nelken's approach to our problem.

***Alignment based on sentence similarity***

The alignment task goes through three stages.

1. Data preparation: for each sentence in the annotated final draft, if it is not a new sentence, create a sentence pair with its aligned sentence in the

[1]66 consolidates 55, 56; while 57 distributes to 67, 68. Notice that *Consolidation* is illustrated as *Modify*, *Delete* and *Distribution* is illustrated as *Modify*, *Add*. As the annotators annotate based on the first draft, *Modify* always appears before *Add* or *Delete*

[2]We calculate the Kappa value following Macken's idea (Macken, 2010), where the aligned sentences are categorized as direct-link, while new added sentences are categorized as null-link (ADD).

first draft. The pair is considered to be an aligned pair. Also, randomly select another sentence from the first draft to make a negative sentence pair. Thus we ensure there are nearly equal numbers of positive and negative cases in the training data.

2. Training: according to the similarity metric defined, calculate the similarity of the sentence pairs. A logistic regression classifier predicting whether a sentence pair is aligned or not is trained with the similarity score as the feature. In addition to classification, the classifier is also used to provide a similarity score for global alignment.

3. Alignment: for each pair of paper drafts, construct sentence pairs using the Cartesian product of sentences in the first draft and sentences in the final. Logistic regression classifier is used to determine whether the sentence pair is aligned or not.

We added Levenshtein distance (LD) (Levenshtein, 1966) as another similarity metric in addition to Nelken's metrics. Together three similarity metrics were compared: Levenshtein Distance, Word Overlap(WO), and TF*IDF.

***Global alignment***

Sentences are likely to preserve the same order between rewritings. Thus, sentence ordering should be an important feature in sentence alignment. Nelken's work modifies the Needleman-Wunsch alignment (Needleman and Wunsch, 1970) to find the sentence alignments and goes in the following steps.

*Step1*: The logistic regression classifier previously trained assigns a probability value from 0 to 1 for each sentence pair $s(i, j)$. Use this value as the similarity score of sentence pair: $sim(i, j)$.

*Step2*: Starting from the first pair of sentences, find the best path to maximize the likelihood between sentences according to the formula $s(i, j) = max\{s(i-1, j-1) + sim(i, j), s(i-1, j) + \mathbf{sim(i, j)}, s(i, j-1) + \mathbf{sim(i, j)}\}$

*Step3*: Infer the sentence alignments by back tracing the matrix $s(i, j)$.

We found out that changing bolded parts in the formula to $s(i, j) = max\{s(i-1, j-1) + sim(i, j), s(i-1, j) + \textit{insertcost}, s(i, j-1) + \textit{deletecost}\}$ shows better performance in our problem. According to our experiment with **C1**, *insertcost* and *deletecost* are both set to 0.1 as they are found to be the most effective during practice.

**Edit sequence generation** This step is an intermediate step, which tries to generate the edit sequence based on the sentence alignment results from the previous step. The edit sequences generated would later be grouped together and classified into different types. In our current work, a rule-based method is proposed for this step.

*Step1*: The index of original document i and the index of the modified document j both start from 0. If sentence i in the original document is aligned to sentence j in the modified one, go to step 2, if not go to step 3.

*Step2*: If the two sentences are exactly the same, add *Keep* to the edit sequence, if not, add *Modify*. Increase i and j by 1, go to step 1.

*Step3*: Check the predicted alignment index of sentence j, if the predicted index is larger than sentence i in the original document, add *Delete* and increase i by 1, otherwise, mark as *Add* and increase j by 1, go to step 1.

**Edit sequence merging** *Distribution* means splitting one sentence into two or more sentences, while *Consolidation* means merging two or more sentences into one sentence. These two operations can be derived with primitives *Modify*, *Add* and *Delete*. They follow the following patterns:

***Consolidation***: *Modify-Delete-Delete-...*

***Distribution***: *Modify-Add-Add-...*

These sequences both start with *Modify* followed with a repetitive number of *Delete* or *Add*. A group of edit sequences can be merged if they can be merged to a sentence close to the sentence in the other draft. We applied a rule-based approach based on our observations.

We first scan through the sequence generated above. Sequences with *Modify-Add-...* or *Modify-Delete-...* are extracted. For each sequence extracted, if there are n consecutive *Add* or *Delete* following *Modify*, create $n$ groups, $Group_i(i \leq n)$ contains sentences from the modified sentence to the next consecutive i sentences. For each group, merge all the sentences, and use the classifier trained above to get the similarity score $Sim_{group_i}$ between the merged sentence and the original one. If there are multiple groups classified as aligned, choose group i that has the largest $Sim_{group_i}$, merge the basic edit operations into *Consolidation* or *Distribution*. If none of the groups are classified as aligned, do not merge.

## 5 Evaluation

**Sentence alignment** We use accuracy as the evaluation metric. For each pair of drafts, we count the number of sentences in the final draft

$N_1$. For each sentence in the final draft, we count the number of sentences that get the correct alignment as $N_2$. The accuracy of the sentence alignment is $\frac{N_2}{N_1}$. [3]

We use Hashemi's approach as the baseline. Compare Suite colors the differences out, as shown in Figure 1. We treat the green sentences as *Modify* and aligned to the original sentence.

For our method, we tried four groups of settings. Group 1 and group 2 perform leave-one-out cross validation on **C1** and **C2** (test on one pair of paper drafts and train on the others). Group 3 and group 4 train on one corpus and test on the other.

| Group | LD | WO | TF*IDF | Baseline |
|---|---|---|---|---|
| 1 | 0.9811 | 0.9863 | **0.9931** | 0.9427 |
| 2 | 0.9649 | 0.9593 | **0.9667** | 0.9011 |
| 3 | **0.9727** | 0.9700 | **0.9727** | 0.9045 |
| 4 | 0.9860 | **0.9886** | 0.9798 | 0.9589 |

Table 4: Accuracy of our approach vs. baseline

Table 4 shows that all our methods beat the baseline [4]. Among the three similarity metrics, TF*IDF is the most predictive.

**Edit sequence generation** We use WER (Word Error Rate) from speech recognition for evaluating the generated sequence by comparing the generated sequence to the gold standard.

WER is calculated based on edit distances between sequences. The ratio is calculated as: $WER = \frac{S+D+I}{N}$, where *S* means the number of modifications, *D* means the number of deletes, *I* means the number of inserts.

We apply our method on the gold standard of sentence alignment. The generated edit sequence is then compared with the gold standard edit sequence to calculate WER. Hashemi's approach is chosen as the baseline. The WER of our method is 0.035 on **C1** and 0.017 on **C2**, comparing to 0.091 on **C1** and 0.153 on **C2** for the baseline, which shows that our rule-based method has promise. Applying our method on the predicted alignment on the first step gets 0.067 on **C1** and 0.025 on **C2**, which although degraded still beats the baseline.

**Edit sequence merging** There are only a limited number of *Consolidation* and *Distribution* examples in our corpus. Together there are 9 *Consolidation* and 5 *Distribution* operations. In our current data, the number of sentences involved in these operations is always 2. Our rule-based method achieved 100% accuracy in the identification of these operations. It needs further work to see if this method would perform equally well in more complicated corpora.

## 6 Conclusion

This paper presents a preliminary work in the effort of describing changes across revisions at a higher level than words, motivated by a long term goal to build educational applications to support revision analysis for writing. Comparing to revision analysis based on words or phrases, our approach is able to capture higher level revision operations. We also propose algorithms to detect revision changes automatically. Experiments show that our method has a reliable performance.

Currently we are investigating applying sequence merging on the automatic generated edit sequences based on edit distances directly. Our next plan is to develop a tool for comparing drafts, and conduct user studies to have extrinsic evaluations on whether our method would provide more useful information to the user. We are also planning to do further analysis based on the revisions detected, and ultimately be able to distinguish between surface changes and text-based changes.

## Acknowledgments

We would like to thank W. Wang, W. Luo, H. Xue, and the ITSPOKE group for their helpful feedback and all the anonymous reviewers for their suggestions.

This research is supported by the Institute of Education Sciences, U.S. Department of Education, through Grant R305A120370 to the University of Pittsburgh. The opinions expressed are those of the authors and do not necessarily represent the views of the Institute or the U.S. Department of Education.

[3]Notice that we have the case that one sentence is aligned to two sentences (i.e. *Consolidation*, as sentence 66 in Table 3). In our evaluation, an alignment is considered to be correct only if the alignment covers all the sentences that should be covered. For example, if Sentence 66 in Table 3 is aligned to Sentence 55 in the first draft, it is counted as an error.

[4]For Groups 1 and 2, we calculate the accuracy of Hashemi's approach under a leave-one-out setting, each time remove one pair of document and calculate the accuracy. A significance test is also conducted, the worst metric LD in Group 1 and WO in Group 2 both beat the baseline significantly ( $p_1 = 0.025$,$p_2 = 0.017$) in two-tailed T-test.

## References

Regina Barzilay and Noemie Elhadad. 2003. Sentence alignment for monolingual comparable corpora. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 25–32. Association for Computational Linguistics.

Stefan Bott and Horacio Saggion. 2011. An unsupervised alignment algorithm for text simplification corpus construction. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, pages 20–26. Association for Computational Linguistics.

Amit Bronner and Christof Monz. 2012. User edits classification using document revision histories. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 356–366. Association for Computational Linguistics.

CompareSuite. 2014. Compare suite, feature-rich file and folder compare tool. `http://www.comparesuite.com`.

Ulla Connor and Karen Asenavage. 1994. Peer response groups in esl writing classes: How much impact on revision? *Journal of Second Language Writing*, 3(3):257–276.

Lester Faigley and Stephen Witte. 1981. Analyzing revision. *College composition and communication*, pages 400–414.

Homa B. Hashemi and Christian D. Schunn. 2014. A tool for summarizing students' shanges across drafts. In *International Conference on Intelligent Tutoring Systems(ITS)*.

Michael Heilman and Noah A Smith. 2010. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 1011–1019. Association for Computational Linguistics.

Hongyan Jing. 2002. Using hidden markov modeling to decompose human-written summaries. *Computational linguistics*, 28(4):527–543.

Vladimir I Levenshtein. 1966. Binary codes capable of correcting deletions, insertions and reversals. In *Soviet physics doklady*, volume 10, page 707.

Lieve Macken. 2010. An annotation scheme and gold standard for dutch-english word alignment. In *7th conference on International Language Resources and Evaluation (LREC 2010)*, pages 3369–3374. European Language Resources Association (ELRA).

Saul B Needleman and Christian D Wunsch. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3):443–453.

Rani Nelken and Stuart M Shieber. 2006. Towards robust context-sensitive sentence alignment for monolingual corpora. In *EACL*.

Alla Rozovskaya and Dan Roth. 2010. Annotating esl errors: Challenges and rewards. In *Proceedings of the NAACL HLT 2010 fifth workshop on innovative use of NLP for building educational applications*, pages 28–36. Association for Computational Linguistics.

Chao Shen, Tao Li, and Chris HQ Ding. 2011. Integrating clustering and multi-document summarization by bi-mixture probabilistic latent semantic analysis (plsa) with sentence bases. In *AAAI*.

Ben Swanson and Elif Yamangil. 2012. Correction detection and error type selection as an esl educational aid. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 357–361. Association for Computational Linguistics.

Dingding Wang, Shenghuo Zhu, Tao Li, and Yihong Gong. 2009. Multi-document summarization using sentence-based topic models. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 297–300. Association for Computational Linguistics.

Jaclyn M. Wells, Morgan Sousa, Mia Martini, and Allen Brizee. 2013. Steps for revising your paper. `http://owl.english.purdue.edu/owl/resource/561/05`.

Huichao Xue and Rebecca Hwa. 2014. Improved correction detection in revised esl sentences. In *Proceedings of The 52nd Annual Meeting of the Association for Computational Linguistics(ACL)*.