Splitting Complex English Sentences

John Lee
Department of Linguistics and Translation
City University of Hong Kong
jsylee@cityu.edu.hk

J. Buddhika K. Pathirage Don*
Hong Kong Applied Science
and Technology Research Institute
bpathiragedon@astri.org

Abstract

This paper applies parsing technology to the task of syntactic simplification of English sentences, focusing on the identification of text spans that can be removed from a complex sentence. We report the most comprehensive evaluation to-date on this task, using a dataset of sentences that exhibit simplification based on coordination, subordination, punctuation/parataxis, adjectival clauses, participial phrases, and appositive phrases. We train a decision tree with features derived from text span length, POS tags and dependency relations, and show that it significantly outperforms a parser-only baseline.

1 Introduction

The task of text simplification aims to rewrite a sentence so as to reduce its complexity, while preserving its meaning and grammaticality. The rewriting may involve various aspects, including lexical simplification, syntactic simplification, content deletion, and content insertion for clarification. This paper focuses on syntactic simplification and, specifically, on splitting a complex sentence into two simpler sentences.[1] Consider the input sentence S in Table 1, a complex sentence containing a participial phrase, "carrying numerous books". After removing this phrase from S, the system generates S_2 from the phrase by turning the participle "carrying" into the finite form "was carrying", and by generating the pronoun "he" as the subject.

A number of systems can already perform this task (Chandrasekar et al., 1996; Siddharthan, 2002a; Inui et al., 2003; Belder and Moens, 2010; Bott et al., 2012; Siddharthan and Angrosh, 2014; Saggion et al., 2015). While some systems have undergone task-based evaluations, such as reading comprehension (Angrosh et al., 2014), most have adopted holistic assessment, which commonly includes human ratings on the grammaticality, fluency, meaning preservation, and simplicity of the system output (Štajner et al., 2016). These ratings are indeed helpful in indicating the overall quality of a system; however, the need for human intervention restricts the scale of the evaluations, and makes direct comparisons difficult. Other systems have been evaluated with automatic metrics, such as BLEU and readability metrics (Aluisio et al., 2010; Narayan and Gardent, 2014), which overcome the limitations of human ratings, but do not reveal what aspects of the simplification process caused the most difficulties.

S	The man, carrying numerous books, entered the room.
S_1	The man entered the room.
S_2	He was carrying numerous books.

Table 1: Example input (S) and output sentences (S_1, S_2) for the task of syntactic simplification.

The contribution of this paper is two-fold. First, it presents the first publicly available dataset that facilitates detailed, automatic evaluation on syntactic simplification. Second, we report the results of a decision tree approach that incorporates parse features, giving a detailed analysis on its performance for various constructs.

2 Previous Work

The phrase-based and syntax-based machine translation approaches have been used in many text simplification systems (Vickrey and Koller,

*The second author completed this work as a Postdoctoral Fellow at City University of Hong Kong.

[1] The simplified sentences can in turn be split iteratively.

Proceedings of the 15th International Conference on Parsing Technologies, pages 50–55,
Pisa, Italy; September 20–22, 2017.

Construct	Complex sentence example	Freq.
Coordination	I ate fish *and he drank wine*.	463
	Peter came *and saw*.	
Adjectival clause	Peter ate an apple, *which was green*.	119
Participial phrase	Peter, *running fast*, arrived.	90
Appositive phrase	Peter, *my friend*, ate the apple.	69
Subordination	*After Peter came*, I left.	158
	After calling Peter, I left.	
Punctuation/Parataxis	I ate fish; *he drank wine*.	155
	Peter (*twenty years old*) is a linguistics major.	
Prepositional phrase	The final was played at Manchester *on 14 May 2008*.	16

Table 2: Distribution of syntactic constructs in the test set (Section 3). The complex sentence can be split into two simpler sentences by (i) removing the text span (italicized); and (ii) transforming the text span into a new sentence with the referent (underlined). This paper focuses on step (i).

2008; Zhu et al., 2010; Coster and Kauchak, 2011; Wubben et al., 2012). While these approaches are effective for lexical substitution and deletion, they are less so for sentence splitting, sentence reordering, and morphological changes (Bott et al., 2012; Siddharthan, 2014).

Most syntactic simplification systems start by analyzing the input sentence via a parse tree, or a deep semantic representation (Narayan and Gardent, 2014). For identifying the referent NP (clause attachment), accuracy can reach 95%; for identifying clause boundary, accuracy is at 97% when there is punctuation, and 80% in general (Siddharthan, 2002b). Noun post-modifiers can be extracted at an F-measure of 92% (Dornescu et al., 2014; Stanovsky and Dagan, 2016).

Given a syntactic analysis of the input sentence, the system then applies manually written transformation rules (Siddharthan, 2002a; Belder and Moens, 2010; Bott et al., 2012; Siddharthan and Angrosh, 2014; Saggion et al., 2015). These rules identify specific constructs in a parse tree, such as the participial phrase in S in Table 1; they then determine whether the construct should be split, and if so, generate an independent sentence from it. For example, Aluísio et al. (2008) used a set of transformation rules to treat 22 syntactic constructs. Siddharthan (2011) used 63 rules, which handle coordination of both verb phrases and full clauses, subordination, apposition and relative clauses, as well as conversion of passive voice to active voice. Siddharthan and Angorsh (2014) used 26 hand-crafted rules for apposition, relative clauses and combinations of the two; as well as 85 rules handle subordination and coordination.

3 Data

Many evaluation datasets are available for lexical simplification (Paetzold and Specia, 2016), but there is not yet any that enables automatic evaluation on syntactic simplification systems. We created an annotated dataset for this purpose, based on the 167,689 pairs of "normal" and simplified sentences from Wikipedia and Simple Wikipedia aligned by Kauchak (2013). While a majority of these pairs are one-to-one alignments, 23,715 of them are one-to-two alignments.[2] These aligned sentences, in their raw form, can serve as triplets of S, S_1 and S_2 (Table 1).

However, as pointed out by Xu et al. (2015), Wikipedia and Simple Wikipedia contain rather noisy data; indeed, upon manual inspection, not all triplets are good examples for syntactic simplification. These fall into two cases. In the first case, significant content from S is deleted and appear neither in S_1 nor S_2; these triplets provide examples of content deletion rather than splitting. In the second case, S_2 (or S_1) consists mostly of new content. In some instances, S_1 (or S_2) is so similar to S that no real splitting occurs. In others, the new content put into doubt whether the splitting of S was motivated by syntactic complexity alone, or were influenced by the new content. To reduce the noise, we employed human annotation to create the test set, and an automatic procedure to clean up the training set.

[2]There is no one-to-n alignments for $n > 2$.

<S><ref>The man</ref> <split type="participial">, carrying numerous books,</split> entered the room.</S> <S1><ref1>The man</ref1> entered the room.</S1> <S2><ref2>He</ref2> was carrying numerous books.</S2>

Table 3: Each triplet in our corpus contains a complex sentence (S) and the two shorter sentences (S_1, S_2) into which it was re-written.

3.1 Test set

An annotator marked up 1,070 triplets of (S, S_1, S_2) in the format shown in Table 3, with the following items:[3]

Removed text span The <split> element encloses the text span that is removed from S. This text span usually, though not necessarily, appears in S_1 or S_2.

Construct Type The type attribute inside the <split> element indicates the construct type of the removed text span. Table 2 gives a list of the construct types and their distribution.

Re-ordering If the removed text span forms the basis of S_1 (S_2), the dest attribute inside the <split> element takes the value S1 (S2). This attribute indicates whether sentence re-ordering has occurred.

Referent There are often referring expressions in S_1 and S_2 for an entity in S. For example, in Table 1, the words "the man" and "he" in S_1 and S_2 refer to "the man" in S. These referring expressions are marked as <ref1> and <ref2>, and the entity in S is marked as <ref>.

3.2 Training set

The rest of the triplets form our training instances. To filter out instances that are not genuine splits (see above) and to determine the value of dest, we require at least 75% of the words in either S_1 or S_2 to appear in S. To determine the value of type, we ran the baseline system (Section 4), which is unsupervised and has relatively high recall, on S. Thus, the training set has all the annotations in Table 3, except for <ref>, <ref1> and <ref2>.

[3]The annotations on re-ordering and referent were not used in this study, but will be useful for evaluation on sentence re-generation.

4 Approach

Baseline system. We manually developed tree patterns, in the form of dependency relations and POS tags, to identify text spans that should be removed from a complex sentence (Table 4). These patterns are designed to yield high recall but lower precision. The system parses the input sentence with the Stanford parser (Manning et al., 2014), and then performs breadth-first search on the dependency tree for these patterns, returning the first match it finds. This algorithm removes at most one text span from each complex sentence; this assumption is consistent with the material in our dataset, which was derived from one-to-two sentence alignments.

Proposed system. Even if one of the constructs in Table 2 is present in a complex sentence, it may not be appropriate or worthwhile to remove it. To refine the tree patterns developed for the baseline system, we trained a decision tree with the scikit-learn package. For each word in the sentence, the decision tree considers the features listed in Table 5. If the decision tree predicts a split, the text span headed by the word is removed from the sentence.

5 Evaluation

We evaluate our system's performance at identifying a text span, if any, in a complex sentence that should be removed to form an independent sentence.

As expected, the baseline system achieved relatively high recall (0.88) but low precision (0.34), since it always tries to split a text span that matches any of the tree patterns in Table 4. The decision tree was able to substantially increase the precision (0.63) by learning some useful rules, at the expense of lower recall (0.72).

Some rules that substantially contributed to the performance gain are as follows. Consider the rule that a comma should separate the word from its parent when the dependency relation is xcomp. It was able to block the system from mistakenly tak-

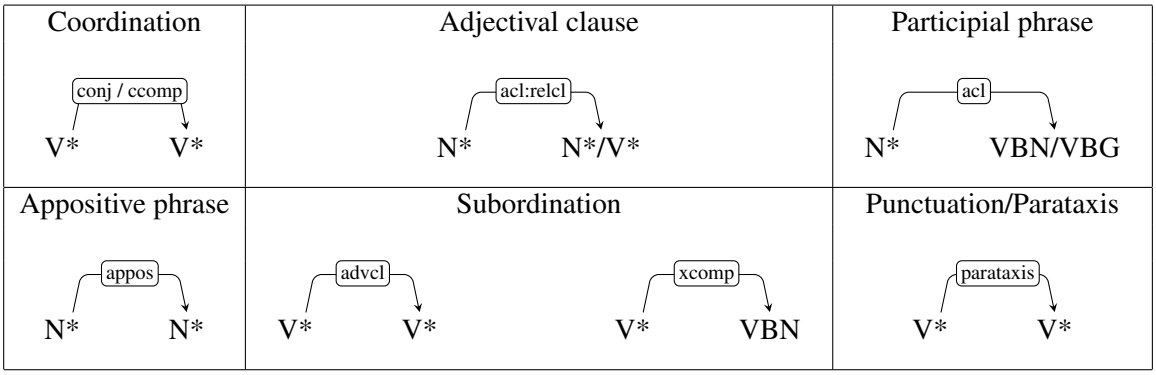

Table 4: Manually crafted tree patterns, written in Stanford dependencies (Manning et al., 2014), that are used in the baseline system. If the pattern exists in S, the text span headed by the child word (e.g., VBN/VBG for participial phrases) is to be removed from S.

Feature	**Description**
POS	Tag is JJ, N*, VBN, VBG, or V*?
Parent POS	Tag of parent word is JJ, N*, VBN, VBG, or V*?
Root	Parent word is root?
Sibling POS	Tag of a sibling word is IN, TO, WP, WRB, WP$, or WDT?
Child POS	Tag of child word is IN, TO, WP, WRB, WP$, or WDT?
Comma	There is a comma between the word and its parent word?
Det	Subject of S is tagged DT?
Length	Number of words in text span

Table 5: Features used by the decision tree.

System → ↓ **Construct**	Baseline P/R/F	Proposed P/R/F
Coordination	0.31/0.84/0.45	0.61/0.80/0.69
Adjectival clause	0.29/0.97/0.45	0.59/0.79/0.68
Participial phrase	0.33/0.90/0.48	0.56/0.58/0.57
Appositive phrase	0.21/0.91/0.34	0.36/0.56/0.44
Subordination	0.39/0.84/0.53	0.70/0.74/0.72
Punctuation/ Parataxis	0.78/0.99/0.87	0.92/0.95/0.93
Overall	0.34/0.88/0.49	**0.63/0.72/0.67**

Table 6: Precision, recall and F-measure for identifying the text span to be removed from S.

ing the phrase "conducting at Montreux ..." out of the sentence "He began conducting at Montreux ...". Another useful rule was that the parent word in the `conj` relation must be root; otherwise, the structure is likely coordinated NPs rather than coordinated clauses. Further, when the modifier is tagged as TO (i.e., an infinitive), or when the subject of the sentence is a determiner (e.g., "this", "that"), no splitting should be done. Finally, shorter text spans are less likely to be split up.

Among the different constructs, the proposed system performed best for punctuation/parataxis, with precision at 0.92 and recall at 0.95. This construct is not only clearly marked, but also more consistently split up. The most challenging construct turned out to be appositive phrases, with precision at 0.36 and recall at 0.56. Many of the errors trickled down from inaccurate analysis by the automatic parser, especially mistakes in relative clause attachment and clause boundary identification (Siddharthan and Angrosh, 2014).

The precision figures can be viewed as lower bounds. In post-hoc analysis, we found that many of the proposed text spans by our system can be acceptable, but they were not deemed necessary to be split up by the editors of Simple Wikipedia. Ultimately, the decision to split a complex sentence should be made in consideration of the reader's proficiency, but our current dataset does not support the modelling of this factor.

6 Conclusions and Future Work

We have presented a study on syntactic simplification, focusing on the identification of text spans that should be removed from a complex sentence.

We trained a decision tree to learn to recognize these text spans, using dependencies, POS tags and text span length as features. Experimental results showed that it outperformed a parser-only baseline.

We have reported the most detailed evaluation to-date on this task. This evaluation was made possible with a new dataset, derived from Wikipedia and Simple Wikipedia, that covers coordinated clauses, subordinated clauses, punctuation/parataxis, adjectival clauses, participial clauses, appositive phrases, and prepositional phrases.

In future work, we plan to investigate the next steps in syntactic simplification, i.e., sentence reordering and the generation of referring expressions. Our dataset, which traces sentence order and annotates referring expressions, is well suited for automatic evaluation for these tasks.

Acknowledgments

This work was partially 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

Sandra Aluisio, Lucia Specia, Caroline Gasperin, and Carolina Scarton. 2010. Readability Assessment for Text Simplification. In *Proc. 5th Workshop on Innovative Use of NLP for Building Educational Applications*. pages 1–9.

Sandra Aluísio, Lucia Specia, T. A. Pardo, E. G. Maziero, and R. P. Fortes. 2008. Towards Brazilian Portuguese Automatic Text Simplification Systems. In *Proc. 8th ACM Symposium on Document Engineering*.

M. A. Angrosh, Tadashi Nomoto, and Advaith Siddharthan. 2014. Lexico-syntactic text simplification and compression with typed dependencies. In *Proc. COLING*.

J. De Belder and M. F. Moens. 2010. Text Simplification for Children. In *Proc. SIGIR Workshop on Accessible Search Systems*.

Stefan Bott, Horacio Saggion, and David Figueroa. 2012. A Hybrid System for Spanish Text Simplification. In *Proc.Workshop on Speech and Language Processing for Assistive Technologies*.

R. Chandrasekar, Christine Doran, and B. Srinivas. 1996. Motivations and Methods for Text Simplification. In *Proc. COLING*.

William Coster and David Kauchak. 2011. Learning to Simplify Sentences using Wikipedia. In *Proc. Workshop on Monolingual Text-to-Text Generation*.

Iustin Dornescu, Richard Evans, and Constantin Orăsan. 2014. Relative Clause Extraction for Syntactic Simplification. In *Proc. Workshop on Automatic Text Simplification: Methods and Applications in the Multilingual Society*. Dublin, Ireland.

Kentaro Inui, Atsushi Fujita, Tetsuro Takahashi, Ryu Iida, and Tomoya Iwakura. 2003. Text Simplification for Reading Assistance: A Project Note. In *Proc. 2nd International Workshop on Paraphrasing*.

David Kauchak. 2013. Improving Text Simplification Language Modeling using Unsimplified Text Data. In *Proc. ACL*.

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.

Shashi Narayan and Claire Gardent. 2014. Hybrid Simplification using Deep Semantics and Machine Translation. In *Proc. ACL*.

Gustavo H. Paetzold and Lucia Specia. 2016. Benchmarking Lexical Simplification Systems. In *Proc. LREC*.

Horacio Saggion, Sanja Štajner, Stefan Bott, Simon Mille, Luz Rello, and Biljana Drndarevic. 2015. Making It Simplext: Implementation and Evaluation of a Text Simplification System for Spanish. *ACM Transactions on Accessible Computing (TACCESS)* 6(4).

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

Advaith Siddharthan. 2002b. Resolving attachment and clause boundary ambiguities for simplifying relative clause constructs. In *Proc.Student Workshop, ACL*.

Advaith Siddharthan. 2011. Text Simplification using Typed Dependencies: A Comparison of the Robustness of Different Generation Strategies. In *Proc. 13th European Workshop on Natural Language Generation*.

Advaith Siddharthan. 2014. A Survey of Research on Text Simplification. *International Journal of Applied Linguistics* 165(2):259–298.

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

Gabriel Stanovsky and Ido Dagan. 2016. Annotating and Predicting Non-Restrictive Noun Phrase Modifications. In *Proc. ACL*.

David Vickrey and Daphne Koller. 2008. Sentence Simplification for Semantic Role Labeling. In *Proc. ACL*.

Sanja Štajner, Maja Popović, Horacio Saggion, Lucia Specia, and Mark Fishel. 2016. Shared task on quality assessment for text simplification. In *Proc. LREC*.

Sander Wubben, Antal van den Bosch, and Emiel Krahmer. 2012. Sentence Simplification by Monolingual Machine Translation. In *Proc. ACL*.

Wei Xu, Chris Callison-Burch, and Courtney Napoles. 2015. Problems in current text simplification research: New data can help. *Transactions of the Association for Computational Linguistics* 3:283–297.

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