Text Simplification as Tree Transduction

Gustavo H. Paetzold[1] and **Lucia Specia**[2]

[1]Departamento de Ciência da Computação, Universidade Estadual do Oeste do Paraná
Rua Universitária nº2069, CEP 85819-110, Cascavel-PR, Brasil
[2]Department of Computer Science, University of Sheffield
Regent Court, 211 Portobello Street, Sheffield S1 4DP, UK

Abstract. *Lexical and syntactic simplification aim to make texts more accessible to certain audiences. Syntactic simplification uses either hand-crafted linguistic rules for deep syntactic transformations, or machine learning techniques to model simpler transformations. Lexical simplification performs a lookup for synonyms followed by context and/or frequency-based models. In this paper we investigate modelling both syntactic and lexical simplification through the learning of general tree transduction rules. Experiments with the Simple English Wikipedia corpus show promising results but highlight the need for clever filtering strategies to remove noisy transformations.*

Resumo. *A simplificação em nível lexical e sintático objetiva tornar textos mais acessíveis a certos públicos-alvo. Simplificação em nível sintático usa regras confeccionadas manualmente para empregar transformações sintáticas, ou técnicas de aprendizado de máquina para modelar transformações mais simples. Simplificação em nível lexical emprega busca por sinônimos para palavras complexas seguida por análise de contexto e/ou busca em modelos de frequência de palavras. Neste trabalho investiga-se a modelagem de ambas estratégias de simplificação em nível sintático e lexical pelo aprendizado de regras através da transdução de árvores. Experimentos com dados da Simple English Wikipedia mostram resultados promissores, porém destacam a necessidade de estratégias inteligentes de filtragem para remover transformações ruidosas.*

1. Introduction

Syntactic text simplification approaches modify a sentence's structure in order to make it easier to comprehend, whereas lexical text simplification approaches mainly apply localised modifications to words based on their local lexical context (often a sentence). Most work on syntactic simplification is based on hand-crafted rules [Siddharthan 2004, Gasperin et al. 2009a]. Manually crafting syntactic rules is costly and time consuming, which often results in sets of rules that are virtually purely syntactic and thus only cover very general phenomena. One such example is the splitting of sentences with multiple clauses, where very little lexical information, such as a small set of clause markers (e.g. relative pronouns or conjunctions), is represented. Adding lexical information to such rules would make the rule creation process even more costly. It is often the case that these rules only apply to a small subset of the complex sentences. For example, in an experiment with the rules available in [Siddharthan 2004]'s system on a random subset of sentences deemed complex in the Simple English Wikipedia corpus [Coster and Kauchak 2011], we found that only 30% of these sentences are covered by the rules available.

Proceedings of the 9th Brazilian Symposium in Information and Human Language Technology, pages 116–125,
Fortaleza, CE, Brazil, October 21–23, 2013.

Efforts on using statistical and machine learning techniques to address syntactic simplification include several approaches inspired by phrase- or tree-based statistical models for machine translation [Specia 2010, Zhu et al. 2010, Wubben et al. 2012], which learn limited transformations such as short-distance reordering. [Bach et al. 2011] design templates for subject-verb-object simplification, which limit the application of rules to cases matching the templates. A more general approach is proposed in [Woodsend and Lapata 2011] using a quasi-synchronous grammar and integer programming to generate syntactic simplification rules. These are generally precise, but at the cost of low coverage.

Most work on lexical simplification is based on synonym substitution using frequency-related or readability statistics, and also context models based mostly on bag-of-words to identify whether a candidate substitution fits the context [Keskisarkka 2012, Kandula et al. 2010, Carroll et al. 1998]. For a comprehensive overview, we refer the reader to the SemEval-2012 shared task on the topic [Specia et al. 2012]. Overall, lexical simplification work disregards explicit syntactic cues.

To overcome these limitations we investigate the learning of tree transduction rules to produce larger volumes of both lexical and lexicalized syntactic simplification rules. This approach has the potential to produce rule sets with high coverage and variability, and at the same time make them more specific by lexicalising some of the rule components of syntactic simplification rules.

2. Simplification Approach

Our simplification approach uses as a basis possible transformations learned with the Tree Transducer Toolkit (T3) [Cohn and Lapata 2009]. The approach is composed by three main components: a training module, a simplification module, and a ranking module.

2.1. Training Module

This module is responsible for generating a large number of candidate transformation rules, some of which will be purely syntactic or purely lexical, while others will be lexico-syntactic. It takes as input a parsed version of a parallel corpus of complex and simple sentences aligned at the word level (produced using Meteor 1.4 [Denkowski and Lavie 2011]). It uses the *harvest* function in T3 to extract a synchronous tree-substitution grammar which describes tree transformations. The *harvest* function extracts the maximally general (smallest) pairs of tree fragments that are consistent with the word-alignment. In Table 1 we show examples of transformations produced by this function using the Simple Wikipedia corpus, where # indicates indexes of place holders for any type of syntactic subtree.

By inspecting Table 1, one will notice that the tree transformations can take very distinct forms. The first example shows a transformation which removes the two items identified by #0 and #2 and transfers the remaining items to a different tree structure. The second example illustrates a transformation that swaps the order of its items #0 and #1 by transferring them from two sentences 'S' connected through the conjunction 'and' into a single sentence 'S', composed by a noun phrase 'NP' and a verbal phrase 'VP'. Transformations of this type resemble simplification operations such as constituent reordering and deletion of non-essential information, since they are able to either rearrange or remove large portions of a sentence's syntactic structure. The third and fourth entries in the Table

Table 1. Examples of rules produced by T3

Original Tree	Simplified Tree
(NP (FW #0)(FW #1)(JJ #2)(NN #3))	(NP (NP (NNS #1))(VP #3))
(S (S #0) (CC and) (S #1) (. #2))	(S (NP #1) (VP #0) (. #2))
(ADJP (ADJP #0) (CONJP #1) (ADJP #2))	NULL
(NP (DT #0) (JJ #1) (NNPS #2))	NULL
(NP (DT a) (JJ significant) (NN role))	(NP (DT an) (JJ important) (NN part))
(VBD intensified)	(VBD strengthened)
(ADVP (RB as) (RB well) (RB as))	(RB well)

may also resemble non-essential segment deletion operations, since both transformations align the original tree to a 'NULL' value (or empty tree).

Unlike the first four entries in Table 1, the fifth, sixth and seventh transformations convert the original into simplified trees based on specific lexical items. These transformations can be compared to operations in lexical simplification, but it is interesting to notice how the syntactic context is taken into account in this case, as opposed to the much simpler bag-of-words context used in previous work. When applied, these transformations replace words in a given complex sentence with their simpler equivalents as long as the syntactic constraints are met.

Extracting reliable tree transformations from the parallel corpus used in our experiments is very challenging for various reasons: (i) not all parallel sentences contain complex-simplified pairs: some original sentences remain the same in the "simplified" version, others are modified, but not simplified; (ii) the simplifications (and modifications in general) are not systematic: many cases are completely rewritten, often in ad hoc ways; (iii) the corpus was compiled automatically, and is thus noisy. Table 2 illustrates examples

Table 2. Examples of sentence pairs extracted from our parallel corpus

Well behaved sentence pairs

Unsimplified:	Ebba von Sydow is a Swedish journalist, TV personality, fashion blogger and author.
Simple:	Ebba von Sydow is a Swedish journalist and author.
Unsimplified:	Depending upon the source, it is estimated that 50,000-70,000 Romans were killed or captured at Cannae.
Simple:	It is estimated that 50,000-70,000 Romans were killed or captured at Cannae.

Noisy sentence pairs

Unsimplified:	As of, Jyvskyl had a population of.
Simple:	There are about 128.016 inhabitants in Jyvskyl. The University of Jyvskyl is popular and respected.
Unsimplified:	Aguilera also released a Christmas album on October 24, 2000 called My Kind of Christmas.
Simple:	Aguilera made two other albums then. They were a Latin pop album Mi Reflejo, and a Christmas album My Kind of Christmas.

of both well behaved and noisy sentence pairs extracted from our parallel corpus.

As a consequence, the very large set of tree transformations learned by the *harvest* function in T3 contains a wide variety of transformations which may or may not be characterised as simplification transformations; many of which are spurious. Therefore, filtering strategies are necessary. In order to better understand the applicability of these rules to the problem of text simplification, and in particular, the applicability of syntactic and lexical rules in isolation, in the training module we apply separate procedures to filter rules into two subsets containing (a) syntactic transformations only, or (b) lexical transformations only.

2.1.1. Selecting Syntactic Rules

This procedure checks all entries in the raw transformation set produced by T3, selecting those transformations that are more likely to represent syntactic simplification operations. Rules satisfying any of following criteria are selected:

1. **Most general rules:** Select rules which have only variables in their leaf nodes, as these often rewrite large portions of a complex sentence as opposed to individual words. Those transformations may be identified by simply evaluating if all leaf nodes in the rule's both original and simplified trees are all started with the character '#'.
2. **Segmentation rules:** Select rules that split a single sentence into two or more smaller sentences. Splitting a long sentence into shorter equivalents is known to facilitate reading comprehension. Identifying such rules is done based on how many leaf nodes of class '.' ('final dot') are found in the rule's left and right handed trees: if the right handed tree possesses more '.' classed leaf nodes than the left handed tree, then add the rule to the syntactic simplification rule set.
3. **Deletion rules:** Select rules that delete information, i.e., rules with the 'NULL' token on the right hand side. Some deletion rules might remove essential information from a sentence's content, such as its subject or object. Taking this problem into account, both the simplification and ranking modules of our system present solutions to prevent this phenomena from compromising the simplification results. The operation addressed by this criteria may be classified as a lexical simplification operation as well.
4. **Rules with connectors:** Select rules which cover sentences with connectors, such as "and" and "or". With a motivation similar to the one for selecting segmentation rules, this criteria aims to find rules that transform sentences composed by multiple clauses, to an equivalent sentence which presents the content in a way that facilitates its reading comprehension. This type of rule is found through the search in the rule's left hand tree for a leaf node of class 'CC', which characterises a coordinating conjunction.

2.1.2. Selecting Lexical Rules

Similar to syntactic rule selection, this procedure evaluates all entries in the raw set to select the most promising transformations, but now from a lexical perspective. The following selection criteria are used to focus strictly on the substitution of lexical items:

1. **Leaf nodes must be composed strictly by words:** Transformations with generic lexical items starting with the character # tend to rearrange large portions of the sentence, which commonly characterize a syntactic simplification transformation.
2. **Must not be a deletion rule:** Removing a specific set of terminals from a given sentence tend to produce unwanted gaps in it, reducing its readability and grammaticality.
3. **Rule must have the same part-of-speech tag in the root of left- and right-hand sides:** Switching words of different lexical categories may produce comic results. Consider, for an example, replacing the verb *'sold'* by the noun *'sale'* in the phrase *'The girl sold lemonade'*.
4. **One side of the rule must not be a substring of the other:** Removing or adding connective words such as prepositions and adverbs to a sentence may compromise its grammaticality. Consider the example of the seventh entry in table 1, which aligns the syntactic structure of the phrase *'as well as'* in its left hand tree, with its right hand syntactic tree of the word *'well'*. By applying this transformation to the sentence *'He was tall as well as thin'*, the resulting sentence would be *'He was tall well thin'*, which is neither simpler nor grammatical.
5. **None of the sides of the rule can contain a numeral or proper noun:** The presence of those word categories may imply in uninformative lexical transformations. Numerals and proper nouns usually characterise information that is very specific to the sentence of the training corpus from which the rule has been extracted.
6. **Rule cannot replace a single word by one of its morphological variants:** Transformations which replace a singular noun for its plural equivalent are examples comprehended by this selection criteria. Consider replacing the word *'dog'* for its singular equivalent *'dogs'* in *'A dog is commonly more loyal than a cat'*.

2.2. Simplification Module

The overall role of the simplification module is to produce syntactically and lexically simplified versions of a new complex sentence. It takes as input the syntactic and lexical simplification rule sets, and the complex sentence to be simplified. In our experiments, we assume this is always a complex sentence that can be simplified in one or more ways. The decision on whether the sentence is in fact complex and needs simplification is outside the scope of this work (see [Gasperin et al. 2009b] for an example of approach to make this decision). As previously mentioned, in order to evaluate the effect of both syntactic and lexical simplification in isolation, the module applies the two sets of rules independently.

In the syntactic simplification step, the module uses each and every one of the selected syntactic simplification rules (Section 2.1.1) to produce multiple simplified versions of the complex input sentence. Those simplified versions then go through a further simple filtering step, which removes candidates which are identical or 50% either longer or shorter than the original complex sentence. This prevents excessive compression or expansion of the original sentence, as these are often very prone to errors. All eligible syntactically simplified candidates are then ranked such that the one-best candidate is selected. The ranking is described in Section 2.3.

Similarly, in the lexical simplification step the module uses the lexical rule set to craft the lexically simplified version of the complex input sentence. This step analyses the complex sentence, searches for terms in its structure which are entries in the lexical rule

set, and replaces them by their simpler counterpart. If the sentence has multiple complex terms or multiple possible replacements for them, the process creates a list of candidates considering all possible combinations of replacements. This list of lexically simplified candidates is then ranked such that the best candidate is selected. The ranking procedure is the same for both syntactic and lexical simplification, as described in Section 2.3.

The time consumed by the production of simplified candidates grows proportionally to the number of syntactic and lexical simiplification rules obtained. To avoid combinatory explosion, we employ algorithm optimization through the usage of fast data structures, such as hash-tables, to both syntactic and lexical simplification algorithms.

2.3. Ranking Module

This is a simple module that scores all candidate simplifications for a given complex sentence produced by the lexical or syntactic simplification modules. The scores are then used to rank the candidates such that the top scoring candidate can be selected. At this stage, we score candidates using their predicted perplexity based on a language model of simplified sentences, as described in the next section.

3. Experimental Settings

Training Corpus For the extraction of the tree transformations, we use 133K pairs of original-simple sentences from the Simple Wikipedia corpus [Coster and Kauchak 2011] parsed by the Stanford constituency parser [Klein and Manning 2003].

Language Model For this experiment a 3-gram language model is used to rank candidate simplifications. The model is trained on a superset of the Simple Wikipedia corpus containing 710K simple sentences. We use the SRILM toolkit [Stolcke et al. 2011] to build the 3-gram language model. The motivation to use a 3-gram rather than a 4 or 5-gram language model is based on the studies presented at [Chen and Goodman 1996], which suggests that language models of higher order tend to lead to data sparsity when the corpus is either noisy or not large enough.

Test Corpus The corpus to be simplified in our tests contains 130 original sentences randomly selected from a held-out portion of the parallel Simple Wikipedia corpus. All sentences have a simpler version in the corpus which is different from the original version, and vary between 50 and 240 characters in length.

Evaluation Metrics We evaluate the simplification both automatically by the string matching metric BLEU [Papineni et al. 2002], and manually using five human subjects. Our system produces two simplified versions for each sentence, resulting in 260 simplified sentences to be evaluated. As part of the manual evaluation, each evaluator is assigned 100 sentences: the syntactically and lexically simplified versions of 50 originally complex sentences. 30 of these sentences are common to all evaluators so that inter-annotator agreement can be computed. Evaluators are asked to answer three questions for each syntactically or lexically simplified sentence:

- Is it **simpler** than the original sentence?
- Is it **grammatical**?
- Does it preserve the **meaning** of the original sentence?

All answers are binary (*yes/no*), but *not applicable* is also an option, e.g. for cases that are too ungrammatical for one to judge simplicity. Short guidelines specify that a simpler sentence must have a structure which eases its understanding by the reader and/or have fewer terms which may challenge non-native readers. To make the evaluation less subjective, all evaluators are non-native speakers of English.

4. Results

Automatic evaluation is a long-standing issue for text simplification. Previous work has used machine translation metrics such as BLEU, which computes n-gram matchings between the system output and a human output. We obtained the same BLEU score of 0.342 for both lexical and syntactic simplifications, suggesting that the two variants of our approach are of indistinguishable quality, which we do not believe to be the case. In addition, these scores are difficult to interpret in absolute terms, particularly for this task where a *lazy* approach that performs no simplifications at all would result in the highest possible BLEU score.

In the manual evaluation, the Cohen's kappa inter-annotator agreement [Carletta 1996] between pairs of evaluators was found to range from 0.32 (fair) and 0.68 (substantial), showing that manual evaluation is a challenge on its own. As for possible reasons to why inter Table 3 shows the percentages of positive (*yes*) answers resulting from this evaluation. Examples of acceptable and poor resulting sentences are given in Table 4.

Table 3. Manual evaluation results

Simplification type	Syntactic	Lexical
Simpler	26.12%	42.15%
Grammatical	38.28%	83.40%
Meaningful	34.68%	56.50%

From the training corpus, T3 produced over 2.3 million tree transformations, which resulted in sets with 562K syntactic and $4,809$ lexical simplification rules after our selection filters were applied. This very large number of syntactic rules ensured that all test sentences were covered, but many of these rules are spurious. As a consequence, the results for syntactic transformations are rather disappointing: only a small percentage of the sentences are judged simpler, at the cost of modifications that render other sentences ungrammatical and/or with an incorrect meaning. A manual inspection of the syntactic simplifications showed several cases of sentences with essential portions of their syntactic structure deleted, such as noun or verbal phrases. Better filtering strategies for rules are necessary, as are better ranking techniques for the resulting transformations.

The results for lexical simplification are more encouraging, with almost half of the sentences being judged simpler while still grammatical in over 80% of the cases. Often replacements have incorrect senses, showing that there is room for better context models to make sure only replacements with the same word sense are performed. In most cases, only one word per sentence is replaced. A larger parallel corpus for training and the combination with external dictionaries could help simplify more words.

Table 4. Examples of acceptable and poor simplifications

Acceptable syntactic simplifications	
Original:	Transportation systems **from flight to automobiles** increasingly use embedded systems.
Simple:	Transportation systems increasingly use embedded systems.
Original:	Individuals seeking membership must be legally defined as adults in their nation **of residence.**
Simple:	Individuals seeking membership must be legally defined as adults in their nation.
Poor syntactic simplifications	
Original:	As with larger cakes, **frosting and other cake decorations,** such as sprinkles, are common on cupcakes.
Simple:	As with larger, cake, such as sprinkles, are on cupcakes.
Original:	Mickey Mouse Clubhouse is a **childrens television series**, that premiered in prime time on Disney Channel on May 5, 2006.
Simple:	Mickey Mouse Clubhouse is a, that premiered in prime time on Disney Channel on May 5, 2006.
Acceptable lexical simplifications	
Original:	Jehans **father** is Akbar Hussain Rizvi, who is the **eldest** son of Noor Jehan.
Simple:	Jehans **dad** is Akbar Hussain Rizvi, who is the **oldest** son of Noor Jehan.
Original:	A digital signature or digital signature scheme is a mathematical **scheme** for **demonstrating** the authenticity of a digital message or document.
Simple:	A digital signature or digital signature scheme is a mathematical **project** for **showing** the authenticity of a digital message or document.
Poor lexical simplifications	
Original:	Niccola Machiavelli, author of The Prince, **also** witnessed and wrote about the execution.
Simple:	Niccola Machiavelli, author of The Prince, **very** witnessed and wrote about the execution.
Original:	He became paralysed by a stroke in 1921 and never **recovered**, dying at Long Ashton near Bristol in England on July 23, 1927.
Simple:	He became paralysed by a stroke in 1921 and never **found**, dying at Long Ashton near Bristol in England on July 23, 1927.

5. Remarks

We have presented an approach for syntactic and lexical text simplification via the learning of tree-substitution grammars followed by rule selection and ranking of the modified sentences to produce "simpler" versions of complex sentences. Our experimental results showed that this approach is yet to be improved, particularly with respect to syntactic simplification, since most transformations come at the cost of loss of grammaticality or change of meaning. On the positive side, we were able to show the potential of automatically induced tree-substitution grammars for this problem: very large sets of rules are generated, with much higher coverage than work based on hand-crafted rules or the learning of small subsets of "reliable" transformations. In future work we will investigate better filtering and ranking strategies to generate more accurate simplifications, including ranking strategies such as the ones suggested in the Coh-Metrix project [Graesser et al. 2004].

References

Bach, N., Gao, Q., Vogel, S., and Waibel, A. (2011). TriS: A statistical sentence simplifier with log-linear models and margin-based discriminative training. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 474–482, Chiang Mai, Thailand.

Carletta, J. (1996). Assessing agreement on classification tasks: the kappa statistic. *Comput. Linguist.*

Carroll, J., Minnen, G., Canning, Y., Devlin, S., and Tait, J. (1998). Practical simplification of english newspaper text to assist aphasic readers. In *AAAI-98 Workshop on Integrating Artificial Intelligence and Assistive Technology*.

Chen, S. F. and Goodman, J. (1996). An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 310–318. Association for Computational Linguistics.

Cohn, T. and Lapata, M. (2009). Sentence Compression as Tree Transduction. *Journal of Artificial Intelligence Research.*

Coster, W. and Kauchak, D. (2011). Simple english wikipedia: a new text simplification task. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 665–669, Portland, Oregon.

Denkowski, M. and Lavie, A. (2011). Meteor 1.3: automatic metric for reliable optimization and evaluation of machine translation systems. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, Stroudsburg, PA, USA. Association for Computational Linguistics.

Gasperin, C., Maziero, E., Specia, L., Pardo, T., and Aluisio, S. M. (2009a). Natural language processing for social inclusion: a text simplfication architecture for different literacy levels. In *SEMISH-XXXVI*.

Gasperin, C., Specia, L., Pereira, T., and Aluisio, S. (2009b). Learning when to simplify sentences for natural text simplification. In *Proceedings of ENIA*, pages 809–818, Bento Gonćalves, Brazil.

Graesser, A., McNamara, D., Louwerse, M., and Cai, Z. (2004). Coh-Metrix: Analysis of text on cohesion and language. pages 193–202.

Kandula, S., Curtis, D., and Zeng-Treitler, Q. (2010). A semantic and syntactic text simplification tool for health content. *AMIA Symposium*.

Keskisarkka, R. (2012). Automatic text simplification via synonym replacement. Master thesis, Linkoping University.

Klein, D. and Manning, C. D. (2003). Accurate unlexicalized parsing. In *41st Annual Meeting of the Association for Computational Linguistics*.

Papineni, K., Roukos, S., Ward, T., and Zhu, W. (2002). Bleu: a method for automatic evaluation of machine translation. In *ACL02*, pages 311–318, Philadelphia, Pennsylvania.

Siddharthan, A. (2004). Syntactic simplification and text cohesion. Technical report, University of Cambridge.

Specia, L. (2010). Translating from complex to simplified sentences. In *Proceedings of the 9th international conference on Computational Processing of the Portuguese Language*, PROPOR'10, pages 30–39, Porto Alegre, RS, Brazil. Springer-Verlag.

Specia, L., Jauhar, S. K., and Mihalcea, R. (2012). Semeval-2012 task 1: English lexical simplification. In *Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 347–355, Montréal, Canada.

Stolcke, A., Zheng, J., Wang, W., and Abrash, V. (2011). SRILM at sixteen: Update and outlook. In *IEEE Automatic Speech Recognition and Understanding Workshop*.

Woodsend, K. and Lapata, M. (2011). Learning to simplify sentences with quasi-synchronous grammar and integer programming. EMNLP.

Wubben, S., van den Bosch, A., and Krahmer, E. (2012). Sentence simplification by monolingual machine translation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 1015–1024, Jeju Island, Korea.

Zhu, Z., Bernhard, D., and Gurevych, I. (2010). A monolingual tree-based translation model for sentence simplification. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 1353–1361, Beijing, China.