Simplification by Lexical Deletion

Matthew Shardlow
Manchester Metropolitan University,
Manchester, United Kingdom
m.shardlow@mmu.ac.uk

Piotr Przybyła
Universitat Pompeu Fabra, Barcelona, Spain
and Institute of Computer Science,
Polish Academy of Sciences, Warsaw, Poland
piotr.przybyla@upf.edu

Abstract

Lexical simplification traditionally focuses on the replacement of tokens with simpler alternatives. However, in some cases the goal of this task (simplifying the form while preserving the meaning) may be better served by removing a word rather than replacing it. In fact, we show that existing datasets rely heavily on the deletion operation. We propose supervised and unsupervised solutions for lexical deletion based on classification, end-to-end simplification systems and custom language models. We contribute a new silver-standard corpus of lexical deletions (called SimpleDelete), which we mine from simple English Wikipedia edit histories and use to evaluate approaches to detecting superfluous words. The results show that even unsupervised approaches (TerseBERT) can achieve good performance in this new task. Deletion is one part of the wider lexical simplification puzzle, which we show can be isolated and investigated.

1 Introduction

Lexical simplification aims to identify words that are too difficult for readers and apply an intervention that will enable them to better understand the term. In almost all lexical simplification studies, the primary intervention is the **replacement** operation, in which the target word is substituted for a simpler alternative (Carroll et al., 1998; Shardlow, 2014). Some studies have also considered applying an **addition** operation by adding a definition or explanation of the difficult term (Srikanth and Li, 2020; Kloehn et al., 2018). We propose that a new operation should be considered for lexical simplification, which is that of **deletion**.

Lexical deletion is a vital element in making texts easier to understand. A prior analysis of simple English Wikipedia showed that 47% of sentence simplifications involved deleting words (Coster and Kauchak, 2011b). To better understand the role of deletions in modern TS datasets, we performed an analysis of reference datasets in the EASSE package for text simplification evaluation (Alva-Manchego et al., 2019). The results in Table 1 show that deletions are important, making up 26.82% of the operations for the turkcorpus and 37.42% of operations for PWKP. Whilst other operations are still essential for full sentence simplification, deletion is a vital yet understudied edit operation, on which we choose to focus this study.

Reference	Delete	Add	Keep
Turk Corpus	26.82	20.19	53.00
PWKP	37.42	23.17	39.41

Table 1: The proportion of operation types between two common text simplification evaluation reference sets.

Consider the following sentence, in which the word 'erudite' has been highlighted as a candidate for simplification:

*"Aristotle was an **erudite** scholar."*

We could choose to substitute the difficult term by searching for a simpler alternative (learned, knowledgeable, intelligent, etc.). However, we could also simply omit the term, leading to the sentence[1]:

"Aristotle was a scholar."

The new sentence loses some details from the original meaning (was Aristotle a particularly well read scholar or just a mediocre one?), yet is undoubtedly simpler for a reader to understand. The overall meaning of the sentence is preserved and a reader is less likely to stumble over the difficult term. This goes beyond traditional lexical simplification, where only complex words are considered, as it may be beneficial to delete simple words from a sentence without losing any meaning (e.g. dropping 'located' in: 'Times Square is **located** in New York'). We present further examples of deletions taken directly from our corpus in Table 2.

[1]We make the article agree manually, which is not strictly part of the task but can be done with a simple algorithm.

2 Related Work

Simplification by deletion has been studied as an emergent property of systems which perform simplification through sentence to sentence translation (Coster and Kauchak, 2011a; Nisioi et al., 2017; Kumar et al., 2020). These systems are trained on parallel datasets that contain a variety of operations including deletion (Alva-Manchego et al., 2017). It is also possible to force these systems to provide certain types of operations through the use of control tokens (Martin et al., 2020).

Sentence deletion has also been studied as a means of discourse simplification, where the aim is to drop redundant sentences in a passage (Štajner and Glavaš, 2017; Stajner et al., 2013). This is similar to the task of extractive summarisation (Knight and Marcu, 2002; Nenkova and McKeown, 2012) where the task is to only retain the relevant sentences. Conversely, lexical deletion is similar to sentence compression (Filippova and Altun, 2013), where the goal is to remove all redundant information from a sentence.

Typical evaluation of simplification has focussed on either matching n-grams (Štajner et al., 2014; Wubben et al., 2012) (e.g., BLEU-score (Papineni et al., 2002)) or analysing the lexical simplification pipeline (Paetzold and Specia, 2016). SARI score (Xu et al., 2016) has become dominant in the evaluation of text simplification, however it is designed for full sentence simplification, and does not explicitly measure text coherence. Nonetheless, SARI score has been used to measure the ability of a system to perform deletions (Kumar et al., 2020). The recent Shared Task on Multilingual Lexical Simplification at the TSAR workshop (Saggion et al., 2022) popularised several metrics for the evaluation of lexical simplification, including Accuracy@k@top1 and Mean Average Precision (MAP@k). These evaluation methods are appropriate when a ranked list of candidates is produced.

Our work leverages simple English Wikipedia edit histories, drawing on a long line of prior simplification studies to generate corpora using this resource. Simple English Wikipedia has been shown to contain the type of language that is useful for simplification models (Kauchak, 2013). The edit histories have been used previously to mine examples (Yatskar et al., 2010) and corpora (Shardlow, 2013) of complex words. English Wikipedia has also previously been used to generate candidate sentences for the complex word identification task (Yimam et al., 2017). Parallel articles from simple and regular English Wikipedia have also been aligned to generate examples of parallel sentences for training text simplification models (Zhu et al., 2010; Jiang et al., 2020).

3 Corpus Development

We take an approach similar to our prior work (Shardlow, 2013), by mining simple English Wikipedia edit histories. We hypothesise that editors are typically trying to simplify the texts when editing them and so any cases we find of a single word being dropped (with some caveats listed below) are likely to be examples of simplification by deletion.

We download the most recent version of the Simple Wikipedia edit histories as an XML file[2] and compare successive revisions of each page using the following pipeline of operations:

1. Converting the WikiText to plain text using *Sweble* (Dohrn and Riehle, 2011).

2. Parsing the document for sentences and tokens using *Stanford CoreNLP* (Manning et al., 2014).

3. Identifying candidate sentences that contained all but one of the tokens, preserving order, from a sentence in the prior revision.

4. Checking if the deleted word was a dictionary word (defined as any token with frequency above 10,000 in the Google Web1T (Brants and Franz, 2006)).

5. Ignoring sentences longer than 30 tokens, as such lines often tended to be spam or vandalism (unfortunately, Wikipedia edit histories exhibit wilful acts of destruction or overwriting to the contents therein, usually quickly reverted by an editor — yet recorded in the edit history).

6. Removing contexts containing very long tokens (20+ characters), usually resulting from errors in parsing malformed wikitext.

7. Ensuring that each deleted word was a single token in lowercase and contained no punctuation.

[2]`https://dumps.wikimedia.org/simplewiki/latest/`, our version was dated 2021-04-01

ID	Example
1	Naturalization makes them **naturalized** citizens of their new country.
2	Plants include **familiar** types such as tree, herb, bushes, grass, vine, fern, moss, and green algae.
3	There were many brooks providing **fresh** water.
4	Bullock is the **usual** word for beef cattle.
5	He was best known for his **trenchant** secularism.

Table 2: Examples from the SimpleDelete Corpus. The dropped token is in boldface type.

8. Excluding cases where the dropped word was the first token in the sentence, as these were often superfluous headings that were being removed.

9. Removing cases, where the deleted word is included in a list of offensive terms[3], extended with several malicious terms (vandalism, etc.) that occurred frequently in the corpus.

10. Ensuring that all examples were a minimum of 2 characters long.

This procedure yields 18,082 cases of lexical deletions. We split these data into train, validation and test subsets according to the deleted token (to prevent the same token occurring in test and train sets). We select one non-deleted token per context to create a negative class (preserving the original token-based stratification) to give a final corpus size of 36,164 instances (train: 28,836, validation: 3,678, test: 3,650). We release the data, the partitions and the code used to generate the corpus via GitHub[4].

Examples of the types of deletions in our corpus are provided in Table 2. Whereas examples 1 and 5 are potentially difficult words, 2–4 are undoubtedly simple. Yet, removing these makes each sentence more intelligible, whilst preserving the meaning.

To validate our silver standard corpus, the first author examined 600 examples from the validation set (300 from each class), deciding on the correctness of each instance. The result of this showed that 92.33% of the positive class (true deletions) in this sample of our corpus were valid, as were 96.00% of the negative class. Rejected examples in the positive class included cases of vandalism that were not picked up by our token blacklist or simply words that had been removed in error by the editor, whereas cases in the negative class represented cases where the randomly selected word was also a good candidate for removal. We did not perform a full validation of our entire corpus due to the high number of instances contained therein. In total, 94.17% of examples in our silver standard corpus were acceptable based on the entire 600 instance sample.

4 Prediction of Lexical Deletions

We test the following four methods for predicting lexical deletion, covering unsupervised and supervised techniques, as well as a sentence simplification system capable of deletions.

4.1 TerseBERT (Unsupervised)

The core question any solution for this problem must address can be stated as: *Is the given word necessary in this context?*. We can easily see how similar this is to the main question of language modelling, namely *What word is likely in this context?*. Therefore, in our first approach, we build on a pretrained language model, namely BERT (Devlin et al., 2018).

As the regular version of BERT can only predict the most probable replacements for a given word in context but not estimate the probability of no word being required, it is not suitable for the purpose of this study. Therefore, we use *TerseBERT* (Przybyła and Shardlow, 2020), which is a custom version of the BERT model, originally developed for multi-word lexical simplification. TerseBERT includes a special token, `[NONE]`, which reflects the probability that the left and right context of the given mask position occur directly after each other, with no words between them. Here we use the model by obtaining its predictions for each token, masked separately, and treating the probability of `[NONE]` as a deletion score.

4.2 SVM with fastText Embeddings (Supervised)

We use the Scikit-Learn (Pedregosa et al., 2011) implementation of the linear kernel SVM (Fan et al., 2008). The features include: *fastText* embeddings (Joulin et al., 2016) for the candidate token, whole context, context preceding the candidate token (left-

[3]taken from: https://www.cs.cmu.edu/~biglou/resources/bad-words.txt

[4]https://github.com/MMU-TDMLab/SimpleDelete

context) and context following the candidate token (right-context). To calculate the embedding for the multi-word context(s) we collect the embeddings for each token in the fragment and select the maximum value across each dimension to give a single embedding vector. Whilst we did check for the relevance of each feature set using the validation data, we found that the best policy was to use all feature sets during training. The SVM is trained using the training portion of our data which contains true deletions (class label = 1) and randomly selected examples (class label = 0).

4.3 Fine-tuned BERT-large (Supervised)

We use the HuggingFace implementation of the PyTorch BERT-Large-uncased model[5]. We fine-tune for 5 epochs on our data using the given parameters (Adam optimiser, warmup steps = 500, weight decay = 0.01, learning rate = 0.001). All experiments are evaluated using the validation data to check configurations of our task. The final results are given by applying the fine-tuned model to the test data. To encode our problem we provide the following sequence: `Context [SEP] Token`. Where the context and token are provided by our corpus and the class variable is assigned as previously.

4.4 ACCESS (Baseline)

We select a state-of-the-art simplification model, ACCESS (Martin et al., 2020), which is capable of lexical or clausal deletion and ran it over the contexts in our test dataset using the default configuration. We check for each context whether a word was still present or not in the simplified outputs of these models. We did not constrain ACCESS to only perform deletes, however this is to the system's benefit as other operations, such as replacements, will be considered deletes.

5 Results

We evaluate our task and methods in a variety of settings as described below in order to better understand the nature of the lexical deletion problem.

5.1 Candidate Rank According to Deletion Score

We calculate the deletion score using TerseBERT for every word in each context in our validation set and check the rank of the candidate token, normalising by sentence length. Figure 1 shows that

[5] https://huggingface.co/transformers/

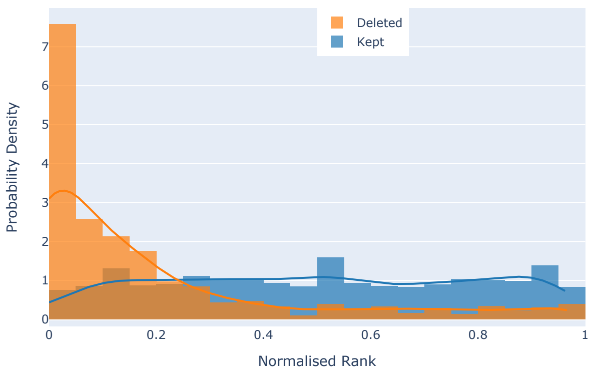

Figure 1: The normalised rank of deleted tokens vs. kept tokens on the validation set.

Type	System	P	R	F1
U	TerseBert$_{0.03}$	0.677	0.942	0.788
U	TerseBert$_{0.27}$	0.746	0.850	0.795
U	ACCESS	0.719	0.472	0.570
S	SVM	0.766	0.666	0.712
S	BERT-large	0.870	0.830	0.850

Table 3: Deletion prediction (as binary classification) performance of different approaches on our dataset. TerseBert$_X$ refers to the deletion score being thresholded at X to give a binary classification. U and S refer to unsupervised and supervised systems with respect to our corpus.

whereas the positive class (truly deleted tokens) follows an exponential decrease, the negative class (tokens not deleted by the editors) follows a flat distribution, which is expected as these were randomly selected.

5.2 Binary Classification

We evaluate in a binary classification setting using the positive and negative classes in our corpus. We find thresholds for converting the deletion score of TerseBERT to a binary decision by selecting the value that gave the highest F1 score (0.02) or the best balance of precision and recall (0.27) on our training and validation data combined. We then perform a further analysis on our test set, which is reported in Table 3. We compare these results on our test set to ACCESS, the SVM and fine-tuned BERT-Large as described previously.

6 Discussion

We introduced the task of lexical deletion in the new context of lexical simplification. This is the first work of which we are aware to explicitly investigate lexical deletion as a simplification operation.

We also developed a new silver-standard corpus, SimpleDelete, mined from simple English Wikipedia edit histories and tested our results on it. Future work could move our corpus from silver to gold standard by verifying all 18,082 instances either manually or semi-automatically.

In our binary classification setting, we demonstrated supervised methods trained on our corpus (SVM, BERT-large) and unsupervised methods (TerseBERT, ACCESS) for the task of lexical deletion. ACCESS gives a low recall, but competitive precision, indicating it is capable of the type of deletions we have identified but does not perform these consistently. TerseBERT with a thresholded deletion score of 0.27 gave an F1 score of 0.795, which was higher than the SVM, but lower than BERT-large. As our corpus is silver standard, it is possible that the supervised methods may have also learnt corpus specific factors.

Our simplifications come directly from simple Wikipedia edit histories and we assume that editors remove words to improve the simplicity of the language. The examples in Table 2 and our manual validation indicate that this assumption is correct and that we have collected true examples of simplification by deletion.

In conclusion, we have introduced and evaluated the capacity of lexical deletion for simplification. As a result, we hope that future works in lexical simplification will also take the deletion operation into account as an alternative to lexical replacement.

Acknowledgements

The work of Piotr Przybyła was supported by the *Polish National Agency for Academic Exchange* through a *Polish Returns* grant number PPN/PPO/2018/1/00006.

Lay Summary

Text Simplification is the task of making written language easier to understand. It is a very natural task for a person, such as when explaining an idea or talking to a child. Research has shown that computer algorithms can be used to automatically make language easier to understand. Some simplification algorithms first identify the difficult words or phrases in a sentence and replace these with easier alternatives. This is usually called ‘lexical simplification’ (lexical here is a term from linguistics that refers to words). A typical lexical simplification system is composed of several operations:

- First, the system identifies any words that might be difficult for the reader.
- Second, the system proposes candidates that may be useful replacements for the difficult word.
- Next, the possible candidates are ranked according to factors such as their simplicity and contextual fit.
- Finally, the highest ranking candidate is inserted into the sentence in place of the original term.

We wanted to know whether difficult words can be deleted, instead of replaced as in previous research. In many sentences, the difficult words are not necessary to the overall meaning. Take the following example:

> He was best known for his **trenchant** secularism.

We could find a simpler word for ‘trenchant’, but we could also remove it and the sentence would mean the same. In particular we wanted to find out whether an algorithm could be used to predict when to delete words.

Our research looked at articles from Simple Wikipedia. Specifically, we examined how editors had changed these articles over time. We used a set of rules to find examples of words that had been deleted to make a sentence easier to read. We kept a record of the original sentence and the word that had been deleted from it. This allowed us to find over 36,000 examples. We noticed that many examples were ‘easy’ words that had been deleted. This was surprising as we did not know that you could make a sentence easier to read by removing simple words. Finally, we compared several algorithms for predicting deletions. We showed that it is possible to automatically find words to delete.

Our work could be used to help make language easier to read. One possible area that it could be used in is education. For example, a student could use a simplification tool to make difficult texts on the web easier to read.

References

Fernando Alva-Manchego, Joachim Bingel, Gustavo Paetzold, Carolina Scarton, and Lucia Specia. 2017. Learning how to simplify from explicit labeling of complex-simplified text pairs. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 295–305, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Fernando Alva-Manchego, Louis Martin, Carolina Scarton, and Lucia Specia. 2019. Easse: Easier automatic sentence simplification evaluation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations*, pages 49–54.

Thorsten Brants and Alex Franz. 2006. Web 1T 5-gram version 1. LDC2006T13.

John Carroll, Guido Minnen, Yvonne Canning, Siobhan Devlin, and John Tait. 1998. Practical simplification of english newspaper text to assist aphasic readers. In *Proceedings of the AAAI-98 Workshop on Integrating Artificial Intelligence and Assistive Technology*, pages 7–10.

Will Coster and David Kauchak. 2011a. Learning to simplify sentences using Wikipedia. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, pages 1–9, Portland, Oregon. Association for Computational Linguistics.

William Coster and David Kauchak. 2011b. 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, USA. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186. Association for Computational Linguistics.

Hannes Dohrn and Dirk Riehle. 2011. Design and implementation of the Sweble wikitext parser: unlocking the structured data of Wikipedia. In *Proceedings of the 7th International Symposium on Wikis and Open Collaboration*, pages 72–81.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *the Journal of machine Learning research*, 9:1871–1874.

Katja Filippova and Yasemin Altun. 2013. Overcoming the lack of parallel data in sentence compression. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1481–1491, Seattle, Washington, USA. Association for Computational Linguistics.

Chao Jiang, Mounica Maddela, Wuwei Lan, Yang Zhong, and Wei Xu. 2020. Neural CRF model for sentence alignment in text simplification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7943–7960, Online. Association for Computational Linguistics.

Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. 2016. Fasttext.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*.

David Kauchak. 2013. Improving text simplification language modeling using unsimplified text data. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1537–1546, Sofia, Bulgaria. Association for Computational Linguistics.

Nicholas Kloehn, Gondy Leroy, David Kauchak, Yang Gu, Sonia Colina, Nicole P Yuan, and Debra Revere. 2018. Improving consumer understanding of medical text: Development and validation of a new subsimplify algorithm to automatically generate term explanations in english and spanish. *Journal of medical Internet research*, 20(8):e10779.

Kevin Knight and Daniel Marcu. 2002. Summarization beyond sentence extraction: A probabilistic approach to sentence compression. *Artificial Intelligence*, 139(1):91–107.

Dhruv Kumar, Lili Mou, Lukasz Golab, and Olga Vechtomova. 2020. Iterative edit-based unsupervised sentence simplification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7918–7928, Online. Association for Computational Linguistics.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Louis Martin, Éric de la Clergerie, Benoît Sagot, and Antoine Bordes. 2020. Controllable sentence simplification. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 4689–4698, Marseille, France. European Language Resources Association.

Ani Nenkova and Kathleen McKeown. 2012. A survey of text summarization techniques. In *Mining text data*, pages 43–76. Springer.

Sergiu Nisioi, Sanja Štajner, Simone Paolo Ponzetto, and Liviu P. Dinu. 2017. Exploring neural text simplification models. In *Proceedings of the 55th Annual Meeting of the Association for Computational*

Linguistics (Volume 2: Short Papers), pages 85–91, Vancouver, Canada. Association for Computational Linguistics.

Gustavo Paetzold and Lucia Specia. 2016. Benchmarking lexical simplification systems. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 3074–3080, Portorož, Slovenia. European Language Resources Association (ELRA).

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Piotr Przybyła and Matthew Shardlow. 2020. Multi-Word Lexical Simplification. In *Proceedings of the 28th International Conference on Computational Linguistics (COLING 2020)*, pages 1435–1446, Barcelona, Spain. International Committee on Computational Linguistics.

Horacio Saggion, Sanja Štajner, Daniel Ferrés, Kim Cheng Sheang, Matthew Shardlow, Kai North, and Marcos Zampieri. 2022. Findings of the TSAR-2022 shared task on multilingual lexical simplification. In *Proceedings of the Workshop on Text Simplification, Accessibility, and Readability (TSAR-2022)*, pages 271–283, Abu Dhabi, United Arab Emirates (Virtual). Association for Computational Linguistics.

Matthew Shardlow. 2013. The CW corpus: A new resource for evaluating the identification of complex words. In *Proceedings of the Second Workshop on Predicting and Improving Text Readability for Target Reader Populations*, pages 69–77, Sofia, Bulgaria. Association for Computational Linguistics.

Matthew Shardlow. 2014. Out in the open: Finding and categorising errors in the lexical simplification pipeline. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 1583–1590.

Neha Srikanth and Junyi Jessy Li. 2020. Elaborative simplification: Content addition and explanation generation in text simplification.

Sanja Stajner, Biljana Drndarevic, and Horacio Saggion. 2013. Corpus-based sentence deletion and split decisions for spanish text simplification. *Computacion y Sistemas. 2013; 17 (2): 251-62.*

Sanja Štajner, Ruslan Mitkov, and Horacio Saggion. 2014. One step closer to automatic evaluation of text simplification systems. In *Proceedings of the 3rd Workshop on Predicting and Improving Text Readability for Target Reader Populations (PITR)*, pages 1–10, Gothenburg, Sweden. Association for Computational Linguistics.

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

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. 2016. Optimizing statistical machine translation for text simplification. *Transactions of the Association for Computational Linguistics*, 4:401–415.

Mark Yatskar, Bo Pang, Cristian Danescu-Niculescu-Mizil, and Lillian Lee. 2010. For the sake of simplicity: Unsupervised extraction of lexical simplifications from Wikipedia. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 365–368, Los Angeles, California. Association for Computational Linguistics.

Seid Muhie Yimam, Sanja Štajner, Martin Riedl, and Chris Biemann. 2017. CWIG3G2 - complex word identification task across three text genres and two user groups. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 401–407, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Zhemin Zhu, Delphine Bernhard, and Iryna Gurevych. 2010. A monolingual tree-based translation model for sentence simplification. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 1353–1361, Beijing, China. Coling 2010 Organizing Committee.

Sanja Štajner and Goran Glavaš. 2017. Leveraging event-based semantics for automated text simplification. *Expert Systems with Applications*, 82:383–395.