A Word-Complexity Lexicon and A Neural Readability Ranking Model for Lexical Simplification

Mounica Maddela and **Wei Xu**
Department of Computer Science and Engineering
The Ohio State University
{maddela.4, xu.1265}@osu.edu

Abstract

Current lexical simplification approaches rely heavily on heuristics and corpus level features that do not always align with human judgment. We create a human-rated word-complexity lexicon of 15,000 English words and propose a novel neural readability ranking model with a Gaussian-based feature vectorization layer that utilizes these human ratings to measure the complexity of any given word or phrase. Our model performs better than the state-of-the-art systems for different lexical simplification tasks and evaluation datasets. Additionally, we also produce SimplePPDB++, a lexical resource of over 10 million simplifying paraphrase rules, by applying our model to the Paraphrase Database (PPDB).[1]

1 Introduction

Lexical simplification is an important subfield that is concerned with the complexity of words or phrases, and particularly how to measure readability and reduce the complexity using alternative paraphrases. There are three major lexical simplification tasks which effectively resemble a pipeline: (i) Complex Word Identification (Paetzold and Specia, 2016a; Yimam et al., 2017; Shardlow, 2013b) which involves identifying complex words in the sentence; (ii) Substitution Generation (Glavaš and Štajner, 2015; Coster and Kauchak, 2011) which involves finding alternatives to complex words or phrases; and (iii) Substitution Ranking (Specia et al., 2012) which involves ranking the paraphrases by simplicity. Lexical simplification also has practical real-world uses, such as displaying alternative expressions of complex words as reading assistance for children (Kajiwara et al., 2013), non-native speakers (Petersen and Ostendorf, 2007; Pellow and Eskenazi, 2014), lay readers (Elhadad and Sutaria, 2007; Siddharthan and Katsos, 2010), or people with reading disabilities (Rello et al., 2013).

Most current approaches to lexical simplification heavily rely on corpus statistics and surface level features, such as word length and corpus-based word frequencies (read more in §5). Two of the most commonly used assumptions are that simple words are associated with shorter lengths and higher frequencies in a corpus. However, these assumptions are not always accurate and are often the major source of errors in the simplification pipeline (Shardlow, 2014). For instance, the word *foolishness* is simpler than its meaning-preserving substitution *folly* even though *foolishness* is longer and less frequent in the Google 1T Ngram corpus (Brants and Franz, 2006). In fact, we found that 21% of the 2272 meaning-equivalent word pairs randomly sampled from PPDB[2] (Ganitkevitch et al., 2013) had the simpler word longer than the complex word, while 14% had the simpler word less frequent.

To alleviate these inevitable shortcomings of corpus and surface-based methods, we explore a simple but surprisingly unexplored idea – creating an English lexicon of 15,000 words with word-complexity ratings by humans. We also propose a new neural readability ranking model with a Gaussian-based feature vectorization layer, which can effectively exploit these human ratings as well as other numerical features to measure the complexity of any given word or phrase (including those outside the lexicon and/or with sentential context). Our model significantly outperforms the state-of-the-art on the benchmark SemEval-2012 evaluation for Substitution Ranking (Specia et al.,

[1]The code and data are publicly available on the authors' homepages and GitHub: https://github.com/mounicam/lexical_simplification.

[2]PPDB is a large paraphrase database derived from static bilingual translation data available at: http://paraphrase.org

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3749–3760
Brussels, Belgium, October 31 - November 4, 2018.

2012; Paetzold and Specia, 2017), with or without using the manually created word-complexity lexicon, achieving a Pearson correlation of 0.714 and 0.702 respectively. We also apply the new ranking model to identify lexical simplifications (e.g., *commemorate* → *celebrate*) among the large number of paraphrase rules in PPDB with improved accuracy compared to previous work for Substitution Generation. At last, by utilizing the word-complexity lexicon, we establish a new state-of-the-art on two common test sets for Complex Word Identification (Paetzold and Specia, 2016a; Yimam et al., 2017). We make our code, the word-complexity lexicon, and a lexical resource of over 10 million paraphrase rules with improved readability scores (namely SimplePPDB++) all publicly available.

2 Constructing A Word-Complexity Lexicon with Human Judgments

We first constructed a lexicon of 15,000 English words with word-complexity scores assessed by human annotators.[3] Despite the actual larger English vocabulary size, we found that rating the most frequent 15,000 English words in Google 1T Ngram Corpus[4] was effective for simplification purposes (see experiments in §4) as our neural ranking model (§3) can estimate the complexity of any word or phrase even out-of-vocabulary.

We asked 11 non-native but fluent English speakers to rate words on a 6-point Likert scale. We found that an even number 6-point scale worked better than a 5-point scale in a pilot experiment with two annotators, as the 6-point scheme allowed annotators to take a natural two-step approach: first determine whether a word is simple or complex; then decide whether it is 'very simple' (or 'very complex'), 'simple' (or 'complex'), or 'moderately simple' (or 'moderately complex'). For words with multiple capitalized versions (e.g., *nature*, *Nature*, *NATURE*), we displayed the most frequent form to the annotators. We also asked the annotators to indicate the words for which they had trouble assessing their complexity due to ambiguity, lack of context or any other reason. All the annotators reported little difficulty, and explained possible reasons such as that word *bug* is simple regardless of its meaning as an *insect* in biology or an *error* in computer software.[5]

Word	Avg	A1	A2	A3	A4	A5
watch	1.0	1	1	1	1	1
muscles	1.6	2	1	2	2	1
sweatshirts	1.8	2	1	2	3	1
giant	2.0	2	3	1	1	3
pattern	2.4	2	3	2	3	2
Christianity	2.8	3	2	2	3	4
educational	3.2	3	3	3	3	4
revenue	3.6	4	4	3	3	4
cortex	4.2	4	4	4	4	5
crescent	4.6	5	5	5	5	3
Memorabilia	5.4	5	6	6	5	5
assay	5.8	6	6	6	5	6

Table 1: Word-Complexity lexicon consists of English words and their complexity scores obtained by averaging over human ratings. A1, A2, A3, A4 and A5 are ratings by five different annotators on a 6-point Likert scale (1 is the simplest and 6 is the most complex).

With our hired annotators, we were able to have most annotators complete half or the full list of 15,000 words for better consistency, and collected between 5 and 7 ratings for each word. It took most annotators about 2 to 2.5 hours to rate 1,000 words. Table 1 shows few examples from the lexicon along with their human ratings.

In order to assess the annotation quality, we computed the Pearson correlation between each annotator's annotations and the average of the rest of the annotations (Agirre et al., 2014). For our final word-complexity lexicon, we took an average of the human ratings for each word, discarding those (about 3%) that had a difference ≥ 2 from the mean of the rest of the ratings. The overall inter-annotator agreement improved from 0.55 to 0.64 after discarding the outlying ratings. For the majority of the disagreements, the ratings of one annotator and the mean of the rest were fairly close: the difference is ≤ 0.5 for 47% of the annotations; ≤ 1.0 for 78% of the annotations; and ≤ 1.5 for 93% of the annotations on the 6-point scale. We hired annotators of different native languages intentionally, which may have contributed to the variance in the judgments.[6] We leave further investigation and possible crowdsourcing annotation to future work.

[3]Download at https://github.com/mounicam/lexical_simplification

[4]https://catalog.ldc.upenn.edu/ldc2006t13

[5]The word-happiness lexicon (Dodds et al., 2011) of 10,222 words was also similarly created by human rating on the most frequent words without context or word-sense disambiguation.

[6]One recent work similarly observed lower inter-annotator agreement among non-native speakers than native speakers when asked to identify complex words in given text paragraphs (Yimam et al., 2017).

3 Neural Readability Ranking Model for Words and Phrases

In order to predict the complexity of any given word or phrase, within or outside the lexicon, we propose a Neural Readability Ranking model that can leverage the created word-complexity lexicon and take context (if available) into account to further improve performance. Our model uses a Gaussian-based vectorization layer to exploit numerical features more effectively and can outperform the state-of-the-art approaches on multiple lexical simplification tasks with or without the word-complexity lexicon. We describe the general model framework in this section, and task-specific configurations in the experiment section (§4).

3.1 Neural Readability Ranker (NRR)

Given a pair of words/phrases $\langle w_a, w_b \rangle$ as input, our model aims to output a real number that indicates the relative complexity $P(y|\langle w_a, w_b \rangle)$ of w_a and w_b. If the output value is negative, then w_a is simpler than w_b and vice versa. Figure 1 shows the general architecture of our ranking model highlighting the three main components:

1. An input **feature extraction layer** (§3.2) that creates lexical and corpus-based features for each input $f(w_a)$ and $f(w_b)$, and pairwise features $f(\langle w_a, w_b \rangle)$. We also inject the word-complexity lexicon into the model as a numerical feature plus a binary indicator.

2. A **Gaussian-based feature vectorization layer** (§3.3) that converts each numerical feature, such as the lexicon scores and n-gram probabilities, into a vector representation by a series of Gaussian radial basis functions.

3. A feedforward neural network performing regression with one **task-specific output node** that adapts the model to different lexical simplification tasks (§4).

Our model first processes each input word or phrase in parallel, producing vectorized features. All the features are then fed into a joint feedforward neural network.

3.2 Features

We use a combination of rating scores from the word-complexity lexicon, lexical and corpus features (Pavlick and Callison-Burch, 2016) and collocational features (Paetzold and Specia, 2017).

We inject the word-complexity lexicon into the NRR model by adding two features for each input word or phrase: a 0-1 binary feature representing the presence of a word (the longest word in a multi-word phrase) in the lexicon, and the corresponding word complexity score. For out-of-vocabulary words, both features have the value 0. We back-off to the complexity score of the lemmatized word if applicable. We also extract the following features: phrase length in terms of words and characters, number of syllables, frequency with respect to Google Ngram corpus (Brants and Franz, 2006), the relative frequency in Simple Wikipedia with respect to normal Wikipedia (Pavlick and Nenkova, 2015) and ngram probabilities from a 5-gram language model trained on the SubIMDB corpus (Paetzold and Specia, 2016c), which has been shown to work well for lexical simplification. For a word w, we take language model probabilities of all the possible n-grams within the context window of 2 to the left and right of w. When w is a multi-word phrase, we break w into possible n-grams and average the probabilities for a specific context window.

For an input pair of words/phrases $\langle w_a, w_b \rangle$, we include individual features $f(w_1)$, $f(w_2)$ and the differences $f(w_a) - f(w_b)$. We also use pairwise features $f(\langle w_a, w_b \rangle)$ including cosine similarity $cos(\overrightarrow{w}_a, \overrightarrow{w}_b)$ and the difference $\overrightarrow{w}_a - \overrightarrow{w}_b$ between the word2vec (Mikolov et al., 2013) embedding of the input words. The embeddings for a mutli-word phrase are obtained by averaging the embeddings of all the words in the phrase. We use the 300-dimensional embeddings pretrained on the Google News corpus, which is released as part of the word2vec package.[7]

3.3 Vectorizing Numerical Features via Gaussian Binning

Our model relies primarily on numerical features as many previous approaches for lexical simplification. Although these continuous features can be directly fed into the network, it is helpful to exploit fully the nuanced relatedness between different intervals of feature values.

We adopt a smooth binning approach and project each numerical feature into a vector representation by applying multiple Gaussian radial basis functions. For each feature f, we divide its

[7] https://code.google.com/archive/p/word2vec/

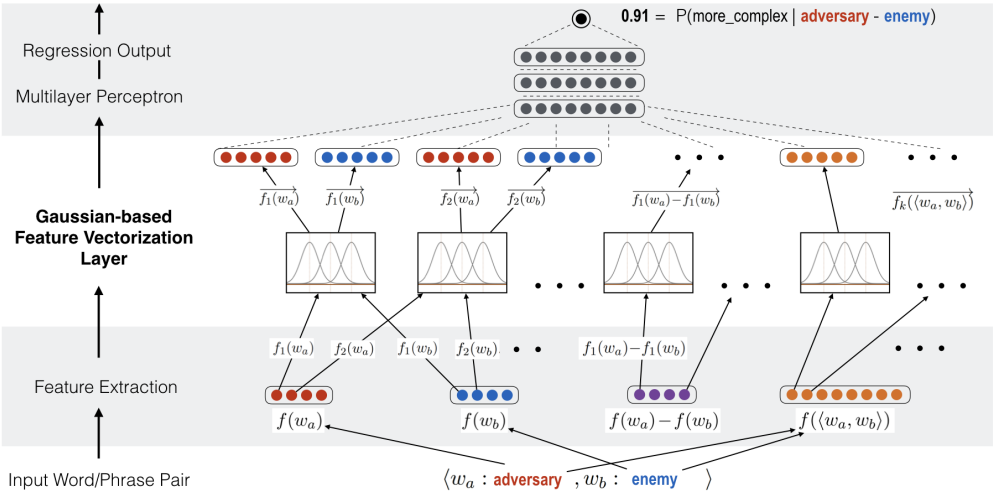

Figure 1: The neural readability ranking (NRR) model.

value range $[f_{min}, f_{max}]$ evenly into k bins and place a Gaussian function for each bin with the mean μ_j ($j \in \{1, 2, \ldots, k\}$) at the center of the bin and standard deviation σ. We specify σ as a fraction γ of bin width:

$$\sigma = \frac{1}{k}(f_{max} - f_{min}) \cdot \gamma \quad (1)$$

where γ is a tunable hyperparameter in the model. For a given feature value $f(\cdot)$, we then compute the distance to each bin as follows:

$$d_j(f(\cdot)) = e^{-\frac{(f(\cdot)-\mu_j)^2}{2\sigma^2}} \quad (2)$$

and normalize to project into a k-dimensional vector $\overrightarrow{f(\cdot)} = (d_1, d_2, \ldots, d_k)$.

We vectorize all the features except word2vec vectors, $\overrightarrow{f(w_a)}$, $\overrightarrow{f(w_b)}$, $\overrightarrow{f(w_a)-f(w_b)}$, and $\overrightarrow{f(\langle w_a, w_b \rangle)}$, then concatenate them as inputs. Figure 2 presents a motivating t-SNE visualization of the word-complexity scores from the lexicon after the vectorization in our NRR model, where different feature value ranges are gathered together with some distances in between.

3.4 Training and Implementation Details

We use PyTorch framework to implement the NRR model, which consists of an input layer, three hidden layers with eight nodes in each layer and the *tanh* activation function, and a single node linear output layer. The training objective is to minimize the Mean Squared Error (MSE):

$$L(\theta) = \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2 \quad (3)$$

where y_i and $\hat{y}_i$ are the true and predicted relative complexity scores of $\langle w_a, w_b \rangle$ which can be configured accordingly for different lexical simplification tasks and datasets, m is the number of training examples, and θ is the set of parameters of the NRR model. We use Adam algorithm (Kingma and Ba, 2014) for optimization and also apply a dropout of 0.2 to prevent overfitting. We set the rate to 0.0005 and 0.001 for experiments in (§4.1) and (§4.2) respectively. For Gaussian binning layer, we set the number of bins k to 10 and γ to 0.2 without extensive parameter tuning. For each experiment,we report results with 100 epochs.

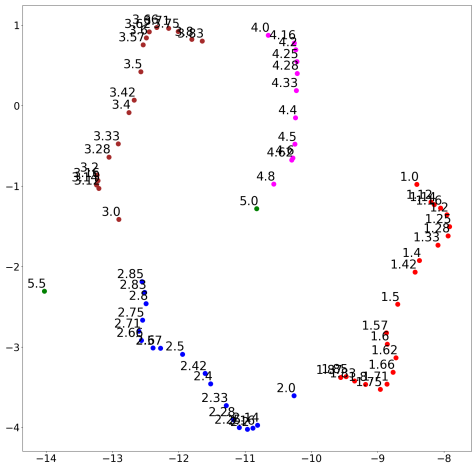

Figure 2: t-SNE visualization of the complexity scores, ranging between 1.0 and 5.0, of 300 random words from the word-complexity lexicon vectorized into 10-dimensional representations by applying Gaussian radial basis functions.

4 Lexical Simplification Applications

As the lexical simplification research field traditionally studies multiple sub-tasks and datasets, we present a series of experiments to demonstrate the effectiveness of our newly created lexicon and neural readability ranking (NRR) model.

4.1 Substitution Ranking

Given an instance consisting of a target complex word in a sentence and a set of candidate substitutions, the goal of the Substitution Ranking task is to rank the candidates in the order of their simplicity. In this section, we show that our proposed NRR model outperforms the state-of-the-art neural model on this task, with or without using the word-complexity lexicon.

Data. We use the dataset from the English Lexical Simplification shared-task at SemEval 2012 (Specia et al., 2012) for evaluation. The training and test sets consist of 300 and 1,710 instances, respectively, with a total of 201 target words (all single word, mostly polysemous) and each in 10 different sentences. One example of such instance contains a target complex word in context:

When you think about it, that's pretty terrible.

and a set of candidate substitutions {*bad*, *awful*, *deplorable*}. Each instance contains at least 2 and an average of 5 candidates to be ranked. There are a total of 10034 candidates in the dataset, 88.5% of which are covered by our word-complexity lexicon and 9.9% are multi-word phrases (3438 unique candidates with 81.8% in-vocabulary and 20.2% multi-word).

Task-specific setup of the NRR model. We train the NRR model with every pair of candidates $\langle c_a, c_b \rangle$ in a candidate set as the input, and the difference of their ranks $r_a - r_b$ as the ground-truth label. For each such pair, we also include another training instance with $\langle c_b, c_a \rangle$ as the input and $r_b - r_a$ as the label. Given a test instance with candidate set C, we rank the candidates as follows: for every pair of candidates $\langle c_a, c_b \rangle$, the model predicts the relative complexity score $S(c_a, c_b)$; we then compute a single score $R(c_a) = \sum_{c_a \neq c_b \in C} S(c_a, c_b)$ for each candidate by aggregating pairwise scores and rank the candidates in the increasing order of these scores.

Comparison to existing methods. We compare with the state-of-the-art neural model (Paetzold and Specia, 2017) for substitution ranking with the best reported results on the SemEval 2012 dataset. Our baselines also include several other existing methods: Biran et al. (2011), Kajiwara et al. (2013), and Glavaš & Štajner (2015), which use carefully designed heuristic scoring functions to combine various information such as corpus statistics and semantic similarity measures from WordNet; Horn et al. (2014) and the Boundary Ranker (Paetzold and Specia, 2015), which respectively use a supervised SVM ranking model and pairwise linear classification model with various features. All of these methods have been implemented as part of the LEXenstein toolkit (Paetzold and Specia, 2015), which we use for the experimental comparisons here. In addition, we also compare to the best system (Jauhar and Specia, 2012) among participants at SemEval 2012, which used SVM-based ranking.

	P@1	Pearson
Biran et al. (2011)	51.3	0.505
Jauhar & Specia (2012)	60.2	0.575
Kajiwara et al. (2013)	60.4	0.649
Horn et al. (2014)	63.9	0.673
Glavaš & Štajner (2015)	63.2	0.644
Boundary Ranker	65.3	0.677
Paetzold & Specia (2017)	65.6	0.679
NRR_{all}	65.4	0.682
$\text{NRR}_{all+binning}$	66.6	0.702*
$\text{NRR}_{all+binning+WC}$	**67.3***	**0.714***

Table 2: Substitution Ranking evaluation on English Lexical Simplification shared-task of SemEval 2012. P@1 and Pearson correlation of our neural readability ranking (NRR) model compared to the state-of-the-art neural model (Paetzold and Specia, 2017) and other methods. $*$ indicates statistical significance ($p < 0.05$) compared to the best performing baseline (Paetzold and Specia, 2017).

Results. Table 2 compares the performances of our NRR model to the state-of-the-art results reported by Paetzold and Specia (2017). We use precision of the simplest candidate (P@1) and Pearson correlation to measure performance. P@1 is equivalent to TRank (Specia et al., 2012), the official metric for the SemEval 2012 English Lexical Simplification task. While P@1 captures the practical utility of an approach, Pearson correlation indicates how well the system's rankings correlate with human judgment. We train our NRR model with all the features (NRR_{all}) mentioned in §3.2 except the word2vec embedding features to avoid overfitting on the small training set. Our full model ($\text{NRR}_{all+binning+WC}$) exhibits a statistically significant improvement over the state-of-

paraphrases of ***'modification'*** ranked by simplicity	
SimplePPDB	*tweak, modify, process, variable, layout*
SimplePPDB++	*change, adjustment, amendment, shift, difference*
paraphrases of ***'aggregation'***	
SimplePPDB	*pod, swarm, node, clump, pool*
SimplePPDB++	*cluster, pool, collection, addition, grouping*
paraphrases of ***'of transnational corporation'***	
SimplePPDB	*of corporation, by corporation, of enterprise, of tncs, of business*
SimplePPDB++	*of business, of firm, of corporation, of company, of enterprise*
paraphrases of ***'should reject'***	
SimplePPDB	*refuse, discard, repudiate, shun, dismiss*
SimplePPDB++	*vote against, set aside, throw out, say no to, turn away*

Table 3: SimplePPDB++ includes lexical and phrasal paraphrases with improved readability ranking scores by our $\text{NRR}_{all+binning+WC}$ model. Shown are the top 5 ranked simplifications according to SimplePPDB++ for several input words/phrases, in comparison to the previous work of SimplePPDB (Pavlick and Callison-Burch, 2016).

	Acc.	$\mathbf{P}_{+1}$	$\mathbf{P}_{-1}$
Google Ngram Frequency	49.4	53.7	54.0
Number of Syllables	50.1	53.8	53.3
Character & Word Length	56.2	55.7	56.1
W2V	60.4	54.9	53.1
SimplePPDB	62.1	57.6	57.8
NRR_{all}	59.4	61.8	57.7
$\text{NRR}_{all+binning}$	64.1	62.1	59.8
$\text{NRR}_{all+binning+WC}$	**65.3***	**65.0***	**61.8***

Table 4: Cross-validation accuracy and precision of our neural readability ranking (NRR) model used to create SimplePPDB++, in comparison to the SimplePPDB and other baselines. P_{+1} stands for the precision of 'simplifying' paraphrase rules and P_{-1} for the precision of 'complicating' rules. * indicates statistical significance ($p < 0.05$) compared to the best performing baseline (Pavlick and Callison-Burch, 2016).

the-art for both measures. We use paired bootstrap test (Berg-Kirkpatrick et al., 2012; Efron and Tibshirani, 1993) as it can be applied to any performance metric. We also conducted ablation experiments to show the effectiveness of the Gaussian-based feature vectorization layer ($_{+binning}$) and the word-complexity lexicon ($_{+WC}$).

4.2 SimplePPDB++

We also can apply our NRR model to rank the lexical and phrasal paraphrase rules in the Paraphrase Database (PPDB) (Pavlick et al., 2015), and identify good simplifications (see examples in Table 3). The resulting lexical resource, SimplePPDB++, contains all 13.1 million lexical and phrasal paraphrase rules in the XL version of PPDB 2.0 with readability scores in 'simplifying', 'complicating', or 'nonsense/no-difference' categories, allowing flexible trade-off between high-quality and high-coverage paraphrases. In this section, we show the effectiveness of the NRR model we used to create SimplePPDB++ by comparing with the previous version of SimplePPDB (Pavlick and Callison-Burch, 2016) which used a three-way logistic regression classifier. In next section, we demonstrate the utility of SimplePPDB++ for the Substitution Generation task.

Task-specific setup of NRR model. We use the same manually labeled data of 11,829 paraphrase rules as SimplePPDB for training and testing, of which 26.5% labeled as 'simplifying', 26.5% as 'complicating', and 47% as 'nonsense/no-difference'. We adapt our NRR model to perform the three-way classification by treating it as a regression problem. During training, we specify the ground truth label as follows: y = -1 if the paraphrase rule belongs to the 'complicating' class, y = +1 if the rule belongs to the 'simplifying'class, and y = 0 otherwise. For predicting, the network produces a single real-value output $\hat{y} \in [-1, 1]$ which is then mapped to three-class labels based on the value ranges for evaluation. The thresholds for the value ranges are -0.4 and 0.4 chosen by cross-validation.

Comparison to existing methods. We compare our neural readability ranking (NRR) model used to create the SimplePPDB++ against SimplePPDB, which uses a multi-class logistic regression model. We also use several other baselines, including W2V which uses logistic regression with only word2vec embedding features.

Results. Following the evaluation setup in previous work (Pavlick and Callison-Burch, 2016), we compare accuracy and precision by 10-fold cross-validation. Folds are constructed in such a way that the training and test vocabularies are disjoint. Table 4 shows the performance of our model compared to SimplePPDB and other baselines. We use all the features (NRR_{all}) in §3.2 except for the context features as we are classifying paraphrase rules in PPDB that come with no context. SimplePPDB used the same features plus additional discrete features, such as POS tags, character unigrams and bigrams. Our neural readability ranking model alone with Gaussian binning ($\text{NRR}_{all+binning}$) achieves better accuracy and precision while using less features. Leverag-

ing the lexicon ($NRR_{all+binning+WC}$) shows statistically significant improvements over SimplePPDB rankings based on the paired bootstrap test. The accuracy increases by 3.2 points, the precision for 'simplifying' class improves by 7.4 points and the precision for 'complicating' class improves by 4.0 points.

4.3 Substitution Generation

Substitution Generation is arguably the most challenging research problem in lexical simplification, which involves producing candidate substitutions for each target complex word/phrase, followed by the substitution ranking. The key focus is to not only have better rankings, but more importantly, to have a larger number of simplifying substitutions generated. This is a more realistic evaluation to demonstrate the utility of SimplePPDB++ and the effectiveness of the NRR ranking model we used to create it, and how likely such lexical resources can benefit developing end-to-end sentence simplification system (Narayan and Gardent, 2016; Zhang and Lapata, 2017) in future work.

Data. We use the dataset from (Pavlick and Callison-Burch, 2016), which contains 100 unique target words/phrases sampled from the Newsela Simplification Corpus (Xu et al., 2015) of news articles, and follow the same evaluation procedure. We ask two annotators to evaluate whether the generated substitutions are good simplifications.

Comparison to existing methods. We evaluate the correctness of the substitutions generated by SimplePPDB++ in comparison to several existing methods: Glavaš (Glavaš and Štajner, 2015), Kauchak (Coster and Kauchak, 2011), WordNet Generator (Devlin and Tait, 1998; Carroll et al., 1999), and SimplePPDB (Pavlick and Callison-Burch, 2016). Glavaš obtains candidates with the highest similarity scores in the GloVe (Pennington et al., 2014) word vector space. Kauchak's generator is based on Simple Wikipedia and normal Wikipedia parallel corpus and automatic word alignment. WordNet-based generator simply uses the synonyms of word in WordNet (Miller, 1995). For all the existing methods, we report the results based on the implementations in (Pavlick and Callison-Burch, 2016), which used SVM-based ranking. For both SimplePPDB and SimplePPDB++, extracted candidates are high quality paraphrase rules (quality score $\geq$3.5 for words and $\geq$4.0 for phrases) belonging to the same syntactic category as target word according to PPDB 2.0 (Pavlick et al., 2015).

	#PPs	MAP	P@1
Glavaš$_{(n=95)}$	—	22.8	13.5
WordNet$_{(n=82)}$	6.63	62.2	50.6
Kauchak$_{(n=48)}$	4.39	**76.4**†	68.9
SimplePPDB$_{(n=100)}$	8.77	67.8	78.0
SimplePPDB++$_{(n=100)}$	**9.52**	69.1	**80.2**

Table 5: Substitution Generation evaluation with Mean Average Precision, Precision@1 and the average number of paraphrases generated per target for each method. n is the number of target complex words/phrases for which the model generated > 0 candidates. Kauchak† has an advantage on MAP because it generates the least number of candidates. Glavaš is marked as '-' because it can technically generate as many words/phrases as are in the vocabulary.

Results. Table 5 shows the comparison of SimplePPDB and SimplePPDB++ on the number of substitutions generated for each target, the mean average precision and precision@1 for the final ranked list of candidate substitutions. This is a fair and direct comparison between SimplePPDB++ and SimplePPDB, as both methods have access to the same paraphrase rules in PPDB as potential candidates. The better NRR model we used in creating SimplePPDB++ allows improved selections and rankings of simplifying paraphrase rules than the previous version of SimplePPDB. As an additional reference, we also include the measurements for the other existing methods based on (Pavlick and Callison-Burch, 2016), which, by evaluation design, are focused on the comparison of precision while PPDB has full coverage.

4.4 Complex Word Identification

Complex Word Identification (CWI) identifies the difficult words in a sentence that need to be simplified. According to Shardlow (2014), this step can improve the simplification system by avoiding mistakes such as overlooking challenging words or oversimplifying simple words. In this section, we demonstrate how our word-complexity lexicon helps with the CWI task by injecting human ratings into the state-of-the-art systems.

Data. The task is to predict whether a target word/phrase in a sentence is 'simple' or 'complex', and an example instance is as follows:

Nine people were killed in the bombardment.

We conduct experiments on two datasets: (i) Semeval 2016 CWI shared-task dataset (Paetzold

	CWI SemEval 2016			CWIG3G2 2018		
	G-score	**F-score**	**Accuracy**	**G-score**	**F-score**	**Accuracy**
Length	47.8	10.7	33.2	70.8	65.9	67.7
Senses	57.9	12.5	43.6	67.7	62.3	54.1
SimpleWiki	69.7	16.2	58.3	73.1	66.3	61.6
NearestCentroid	66.1	14.8	53.6	75.1	66.6	76.7
SV000gg	77.3	24.3	77.6	74.9	73.8	78.7
WC-only	68.5	**30.5**	**87.7**	71.1	67.5	69.8
NearestCentroid$_{+WC}$	70.2	16.6	61.8	**77.3**	68.8	78.1
SV000gg$_{+WC}$	**78.1**	26.3	80.0	75.4	**74.8**	**80.2**

Table 6: Evaluation on two datasets for English complex word identification. Our approaches that utilize the word-complexity lexicon (WC) improve upon the nearest centroid (Yimam et al., 2017) and SV000gg (Paetzold and Specia, 2016b) systems. The best performance figure of each column is denoted in **bold** typeface and the second best is denoted by an underline.

CWI SemEval 2016	total (IV%)	unique (IV%)
simple	85621 (94.7%)	14129 (77.6%)
complex	4837 (57.4%)	3836 (54.8%)
CWIG3G2	total (IV%)	unique (IV%)
simple	20451 (89.8%)	5576 (82.1%)
complex	14428 (81.1%)	8376 (76.0%)

Table 7: Statistics of CWI datasets – total number of target words/phrases, number of unique targets, and in-vocabulary (IV) ratio with respect to our word-complexity lexicon.

and Specia, 2016a), which has been widely used for evaluating CWI systems and contains 2,237 training and 88,221 test instances from Wikipedia; and (ii) CWIG3G2 dataset (Yimam et al., 2017), which is also known as English monolingual CWI 2018 shared-task dataset (Yimam et al., 2018) and comprises of 27,299 training, 3,328 development and 4,252 test instances from Wikipedia and news articles. Table 7 shows the coverage of our word-complexity lexicon over the two CWI datasets.

Comparison to existing methods. We consider two state-of-the-art CWI systems: (i) the nearest centroid classifier proposed in (Yimam et al., 2017), which uses phrase length, number of senses, POS tags, word2vec cosine similarities, n-gram frequency in Simple Wikipedia corpus and Google 1T corpus as features; and (ii) SV000gg (Paetzold and Specia, 2016b) which is an ensemble of binary classifiers trained with a combination of lexical, morphological, collocational, and semantic features. The latter is the best performing system on the Semeval 2016 CWI dataset. We also compare to threshold-based baselines that use word length, number of word senses and frequency in the Simple Wikipedia.

Utilizing the word-complexity lexicon. We enhance the SV000gg and the nearest centroid classifier by incorporating the word-complexity lexicon as additional features as described in §3.2. We added our modifications to the implementation of SV000gg in the LEXenstein toolkit, and used our own implementation for the nearest centroid classifier. Additionally, to evaluate the word-complexity lexicon in isolation, we train a decision tree classifier with only human ratings as input (WC-only), which is equivalent to learning a threshold over the human ratings.

Results. We compare our enhanced approaches (SV000gg$_{+WC}$ and NC$_{+WC}$) and lexicon only approach (WC-only), with the state-of-the-art and baseline threshold-based methods. For measuring performance, we use F-score and accuracy as well as G-score, the harmonic mean of accuracy and recall. G-score is the official metric of the CWI task of Semeval 2016. Table 6 shows that the word-complexity lexicon improves the performance of SV000gg and the nearest centroid classifier in all the three metrics. The improvements are statistically significant according to the paired bootstrap test with $p < 0.01$. The word-complexity lexicon alone (WC-only) performs satisfactorily on the CWIG3G2 dataset, which effectively is a simple table look-up approach with extreme time and space efficiency. For CWI SemEval 2016 dataset, WC-only approach gives the best accuracy and F-score, though this can be attributed to the skewed distribution of dataset (only 5% of the test instances are 'complex').

5 Related Work

Lexical simplification: Prior work on lexical simplification depends on lexical and corpus-based features to assess word complexity. For complex word identification, there are broadly two lines of research: learning a frequency-based threshold over a large corpus (Shardlow, 2013b) or training an ensemble of classifiers over a combination of lexical and language model features

(Shardlow, 2013a; Paetzold and Specia, 2016a; Yimam et al., 2017; Kriz et al., 2018). Substitution ranking also follows similar trend. Biran et al. (2011) and Bott et al. (2012) employed simplicity measures based on word length and word frequencies from Wikipedia and Simple Wikipedia. Kajiwara et al. (2013) combined WordNet similarity measures with Simple Wikipedia frequencies. Glavaš and Štajner (2015) averaged the rankings produced by a collection of frequency, language model and semantic similarity features. Horn et al. (2014) trained an SVM classifier over corpus-based features.

Only recently, researchers started to apply neural networks to simplification tasks. To the best of our knowledge, the work by Paetzold and Specia (2017) is the first neural model for lexical simplification which uses a feedforward network with language model probability features. Our NRR model is the first pairwise neural ranking model to vectorize numeric features and to embed human judgments using a word-complexity lexicon of 15,000 English words.

Besides lexical simplification, another line of relevant research is sentence simplification that uses statistical or neural machine translation (MT) approaches (Xu et al., 2016; Nisioi et al., 2017; Zhang and Lapata, 2017; Vu et al., 2018; Guo et al., 2018). It has shown possible to integrate paraphrase rules in PPDB into statistical MT for sentence simplification (Xu et al., 2016) and bilingual translation (Mehdizadeh Seraj et al., 2015), while how to inject SimplePPDB++ into neural MT remains an open research question.

Lexica for simplification: There have been previous attempts to use manually created lexica for simplification. For example, Elhadad and Sutaria (2007) used UMLS lexicon (Bodenreider, 2007), a repository of technical medical terms; Ehara et al. (2010) asked non-native speakers to answer multiple-choice questions corresponding to 12,000 English words to study each user's familiarity of vocabulary; Kaji et al. (2012) and Kajiwara et al. (2013) used a dictionary of 5,404 Japanese words based on the elementary school textbooks; Xu et al. (2016) used a list of 3,000 most common English words; Lee and Yeung (2018) used an ensemble of vocabulary lists of different complexity levels. However, to the best of our knowledge, there is no previous study on manually building a large word-complexity lexicon with human judgments that has shown substantial improvements on automatic simplification systems. We were encouraged by the success of the word-emotion lexicon (Mohammad and Turney, 2013) and the word-happiness lexicon (Dodds et al., 2011, 2015).

Vectorizing features: Feature binning is a standard feature engineering and data processing method to discretize continuous values, more commonly used in non-neural machine learning models. Our work is largely inspired by recent works on entity linking that discussed feature quantization for neural models (Sil et al., 2017; Liu et al., 2016) and neural dependency parsing with embeddings of POS tags as features (Chen and Manning, 2014).

6 Conclusion

We proposed a new neural readability ranking model and showed significant performance improvement over the state-of-the-art on various lexical simplification tasks. We release a manually constructed word-complexity lexicon of 15,000 English words and an automatically constructed lexical resource, SimplePPDB++, of over 10 million paraphrase rules with quality and simplicity ratings. For future work, we would like to extend our lexicon to cover specific domains, different target users and languages.

Acknowledgments

We thank anonymous reviewers for their thoughtful comments. We thank Avirup Sil and Anastasios Sidiropoulos for valuable discussions, Sanja Štajner and Seid Muhie Yimam for sharing their code and data. We also thank the annotators: Jeniya Tabassum, Ashutosh Baheti, Wuwei Lan, Fan Bai, Alexander Konovalov, Chaitanya Kulkarni, Shuaichen Chang, Jayavardhan Reddy, Abhishek Kumar and Shreejit Gangadharan.

This material is based on research sponsored by the NSF under grants IIS-1822754 and IIS-1755898. The views and conclusions contained in this publication are those of the authors and should not be interpreted as representing official policies or endorsements of the NSF or the U.S. Government.

References

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Wiebe Janyce. 2014. SemEval-2014 Task 10: Multilingual Semantic Textual Similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval@COLING)*.

Taylor Berg-Kirkpatrick, David Burkett, and Dan Klein. 2012. An Empirical Investigation of Statistical Significance in NLP. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

Or Biran, Samuel Brody, and Noemie Elhadad. 2011. Putting it Simply: A Context-Aware Approach to Lexical Simplification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT)*.

Olivier Bodenreider. 2007. The Unified Medical Language System (UMLS): integrating biomedical terminology. In *Proceedings of the Workshop on Biological, Translational, and Clinical Language Processing (BioNLP)*.

Stefan Bott, Luz Rello, Drndarvic Biljang, and Saiggon Horacio. 2012. Can Spanish Be Simpler? LexSiS: Lexical Simplification for Spanish. *Proceedings of the International Conference on Computational Linguistics (COLING)*.

Thorsten Brants and Alex Franz. 2006. Web 1T 5-gram version 1. *Linguistic Data Consortium (LDC)*.

John Carroll, Guido Minnen, Darren Pearce, Yvonne Canning, Siobhan Devlin, and John Tait. 1999. Simplifying Text for Language-Impaired Readers. In *Proceedings of the 9th Conference of European chapter of the Association for Computational Linguistics (EACL)*.

Danqi Chen and Christopher Manning. 2014. A Fast and Accurate Dependency Parser using Neural Networks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

William Coster and David Kauchak. 2011. Learning to Simplify Sentences using Wikipedia. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation (MTTG@ACL)*.

Siobhan Devlin and John Tait. 1998. The use of a psycholinguistic database in the simplification of text for aphasic readers. *Linguistic Databases*.

P. S. Dodds, E. M. Clark, S. Desu, M. R. Frank, A. J. Reagan, J. R. Williams, L. Mitchell, K. D. Harris, I. M. Kloumann, J. P. Bagrow, K. Megerdoomian, M. T. McMahon, B. F. Tivnan, and C. M. Danforth. 2015. Human language reveals a universal positivity bias. *Proceedings of the National Academy of Sciences (PNAS)*, 112(8):2389–2394.

Peter Sheridan Dodds, Kameron Decker Harris, Isabel M. Kloumann, Catherine A. Bliss, and Christopher M. Danforth. 2011. Temporal patterns of happiness and information in a global social network: Hedonometrics and Twitter. *Public Library of Science (PLOS ONE)*, 6(12):1–1.

Bradley Efron and Robert J. Tibshirani. 1993. *An Introduction to the Bootstrap*. Chapman & Hall/CRC.

Yo Ehara, Nobuyuki Shimizu, Takashi Ninomiya, and Hiroshi Nakagawa. 2010. Personalized reading support for second-language web documents by collective intelligence. In *Proceedings of the 15th International Conference on Intelligent User Interfaces (IUI)*.

Noemie Elhadad and Komal Sutaria. 2007. Mining a Lexicon of Technical Terms and Lay Equivalents. In *Proceedings of the Workshop on Biological, Translational, and Clinical Language Processing (BioNLP)*.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Goran Glavaš and Sanja Štajner. 2015. Simplifying Lexical Simplification: Do We Need Simplified Corpora? In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*.

Han Guo, Ramakanth Pasunuru, and Mohit Bansal. 2018. Dynamic multi-level multi-task learning for sentence simplification. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING)*.

Colby Horn, Cathryn Manduca, and David Kauchak. 2014. Learning a Lexical Simplifier Using Wikipedia. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*.

Sujay Kumar Jauhar and Lucia Specia. 2012. UOW-SHEF: Simplex - Lexical Simplicity Ranking based on Contextual and Psycholinguistic features. In *Proceedings of the 6th International Workshop on Semantic Evaluation (SemEval@NAACL)*.

Nobuhiro Kaji, Daisuke Kawahara, Sadao Kurohash, and Satoshi Sato. 2012. Verb Paraphrase based on Case Frame Alignment. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*.

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

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. In *Proceedings of the 3rd International Conference for Learning Representations (ICLR)*.

Reno Kriz, Eleni Miltsakaki, Marianna Apidianaki, and Chris Callison-Burch. 2018. Simplification using paraphrases and context-based lexical substitution. In *The 2018 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

John Lee and Chak Yan Yeung. 2018. Personalizing lexical simplification. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING)*.

Dan Liu, Wei Lin, Shiliang Zhang, Si Wei, and Hui Jiang. 2016. Neural Networks Models for Entity Discovery and Linking. *Computing Research Repository (CoRR)*, 1611.03558.

Ramtin Mehdizadeh Seraj, Maryam Siahbani, and Anoop Sarkar. 2015. Improving Statistical Machine Translation with a Multilingual Paraphrase Database. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *Proceedings of Workshop at International Conference on Learning Representations (ICLR)*.

George A. Miller. 1995. Wordnet: A Lexical Database for English. *Communications of the ACM*, 38(11):39–41.

Saif M. Mohammad and Peter D. Turney. 2013. Crowdsourcing a Word-Emotion Association Lexicon. *Computational Intelligence*, 29(3):436–465.

Shashi Narayan and Claire Gardent. 2016. Unsupervised Sentence Simplification Using Deep Semantics. In *Proceedings of the 9th International Conference on Natural Language Generation (INLG)*.

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 (ACL)*.

Gustavo Paetzold and Lucia Specia. 2015. LEXenstein: A Framework for Lexical Simplification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing: System Demonstrations (ACL-IJCNLP)*.

Gustavo Paetzold and Lucia Specia. 2016a. SemEval 2016 Task 11: Complex Word Identification. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval@NAACL)*.

Gustavo Paetzold and Lucia Specia. 2016b. SV000gg at Semeval-2016 Task 11: Heavy Gauge Complex Word Identification with System Voting. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval@NAACL)*.

Gustavo Paetzold and Lucia Specia. 2017. Lexical Simplification with Neural Ranking. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

Gustavo Henrique Paetzold and Lucia Specia. 2016c. Unsupervised Lexical Simplification for Non-Native Speakers. In *Proceedings of the 30th Association for the Advancement of Artificial Intelligence Conference on Artificial Intelligence (AAAI)*.

Ellie Pavlick and Chris Callison-Burch. 2016. Simple PPDB: A paraphrase database for simplification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Ellie Pavlick and Ani Nenkova. 2015. Inducing Lexical Style Properties for Paraphrase and Genre Differentiation. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Ellie Pavlick, Pushpendre Rastogi, Juri Ganitkevich, and Chris Callison-Burch Ben Van Durme. 2015. PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL)*.

David Pellow and Maxine Eskenazi. 2014. An Open Corpus of Everyday Documents for Simplification Tasks. In *Proceedings of the 3rd Workshop on Predicting and Improving Text Readability for Target Reader Populations (PITR@EACL)*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for Word Representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Sarah E Petersen and Mari Ostendorf. 2007. Text Simplification for Language Learners: A Corpus Analysis. In *In Proceedings of Workshop on Speech and Language Technology in Education (SLaTE)*.

Luz Rello, Ricardo Baeza-Yates, and Horacio Saggion. 2013. The Impact of Lexical Simplification by Verbal Paraphrases for People with and without Dyslexia. In *Proceedings of the 14th International Conference on Computational Linguistics and Intelligent Text Processing (CICLing)*.

Matthew Shardlow. 2013a. A Comparison of Techniques to Automatically Identify Complex Words. In *Proceedings of the ACL Student Research Workshop*.

Matthew Shardlow. 2013b. The CW Corpus: A New Resource for Evaluating the Identification of Complex Words. In *Proceedings of the 2nd Workshop on Predicting and Improving Text Readability for Target Reader Populations (PITR@ACL)*.

Matthew Shardlow. 2014. Out in the Open: Finding and Categorising Errors in the Lexical Simplification Pipeline. In *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC)*.

Advaith Siddharthan and Napoleon Katsos. 2010. Reformulating Discourse Connectives for Non-Expert Readers. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Avirup Sil, Gourab Kundu, Radu Florian, and Wael Hamza. 2017. Neural Cross-Lingual Entity Linking. In *Proceedings of the 30th Association for the Advancement of Artificial Intelligence Conference on Artificial Intelligence (AAAI)*.

Lucia Specia, Sujay Kumar Jauhar, and Rada Mihalcea. 2012. SemEval-2012 Task 1: English Lexical Simplification. In *Proceedings of the 6th International Workshop on Semantic Evaluation (SemEval@NAACL)*.

Tu Vu, Baotian Huu, Tsendsuren Munkhdalai, and Hong Yu. 2018. Sentence Simplification with Memory-Augmented Neural Networks. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

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 (TACL)*, 3:283–297.

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 (TACL)*, 4:401–415.

Seid Muhie Yimam, Chris Biemann, Shervin Malmasi, Gustavo H. Paetzold, Specia Lucia, Sanja tajner, Anas Tack, and Marcos Zampieri. 2018. A Report on the Complex Word Identification Shared Task 2018. In *Proceedings of the 13th Workshop on Innovative Use of NLP for Building Educational Applications (BEA@NAACL)*.

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 8th International Joint Conference on Natural Language Processing (IJCNLP)*.

Xingxing Zhang and Mirella Lapata. 2017. Sentence Simplification with Deep Reinforcement Learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.