

Sentence-level Rewriting Detection

Fan Zhang¹, Diane Litman^{1,2} {zhangfan, litman}@cs.pitt.edu

¹Department of Computer Science, University of Pittsburgh | ²Learning Research and Development Center, University of Pittsburgh



DEPARTMENT OF
COMPUTER SCIENCE

LRDC Learning Research & Development Center

1. Our goal

- Current goal: Help to have a better understanding of the rewriting process
- Ultimate goal: Provide automatic revision suggestions

2. Why do we need sentence-level rewriting detection?

Example: Text fragments from the original document

Text fragments from revised document, notice the yellow and green boxes represent the sections that have been revised

• Revision detection at word-level

Result from Hashemi's work^[1], where a word-level based text compare tool(CompareSuite) is applied, green for recognized modifications, blue for insertions and red for deletion

✗ Overwhelming information of changes

✗ Likely to make mistakes when detecting changes based on word-level edit distances

• Our approach: revision detection at sentence-level in 3 steps

➤ **Sentence alignment:** align sentences of the revised document to the ones of the original document

- Allows many-to-one and one-to-many alignments
- For the example above:

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

*Line 65 aligned to Line 54, 66 aligned to 55,56, and 67, 68 are aligned to 57

➤ **Edit sequence generation:** generate the edit sequence from the original document to the revised

- 4 basic primitives: Add, Delete, Modify, Keep
- For the example above:

Sentence Index (First)	54	55	56	57
Edit Operations	Keep	Modify	Delete	Modify, Add

*Line 54 is not modified, line 55 is tagged as modified to 66 while line 56 got deleted, line 57 considered to be modified to line 67 and then add new line 68

➤ **Edit sequence merging:** merge the basic primitives into more meaningful advanced edit primitives

- Advanced edit primitives: permutation, distribution, consolidation
- For the example above:

Sentence Index (First)	54	55,56	57
Edit Operations	Keep	Consolidation	Distribution

*Operation on line 55,56 is actually merging them together to line 66, and line 57 is split to line 67 and line 68

3. Our work

• Data preparation

- 2 undergraduate paper assignments from a "Social Implications of Computing Technology" course
- Collected via a web-based peer review system^[2], each paper has two drafts

	# of pairs	# of sentences in the first draft	# of sentences in the second draft	Average words in one sentence in C1	Average words in one sentence in C2
Corpus1	11	761	791	22.5	22.7
Corpus2	10	645	733	24.7	24.5

• Manual annotation

- Sentence alignment: two annotators annotate on one paper, kappa: 0.794
- Edit sequence generation: annotate edit sequence from the first draft
- Edit sequence merging: annotates "consolidation", "permutation" currently

• Automatic sentence-level revision detection in 3 steps

➤ Sentence alignment

❑ Method: adapting Nelken's approach^[3]

- Logistic regression classifier using sentence similarity score (Word Overlap, TF*IDF, Levenshtein Distance)
- Global alignment based on sentence order (Needleman-Wunsch^[4])
- Evaluation: accuracy (percentage of sentences that are correctly aligned)

❑ Performance

- Baseline: Hashemi's word-based approach (as in section 1), performance collected by manual inspection

Group	Levenshtein Distance	Word Overlap	TF*IDF	Baseline
Cross validation on corpus 1	0.9811	0.9863	0.9931	0.9427
Cross validation on corpus 2	0.9649	0.9593	0.9667	0.9011
Train on corpus 1 and test on corpus 2	0.9727	0.9700	0.9727	0.9045
Train on corpus 2 and test on corpus 1	0.9860	0.9886	0.9798	0.9589

*All the algorithms achieved high accuracy, TF*IDF achieves the best performance among all similarity metrics

➤ Edit sequence generation

❑ Method: Rule-based approach

❑ Evaluation: Word error rate (WER), rate of segments to be modified to match with the correct sequence

❑ Performance

- Baseline: Hashemi's word-based approach (as in section 1), performance collected by manual inspection

	Corpus1	Corpus2
Baseline	0.091	0.153
Rule-based method	0.035	0.017
Rule-based on alignment results	0.067	0.025

* Rule-base approach achieved a much better performance of edit sequences comparing to baseline, the result is still better than the baseline even applied on the automatic aligned results from the first step

➤ Edit sequence merging

❑ Method: Rule-based approach, now only recognizes "Distribution" and "Consolidation"

❑ Evaluation: accuracy (percentage of the "Distribution" and "Consolidation" cases recognized)

❑ Performance: The 9 consolidation and 5 distribution cases are all successfully recognized (100% accuracy)

4. Future work

- Improve the accuracy of current algorithm
- Replace rule-based approach used in edit sequence generation phase with approach based on edit distance, and then infer advanced edits based on the automatic generated sequence
- Identify more meaningful advanced rewriting operations
- Conduct user study comparing the utility of sentence versus word-level rewriting detection

References

- [1] Homa B. Hashemi and Christian D. Schunn. 2014. A tool for summarizing students' changes across drafts. In *International Conference on Intelligent Tutoring Systems (ITS)*
- [2] Cho K, Schunn C D. Scaffolded writing and rewriting in the discipline: A web-based reciprocal peer review system[J]. *Computers & Education*, 2007, 48(3): 409-426.
- [3] Rani Nelken and Studart M Shieber. 2006. Towards robust context-sensitive sentence alignment for monolingual corpora. In *EACL*
- [4] 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