A Customizable Editor for Text Simplification

John Lee, Wenlong Zhao
Dept. of Linguistics and Translation
City University of Hong Kong
Hong Kong SAR, China
jsylee@cityu.edu.hk
wenlzhao@gmail.com

Wenxiu Xie
Cisco School of Informatics
Guangdong University of Foreign Studies
Guangzhou, China
vasiliky@outlook.com

Abstract

We present a browser-based editor for simplifying English text. Given an input sentence, the editor performs both syntactic and lexical simplification. It splits a complex sentence into shorter ones, and suggests word substitutions in drop-down lists. The user can choose the best substitution from the list, undo any inappropriate splitting, and further edit the sentence as necessary. A significant novelty is that the system accepts a customized vocabulary list for a target reader population. It identifies all words in the text that do not belong to the list, and attempts to substitute them with words from the list, thus producing a text tailored for the targeted readers.

1 Introduction

The task of *text simplification* aims to rewrite a sentence so as to reduce its lexical and syntactic complexity, while preserving its meaning and grammaticality. Consider the complex sentence "The professor, carrying numerous books, entered the room." It can be rewritten into two simple sentences, "The teacher entered the room." and "He was carrying many books." The rewriting process involves both syntactic and lexical simplification. The former decomposes the complex sentence, extracting the participial phrase "carrying numerous books" and turning it into a separate sentence. The latter replaces the word "professor" with the simpler word "teacher", and "numerous" with "many".

It is well known that sentences with difficult vocabulary, passive voice or complex structures, such as relative and subordinated clauses, can be challenging to understand. Text simplification has been found to be beneficial for language learners (Shirzadi, 2014), children (Kajiwara et al., 2013), and adults with low literacy skills (Arnaldo Candido Jr. and Erick Maziero and Caroline Gasperin and Thiago A. S. Pardo and Lucia Specia and Sandra M. Aluisio, 2009) or language disabilities (John Carroll and Guido Minnen and Darren Pearce and Yvonne Canning and Siobhan Devlin and John Tait, 1999; Luz Rello and Ricardo Baeza-Yates, 2014). To cater to these target reader populations, language teachers, linguists and other editors are often called upon to manually adapt a text. To automate this time-consuming task, there has been much effort in developing systems for lexical simplification (Zhu et al., 2010; Biran et al., 2011) and syntactic simplification (Siddharthan, 2002; Siddharthan and Angrosh, 2014).

The performance of the state-of-the-art systems has improved significantly (Horn et al., 2014; Siddharthan and Angrosh, 2014). Nonetheless, one cannot expect any single system, trained on a particular dataset, to simplify arbitrary texts in a way that would suit all readers — for example, the kinds of English words and structures suitable for a native speaker in Grade 6 are unlikely to be suitable for a non-native speaker in Grade 4. Hence, human effort is generally needed for modifying the system output.

To support human post-editing, a number of researchers have developed specialized editors for text simplification. While the editor described in Max (2006) shares similar goals as ours, it requires human intervention in much of the simplification process. The Automatic Text Adaptation tool suggests synonyms (Burstein et al., 2007), but does not perform syntactic simplification. Conversely, the *Simpli-*

This work is licensed under a Creative Commons Attribution 4.0 International License. License details: http://creativecommons.org/licenses/by/4.0/

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations, pages 93–97, Osaka, Japan, December 11-17 2016.

fica tool, developed for Brazilian Portuguese, does not perform lexical simplification. Other packages for lexical simplification, such as LEXenstein (Paetzold and Specia, 2015), are not designed for post-editing.

To fill this gap, we developed a customizable, browser-based editor for simplifying English text. Besides performing automatic lexical and syntactic simplification, it facilitates user post-editing, for example in choosing candidate substitutions or undoing sentence splits. Importantly, the user can supply a vocabulary list tailored for a target reader population. This list serves to specify which words are considered "simple," thus guiding the system in tailoring lexical substitution for the target readers.

2 Lexical Simplification

The lexical simplification task generally consists of three steps (Paetzold and Specia, 2015). The first step, substitution generation, produces a list of candidate words to substitute for the target word w. Typically, the context of w in the input sentence is not considered in this step. In the second step, substitution selection, the system selects the best candidates to replace w in the input sentence. Finally, the substitution ranking step re-ranks the candidates in terms of their simplicity.

Often, the expected vocabulary level of a target reader population is explicitly prescribed. For example, many governments have drawn up graded vocabulary lists to guide students of English as a foreign language; likewise, developers of machine translation systems have specified controlled languages with restricted vocabulary. In this context, lexical simplification can be defined as follows: to rewrite a sentence by replacing all words that are not in the given vocabulary list (and hence presumed to be difficult for the reader) with those from the list (and hence presumed to be simple). For example, Kajiwara et al. (2013) performed lexical simplification based on 5,404 words that elementary school children are expected to know.

2.1 Algorithm

By default, the editor uses a list of approximately 4,000 words that all students in Hong Kong are expected to know upon graduation from primary school (EDB, 2012). However, the user can also upload his or her own vocabulary list. Given an input sentence, we first identify the target words, namely those words that do not appear in the vocabulary list. Following Horn et al. (2014), our system simplifies neither proper nouns, as identified by the Natural Language Toolkit (Bird et al., 2009), nor words in our stoplist, which are already simple. In terms of the three-step framework described above, we use the word2vec model[1] to retrieve candidates for substitution in the first step. We trained the model with all sentences from Wikipedia. For each target word, the model returns a list of the most similar words; we extract the top 20 in this list that are included in the user-supplied vocabulary list. In the next step, substitution selection, we re-rank these 20 words with a language model. We trained a trigram model with the kenlm (Heafield, 2011), again using all sentences from Wikipedia. We then place the 10 words with the highest probabilities in a drop-down list in our editor[2]; for example, Figure 1 shows the ten candidates offered for the word "municipal". If none of the candidates are appropriate, the user can easily revert to the original word, which is also included in the drop-down list; alternatively, the user can click on the text to directly edit it.

2.2 Evaluation

We evaluated the performance of our algorithm on the Mechanical Turk Lexical Simplification Data Set (Horn et al., 2014). This dataset contains 500 manually annotated sentences; the target word in each sentence was annotated by 50 independent annotators. To simulate a teacher adapting an English text for Hong Kong pupils, we used the vocabulary list from the Hong Kong Education Bureau (EDB, 2012). To enable automatic evaluation, we considered only the 249 sentences in the dataset whose target word is not in our vocabulary list, but whose human annotations contain at least one word in the list. Precision is at 31% for the top candidate; it is at 57% for the top ten candidates. In other words, for 57% of the target words, a valid substitution can be found in the drop-down list in the editor.

[1] http://code.google.com/archive/p/word2vec/

[2] We regard all words in the vocabulary list to be sufficiently simple, and do not perform the third step, substitution ranking.

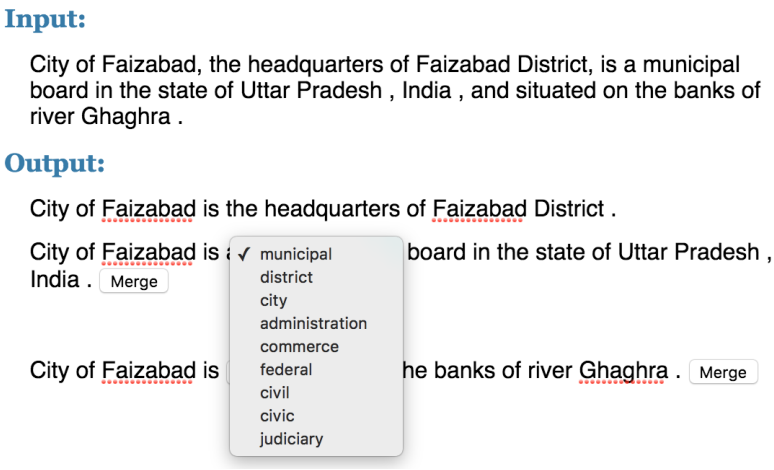

Figure 1: The input sentence is "City of Faizabad, the headquarters of Faizabad District, is a municipal board in the state of Uttar Pradesh, India, and situated on the banks of river Ghaghra." For syntactic simplification (Section 3), the system first splits its coordinated clauses into two sentences, S_1="City of Faizabad ... state of Uttar Pradesh, India."; and S_2="City of Faizabad is situated on the banks of river Ghaghra". It then further extracts the appositive phrase "the headquarters of Faizabad District" from S_1, and turns into a separate sentence. For lexical simplification (Section 2), the system offers eight substitution candidates for the word "municipal" in a drop-down list.

3 Syntactic Simplification

The editor performs automatic syntactic simplification for seven grammatical constructs. In a complex sentence, it identifies relative clauses, adverbial clauses, coordinated clauses, subordinated clauses, participial phrases and appositive phrases; it then splits the sentence into two simpler ones. Further, it transforms passive voice into active voice when the agent is explicitly mentioned. Examples of these constructs and their simplifications are listed in Table 1.

3.1 Algorithm

The system follows the three-step framework of analysis, transformation and regeneration, as laid out in Siddharthan (2002). In the analysis step, it parses the input sentence with the Stanford dependency parser (Manning et al., 2014). In the transformation step, it scans the parse tree of the input sentence to match subtree patterns that have been manually crafted for each of the seven constructs in Table 1. In Figure 1, the input sentence matches the subtree pattern for coordination; it is therefore split into two shorter sentences, S_1="City of Faizabad ... India." and S_2="and situated ... river Ghaghra". Since S_1 then matches the pattern for appositive phrase, the phrase "the headquarters of Faizabad District" is taken out to form its own sentence. If the user finds a sentence split to be inappropriate, he or she can click on the "Merge" button to undo the split. Finally, in the regeneration step, the editor restores the subject (e.g., "City of Faizabad") to newly formed sentences. Often, this step also requires generation of referring expressions, determiners, conjunctions and sentence re-ordering. Since most of these tasks require real-world knowledge, the editor currently leaves it to the user for post-editing.

3.2 Evaluation

We evaluated the quality of syntactic simplification on the first 300 sentences in the Mechanical Turk Lexical Simplification Data Set (Horn et al., 2014). For each sentence, we asked a professor of linguistics to mark the types of syntactic simplification (Table 1) that are applicable, without regard to regeneration requirements. Compared with this human gold standard, the system achieved 79% precision and 64% recall.

Type	Example
Coordination	"I ate an apple and he ate an orange." → "I ate an apple. He ate an orange."
Subordination	"Since he was late, I left." → "He was late. So, I left."
Adverbial clauses	"Impatient, he stood up." → "He was impatient. He stood up."
Participial phrases	"Peter, sweating hard, arrived." → "Peter arrived. He was sweating hard."
Relative clauses	"Peter, who liked fruits, ate an apple" → "Peter liked fruits. He ate an apple."
Appositive phrases	"Peter, my friend, ate an apple" → "Peter was my friend. He ate an apple."
Passive voice	"An apple was eaten by Peter" → "Peter ate an apple."

Table 1: Types of syntactic simplification supported by the editor.

4 Conclusions and Future Work

We have presented a browser-based editor that performs lexical and syntactic simplification and supports human post-editing. The editor takes a customized vocabulary list as input, such that its lexical substitutions are tailored to the needs of the target reader population. Evaluation shows that, for a majority of sentences in a test set, the editor is able to propose appropriate word substitutions and to split up complex syntactic structures. In future work, we aim to further improve the quality of simplification, and to offer annotations for difficult words that cannot be simplified.[3] We also intend to perform empirical studies, to measure the editor's effectiveness in assisting teachers in language lesson planning.

Acknowledgements

This work was supported by the Innovation and Technology Fund (Ref: ITS/132/15) of the Innovation and Technology Commission, the Government of the Hong Kong Special Administrative Region.

References

Arnaldo Candido Jr. and Erick Maziero and Caroline Gasperin and Thiago A. S. Pardo and Lucia Specia and Sandra M. Aluisio. 2009. Supporting the Adaptation of Texts for Poor Literacy Readers: a Text Simplification Editor for Brazilian Portuguese. In *Proc. NAACL-HLT Workshop on Innovative Use of NLP for Building Educational Applications*.

Or Biran, Samuel Brody, and Noémie Elhadad. 2011. Putting it Simply: a Context-aware Approach to Lexical Simplification. In *Proc. NAACL-HLT*.

Steven Bird, Edward Loper, and Ewan Klein. 2009. *Natural Language Processing with Python*. O'Reilly Media Inc.

Jill Burstein, Jane Shore, John Sabatini, Yong-Won Lee, and Matthew Ventura. 2007. The Automated Text Adaptation Tool. In *Proc. NAACL-HLT Demonstration Program*.

EDB. 2012. *Enhancing English Vocabulary Learning and Teaching at Secondary Level*. http://www.edb.gov.hk/vocab_learning_sec.

Kenneth Heafield. 2011. KenLM: Faster and Smaller Language Model Queries. In *Proc. 6th Workshop on Statistical Machine Translation*.

Colby Horn, Katie Manduca, and David Kauchak. 2014. Learning a Lexical Simplifier Using Wikipedia. In *Proc. ACL*.

John Carroll and Guido Minnen and Darren Pearce and Yvonne Canning and Siobhan Devlin and John Tait. 1999. Simplifying Text for Language-Impaired Readers. In *Proc. EACL*.

Tomoyuki Kajiwara, Hiroshi Matsumoto, and Kazuhide Yamamoto. 2013. Selecting Proper Lexical Paraphrase for Children. In *Proc. 25th Conference on Computational Linguistics and Speech Processing (ROCLING)*.

Luz Rello and Ricardo Baeza-Yates. 2014. Evaluation of DysWebxia: A Reading App Designed for People with Dyslexia. In *Proc. W4A*, Seoul, South Korea.

[3]For example, Burstein et al. (2007) used marginal notes.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Proc. ACL System Demonstrations*, pages 55–60.

Aurélien Max. 2006. Writing for Language-Impaired Readers. In *Proc. CICLing*.

Gustavo Paetzold and Lucia Specia. 2015. LEXenstein: A Framework for Lexical Simplification. In *Proc. ACL-IJCNLP System Demonstrations*.

Solmaz Shirzadi. 2014. Syntactic and Lexical simplification: the Impact on EFL Listening Comprehension at Low and High Language Proficiency Levels. *Journal of Language Teaching and Research*, 5(3):566–571.

Advaith Siddharthan and M. A. Angrosh. 2014. Hybrid Text Simplification Using Synchronous Dependency Grammars with Hand-Written and Automatically Harvested Rules. In *Proc. EACL*.

Advaith Siddharthan. 2002. An Architecture for a Text Simplification System. In *Proc. Language Engineering Conference (LEC)*.

Zhemin Zhu, Delphine Bernhard, and Iryna Gurevych. 2010. A Monolingual Tree-based Translation Model for Sentence Simplification. In *Proc. COLING*.