Text; Its Generation, Simplification, and Complications

Mucun Tian
Michael Green
mucuntian@u.boisestate.edu
michaelgreen1@u.boisestate.edu

ABSTRACT

As Natural Language Processing becomes more wide spread and recognized, there are still plenty of unsolved problems and applications of their solutions. One of these obstacles is the automatic generation of text from a smaller sample of paragraphs, sentences, or words. Another is the simplification of text, in particular lexical simplification. This is the process of identifying and replacing complex words with simpler variations. We propose a combination of these two methods to generate simplified texts that could be used in a variety of circumstances. While the ultimate goal may not be with in the scope of this project, we hope to one day be able to combine these two complex fields in order to provide examples of a specific readability to help users learn the language, scaling between difficulties. However, for now we will focus on the simpler task of generating and simplifying our own text.

KEYWORDS

text generation, text simplification

ACM Reference Format:
Mucun Tian and Michael Green. 2019. Text; Its Generation, Simplification, and Complications. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

As Natural Language Processing becomes increasingly prevalent in our society, there are a number of subcategories that have drawn a fair amount of attention lately. One of these categories is automatic text simplification, the other is automatic text generation.

1.1 Automatic Text simplification

Automatic text simplification is a field of research under the aegis of NLP that considers many different ways to simplify text. It is accepted that a loss of clarity can occur open the simplification of text, and therefore ATS may be seen as a preprocessing of text rather than a direct solution to providing texts for all. This is due to texts being very reliant on being error free to retain their comprehensibility. Still, there is a core paradigm that text simplification can be used to makes texts available to all. This is a driving idea behind our text simplification.

Conference'17, July 2017, Washington, DC, USA
© 2019 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM...$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

ATS can be broken down into two distinct tasks, lexical simplification and semantic simplification. Lexical simplifications goal is to provide more legible alternatives for complicated words. For example, the sentence "We ran from the canines." could be simplified into "We ran from the dogs.". This provides the reader with a simpler sentence that is easier to understand. With semantic simplification the idea is to split a complex sentence in a smaller one. For example, "We went downtown last night to see a show and then we headed out to the movies on Overland" could be simplified into "We went downtown last night to see a show. Then we headed to the movies on Overland." As this paper focuses on lexical simplification, we will not be referring much more to semantic simplification.

There are two primary ways to lexically simplify text, one through machine learning and the other through the use of dictionaries. The machine learning method requires the user to have a tagged set of complex and simplified words, matched ideally one to one, in order to provide simplifications. Dictionary based lexical simplification rely on a dictionary to locate complex words and then generate synonyms(which can be done in a variety of ways), remove all the synonyms that are complex words themselves, and then order the remaining synonyms based on similarity to the original word. We have chosen to approach the simplification part of our project using the dictionary based model.

We hope to be able to scale text simplification to a specific grade level by incorporating the methods above, and creating dictionaries based on common complex words found in each grade level. By isolating these complex words we hope to be able to replace them with words found in a lower grade level, lowering the readability score.

1.2 Automatic Text Generation

One challenging part of text simplification using machine learning techniques is that there are few existing data sets labeled by complexity levels that can be used for training, and labelling new data sets requires huge amount of labors. To overcome this issue, we apply automatic text generation to generate texts from text sources. We select recurrent neural networks with long short-term memory (LSTM) units [12] as our model since this model can capture dependence in sequential data and was found to be suitable for generating text with varying length [9, 27]. To analyze the effectiveness of text generations using LSTM, we explore two base models — character-level LSTM [15, 27] and word-level LSTM [10, 25], and compare their performance by varying the model depth — a one-LSTM-layer model (1-Layer LSTM) and a two-LSTM-layers model (2-Layers LSTM). We also trained these models from a simple and a complex text source in order to investigate the impact of the complexity of training text sources on the generated texts. To evaluate the complexity of generated texts, we compute the Flesch[]

score of generated text for each combination of text sources, model structures, and simplification levels.

1.3 Research question

Our research questions are as follows:

RQ1 Using the text simplification model above, can we simplify to a predetermined grade level?
RQ2 Can we use automatically generated texts as the corpora for this simplification rather than relying on outside libraries like NewsAPI?
RQ3 Do text generators trained on simple texts generate simple texts?
RQ4 How do the model structure and training source impact on the complexity of generated texts?

2 BACKGROUND

2.1 Automatic Text Generation

Text generation has been researched since decades ago, and traditional text generation tasks can be defined by a series of pipeline tasks — content determination, text structuring, sentence aggregation, lexicalization, referring expression generation, and linguistic realisation [9, 20]. The main procedure is: 1. extracting useful information in text sources and sentences; 2. conducting lexicalization to select words and phrases that can well express the information extracted by step 1; 3. organizing all words and phrases in a grammatically-correct way to generate sentences. Research were focused on each of these stages, and methods are proposed based on two techniques — rule-based [2, 20] and statistical inference[5, 18].

Rule-based approaches generally use hand-designed syntactic rules (e.g. part-of-speech tagging, syntactic templates, etc.) to extract source texts and generate new texts. While these approaches exhibited good qualities of generated texts for some specific tasks in certain domains (e.g. sport score report) [28], it requires domain expertise with well-designed rules in order to produce accurate and understandable texts. This make it hard to be generalized to different domains.

Statistic-based methods, however, are driven by data sources, and can be trained to automatically generate texts for any domains without losing many qualities given enough data. Recently, it becomes popular due to the benefits of huge increasing of text sources in Internet. This is particularly true for deep learning models — the recurrent neural networks (RNNs) exhibited potential utilities on text generation tasks [27]. Sutskever et al. [27] proposed a multiplicative RNNs trained from character-level texts by Hessian-Free optimizer. This model outperforms other hierarchical non-parametric sequence models. Wen et al. [30] demonstrated that feasible texts can be generated from samples of output texts of a word-level LSTM model. To control semantics of generated texts, the authors optimized the model for sentence planning and surface realisation. In dialogue systems, Goyal et al. [10] showed that a character-based model outperforms a word-based model on the text quality. Drawing on the literature, we explore the utility of LSTM models on the task of text generation and simplification. We also compare the performance of a character-based model and a word-based model.

2.2 Automatic Text Simplification

Text readability is a field of study that has been researched, and debated, for quite a while. There were over 200 readability formulas [6] documented when this paper was written over a decade ago, however a number of these have risen to the top. Of note is the Flesch [7], Flesch-Kincaid [13], FOG [11], and SMOG [16]. Each of these metrics are based on similar computations, mostly comprising of syllable count, word count, and permutations of the two. While each utilize similar variables, they are not comparable to one another in an evaluative setting. Better to use one metric to compare multiple texts.

The first example of English lexical simplification[3] used Wordnet to find synonyms for words and then ordered them based on their frequencies. Complexity was determined by word occurrence. This was later improved upon [24] by changing the threshold, with the results measured in terms of number of correct terms being simplified. LexSiS [1] then provides an example that this paper follows. After a complex word is located a series of synonyms are generated, each of these is given a similarity score based on word vector models drawn from local context. There are some interesting ideas on the idea of substitution, sometimes choosing not to replace a word depending on the similarity, or lack thereof.

There are several frameworks for the evaluation of lexical simplification, the foremost of which is LEXenstein [19]. A thoroughly extensive framework, LEXenstein provides a start to finish explanation of lexical simplification. For each step in the lexical simplification process, LEXenstein provides numerous methods introduced in a variety of papers to execute that specific task. Of note is their VICTOR format, a digital way of representing complex words in a text and their synonyms. With this file format it becomes easier to evaluate the ranking of potential substitutions. However, LEXenstein does not automate the simplification of texts and requires very exact inputs, as previously mentioned, to even begin to provide evaluation on simplifications.

Above all of these is the book [22] on the matter. Contained in this work are 9 chapters, and an extensive bibliography, that cover almost every aspect of both lexical and semantic simplification. Of note are the chapters on Readability, Lexical Simplification, and Text Simplification Resource and Evaluation, all of which have contributed heavily to the writing of this paper.

3 DATA

There are two datasets that we aggregated for the project: one that is a dictionary of words, the other were a series of sample texts to apply the simplification on. The dictionary draws on resources from a teaching website [8], which lists "difficult words that appear in basal readers and other books commonly taught" for grades K-12. Scraping this website provided us with 13 dictionaries of difficult words associated with each grade level. The second site [17] provided us with a handy lists of potential resources to test our model on, as well as train our model with. These resources include: a CNN article (CNN) [26], a New York Times article (NYT) [29], a USA Today article (USA) [23], an excerpt from The Green Mile (TGM) [14], an excerpt from Harry Potter and the Sorcerer's Stone (HPSS) [21], and Alice's Adventures in Wonderland (AAiW) [4].

Table 1: Summary of data sets

Datasets	Chars	Uniq. Chars	Words	Uniq. Words
AAiW	144,412	47	27,827	2,914
NYT	144,462	61	24,049	6,329

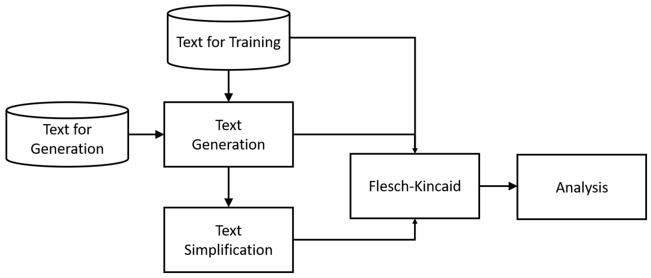

Figure 1: Model architecture

Table 1 shows the summary of data sets we used to train our text generation models. We selected the entire book of AAiW as our simple text source since its content matches kids' readability, and sampled text snippets from New York Times articles for our complex text source in order to make two training sets have comparable number of words and characters. We can see that NYT data has more unique words and characters than AAiW, which may indicate that texts in NYT are more complex. For the text generation, we used other text sources except training text sources.

4 MODEL AND APPROACH

Figure 1 shows our model architecture. Our experiment procedure is as follows:

(1) Train 1-Layer LSTM and 2-Layers LSTM on AAiW and NYT data sets, respectively.
(2) Generate texts from CNN, USA, TGM, HPSS using each of trained models.
(3) Compute Flesch-Kincaid score for each of generated texts and original texts used for training and generation.
(4) Analyze the results.

4.1 Automatic Text Generation

Table 2: Model structures of the text generation

Char-based LSTM		Word-based LSTM	
Layer	Neuron Params	Layer	Neuron Params
LSTM	256 tanh	Embedding	10
Dropout	0.2	LSTM	256 tanh
LSTM	256 tanh	Dropout	0.2
Dropout	0.2	LSTM	256 tanh
Dense	47 (61) softmax	Dropout	0.2
		Dense	2915 (6330) softmax

We employed two commonly-used LSTM models for text generation — character-based and word-based. The basic idea of these two models is to predict next character (word) using previous N characters (words). During the training process, each character (word) is converted to a numerical representation by its index trained from the whole training set, then each of N characters (words) are appended to a vector to be sent to the LSTM sequentially for training. Table 2 shows our model structures.

4.1.1 Char-based Model. Our char-based LSTM models predict next character using previous 100 characters. For 1-Layer LSTM, we used one LSTM layer with 256 units of tanh activation functions as the first layer, and the fully-connected dense layer of the softmax function as the output layer. We selected dropout with 0.2 rate as our regularization. For 2-Layer LSTM, we added one more LSTM layer behind the first layer. The model is optimized by adam algorithm to minimize the categorical cross entropy of the predicted and labeled characters. In text generation, we split the text source to sentences, truncate (or pad with white space string) each sentence to a sentence of 100 characters, predict next character, and append input characters with the predicted character to predict new character. This process is repeated until the maximum number of characters (300) is generated.

4.1.2 Word-based Model. For a "fair" comparison, we employed a similar structure of word-based LSTM model to the character-based model except a embedding layer is added as the first layer to convert index representations of words to a dense vector of length of 10. In the training process, we split the source text to sentences, and use n-gram words in each sentence as input to predict the next word. For example, given a sentence of "Alice's adventures in wonderland", we construct three pairs of features and labels as: 1. "Alice's" — "adventures"; 2. "Alice's adventures" — "in"; 3. "Alice's adventures in" — "wonderland". In the generation process, we split text sources into sentences and prepend the vector representation of each sentence with zeros to construct a input vector that has the length equal to the maximum length of all the training sentences (276 and 64 for AAiW and NTY, respectively). With new predicted next word, we append it to the original sentence and continue to predict next word until 100 new words are generated.

To answer our RQ3 and RQ4, We trained these four models on AAiW and NTY separately and generated texts from other four text sources. These texts then are prepared for simplification.

4.2 Automatic Text Simplification

Text simplification can be broken down into a handful of steps as laid out in LEXenstein's [19] documentation: identify the complex words, generate a list of suitable replacements, rank those replacements, replace the complex word with the top ranked replacement. Since we wanted this to be a scaleable project, our initial idea was to have the user input a grade level and simplify to that grade level. First the model is given a text and a grade level. We pass the text through a readability method that returns a Flesch-Kincaid score. The formula for Flesch Kincaid is:

$$FK(text) = (0.39 * avgSL) + (11.8 * avgSPW) - 15.59 \quad (1)$$

where *avgSL* is the average sentence length in number of words, and *avgSPW* is average syllables per word.The text is then tokenized and lower cased. Our experiments with lemmatization returned poor results, so we chose not to lemmatize the words. Other reasons include the affect this may have on the complexity of the word, as well as the impact it has on Flesch-Kincaid scores.

Once we have preprocessed the text, we pass it to our algorithm. The first step taken is the identification of complex words. For every word in the text we compare it to our dictionary of words for that grade level. If that word is found with in the dictionary, then it is marked as a 1. If that word is not found in the grade level dictionary, it is appended to a list of complex words, which also marks the location of the word in the initial text. After we have located all the complex words, we then generate a list of synonyms for each of those words.

For each synonym we compute the *Sim(S,W)* (*S* being the synonym and *W* being the word) of the synonym to the original word using the glove-wiki-gigaword-100 word vector from gensim. Then, for each synonym we determine its grade level by searching through all of our dictionaries of words arranged by grade level, and store the grade level of the word. If it is not found inside of our dictionaries, then the *gradeLevel* is set to 13. In order to determine the total weight of the synonym we use the following equation:

$$GradeSim(S,W) = Sim(S,W) * ((gradeLevel(S) - gradeDesired) * -1) \tag{2}$$

Where *gradeDesired* is the grade we are attempting to simplify down to. The reason that the difference between *gradeLevel* and *gradeDesired* is multiplied by negative one is we are weighting the synonyms in a lower grade word set than the ones that are in a greater word set. Once we have generated our list of synonyms and their weights, we organize the synonyms based on that weight, and return the highest value. Then that word replaces the more complex word. Afterwards we return the simplified text and check the Flesch-Kincaid score again. The goal at this point is not to return a legible text, only to simplify the words.

5 EVALUATION

With all texts — original texts, generated texts, and simplified texts in hand, we compute Flesch-Kincaid scores for each of them, summarized in Table 3. In the columns, the number 0, 1, 2, ..., 12 represents the grade level we simplified for. The column Original represents the FK scores of texts before simplification. In each data set, the first row represents FK scores computed from original texts instead of generated texts. The last two rows contain results of our training data sets.

5.1 RQ1

By looking at FK scores after simplification in each row, we can find a generally decreasing trend. This answers our **RQ1** that the simplification model can simplify texts into different levels in a reversed order to the grade level.

5.2 RQ2

All models can generate texts from the source texts, and Char-based models seems to be more robust than Word-based models in dealing with stop words explosion.

5.3 RQ3

In the last two rows, we see that FK score of AAiW is 5.1 that is about half of the one of NYT, this indicates that texts from AAiW are simpler than NYT, validating our experiment settings — training on two data sets with different text complexity. For each data set, the overall FK scores (Original column) across all models trained from the simple data source seem to be comparable to the one trained from the complex data source. While this may tell us that text generators trained on simple text do not necessarily generate simple texts, it may be caused by the poor fit of our generation models since we saw some of generated texts are just repeating the most frequent stop words in that data set. So we select results of the best fitting models to answer **RQ3**. We manually checked the quality of generated texts by all our models and found Word-based 1-Layer LSTM and Char-based 1-Layer LSTM models have the best quality of generated texts. In each data set, the FK scores (Original column) of texts generated by the Word-based 1-Layers LSTM model trained from simple texts are always smaller than the one trained from complex texts, and the Char-based 1-Layer model are almost following the same results. Based on the results of the best fit models, text generators trained from simple texts can generate texts of simpler readability than the one from the complex text source in most cases.

5.4 RQ4

By manual check of generated texts, we found the 1-Layer model for both Char-based and Word-based models can generate texts with better quality than the ones from 2-Layer models. The training time of 1-Layer models is also substantially lower than the 2-Layers models (~4 minutes/epochs vs. ~20 minutes/epochs). Given limited training time and training epochs, the 1-Layer models seems to work better than 2-Layer models. This results may be limited by our training epochs and prepossessing since we only use two relative small data sets as our training sets and we also didn't remove stop words in the training data sets.

6 CONCLUSION

From our project we can surmise that automatic text generation and evaluation is a difficult subject that could easily consume a year or two of research, much less a semester. We believe our expirement to mostly be a success, in that we are able to automatically generate texts and then fairly consistently return simplified texts that are less complex based on the Flesch-Kincaid metric. This metric is clearly not the end all be all of readability metrics, however it is one of the few suitable ones for this task. There are several ways that this project could have been improved upon. One of these ways would be a more extensive and hand selected dictionary for each grade level. The website they are scraped from specifically state that these are the top 80% of words in each grade's spelling curriculum. Having that other 20% could definitely aid in the identification and replacement of complex words.

Another potential improvement could be on the *GradeSim* metric. We did not have enough time to fully test this metric out, it is highly theoretical. Perhaps a better approach would be to multiply by a number smaller than -1, perhaps -.25? And maybe set a threshold as well? If the weight of the greatest synonym is greater than -2

Table 3: Evaluation results

Data	Train Source	Model	Model Depth	Original	0	1	2	3	4	5	6	7	8	9	10	11	12
CNN	-	-	-	10.1	11.9	1.1	10.9	10.1	9.8	9.3	9.2	9.3	9.3	9.4	X	X	X
	Simple	Char-based	1-Layer	36.4	26.6	25.3	25.2	24.8	24.7	24.6	**24.5**	**24.5**	24.6	24.6	24.6	24.7	24.7
			2-Layer	22.2	21.8	20.2	20.0	19.4	19.3	19.2	**19.1**	19.1	19.2	19.3	19.3	19.3	19.3
		Word-based	1-Layer	41.0	43.7	42.5	42.3	41.5	41.4	41.2	**40.9**	40.9	41.0	41.0	41.0	41.1	41.1
			2-Layer	39.4	39.7	39.6	39.6	39.4	39.3	**39.2**	39.2	39.2	39.2	39.3	39.3	39.3	39.3
	Complex	Char-based	1-Layer	31.2	28.4	27.9	27.8	27.5	27.4	27.3	**27.1**	27.1	27.1	27.2	27.2	27.3	27.3
			2-Layer	36.0	33.4	33.2	33.1	**32.8**	34.9	34.8	34.7	34.7	34.7	34.8	34.8	34.8	34.8
		Word-based	1-Layer	41.4	44.8	44.4	44.0	43.7	43.6	43.4	42.5	42.5	**42.0**	42.0	42.1	42.4	42.4
			2-Layer	39.4	39.7	39.6	39.5	39.4	39.3	**39.2**	39.2	39.2	39.2	39.3	39.3	39.3	39.3
USA	-	-	-	8.1	10.0	9.8	9.3	8.9	8.3	8.5	8.0	7.0	7.0	X	X	X	X
	Simple	Char-based	1-Layer	28.3	10.6	9.4	9.2	8.9	8.8	8.9	8.8	**8.6**	8.7	8.6	8.7	8.7	8.7
			2-Layer	18.5	13.1	11.4	11.2	10.8	10.7	10.7	10.7	**10.5**	10.6	10.6	10.6	10.6	10.6
		Word-based	1-Layer	25.9	29.0	27.8	27.6	26.9	26.8	26.7	26.1	25.9	**25.8**	25.9	25.9	25.9	25.9
			2-Layer	24.6	24.6	24.6	24.5	24.5	24.4	24.4	24.4	**24.2**	24.2	24.2	24.2	24.2	24.2
	Complex	Char-based	1-Layer	14.2	12.3	12.0	11.8	11.6	11.6	11.7	11.3	**11.1**	11.1	11.1	11.1	11.3	11.3
			2-Layer	36.0	33.4	33.2	33.1	32.8	34.9	34.8	**34.7**	34.7	34.7	34.8	34.8	34.8	34.8
		Word-based	1-Layer	28.7	31.6	31.4	31.1	29.8	29.7	29.6	29.4	29.2	**29.0**	29.0	29.0	29.1	29.1
			2-Layer	24.6	24.6	24.6	24.5	24.5	24.4	24.4	24.4	**24.2**	24.2	24.2	24.2	24.2	24.2
TGM	-	-	-	7.9	11.3	9.9	9.8	9.0	8.9	8.8	8.5	8.4	X	X	X	X	X
	Simple	Char-based	1-Layer	29.0	16.6	14.9	14.8	14.5	14.4	14.4	**14.3**	14.3	14.3	14.3	14.3	14.3	14.3
			2-Layer	21.1	17.2	15.4	15.3	14.8	14.7	14.7	**14.6**	14.6	14.6	14.7	14.7	14.7	14.7
		Word-based	1-Layer	42.8	45.9	44.6	44.5	43.7	43.6	43.5	**43.0**	43.0	43.0	43.1	43.1	43.1	43.1
			2-Layer	41.3	41.8	41.6	41.6	41.5	**41.4**	41.4	41.4	41.4	41.4	41.4	41.4	41.4	41.4
	Complex	Char-based	1-Layer	22.1	20.2	19.5	19.44	19.0	19.1	19.1	**18.8**	18.8	18.8	18.8	18.8	18.9	18.9
			2-Layer	28.4	24.3	23.8	23.8	23.5	25.0	25.0	24.9	**24.8**	24.9	24.8	24.9	24.9	24.9
		Word-based	1-Layer	44.6	49.6	48.5	48.3	47.2	47.3	46.9	46.7	46.8	**45.9**	45.9	45.9	46.0	46.0
			2-Layer	41.3	41.8	41.6	41.6	41.5	**41.4**	41.4	41.4	41.4	41.4	41.4	41.4	41.4	41.4
HPSS	-	-	-	6.9	8.7	7.7	7.5	6.6	6.4	6.2	6.0	X	X	X	X	X	X
	Simple	Char-based	1-Layer	41.5	14.8	13.2	13.1	12.8	12.7	12.7	12.6	**12.5**	12.6	12.6	12.6	12.6	12.6
			2-Layer	25.9	14.1	12.3	12.2	11.7	11.6	**11.5**	11.5	11.4	11.6	11.6	11.6	11.6	11.6
		Word-based	1-Layer	40.9	43.8	42.6	42.5	41.5	41.4	41.3	**40.8**	40.8	40.8	40.8	40.8	40.9	40.9
			2-Layer	39.3	39.4	39.2	39.2	39.1	39.1	**39.0**	39.0	39.0	39.0	39.0	39.0	39.0	39.0
	Complex	Char-based	1-Layer	24.9	18.4	17.9	17.8	17.5	17.6	17.5	17.3	**17.2**	17.2	17.2	17.2	17.3	17.4
			2-Layer	25.3	16.5	16.2	16.1	**15.8**	17.6	17.6	17.5	17.4	17.5	17.5	17.5	17.5	17.5
		Word-based	1-Layer	42.7	45.6	44.9	44.7	44.1	44.0	43.8	43.1	43.2	42.8	**42.7**	42.8	43.0	43.0
			2-Layer	39.3	39.4	39.2	39.2	39.1	39.1	**39.0**	39.0	39.0	39.0	39.0	39.0	39.0	39.0
AAiW	-	-	-	5.1	9.1	8.1	7.9	6.9	6.8	6.7	X	X	X	X	X	X	X
NYT	-	-	-	9.2	11.4	10.9	10.7	10.0	10.0	9.8	9.6	9.4	9.3	8.9	9.1	X	X

then don't replace the complex word? Greater than negative -3? All of these suppositions would require a battery of tests to better train and hone this metric.

On the subject of metric, the Flesch-Kincaid is not the most trustworthy of metrics. It is easy to fit to this metric, but we are not entirely convinced of its completely utility. To be noted, we had trouble getting TextStat's built in method to function so we wrote our own Flesch-Kincaid. We are counting every instances of a period, exclamation point, and question mark as the end of a sentence. Common sense dictates that there are numerous situations where that may not be the case, however, our assumption is that even if the punctuation is used in an outlier case (like an abbreviation, or title (Mr., Mrs., Etc)) since we are not modifying these characters, the number of sentences will be maintained between the original and simplified text.

Ultimately, we could easily put another 40 to 50 hours into this project, fine tuning all the hyper parameters to produce not only simplified text, but also legible text. Then there could also be the idea of complexifying text; finding all the too simple words in a sentence and making them more complex so that sentences can be presented on a sliding scale or readability. We believe this could be an excellent tool to help students learn English.

REFERENCES

[1] Stefan Bott, Luz Rello, Biljana Drndarevic, and Horacio Saggion. 2012. Can spanish be simpler? lexsis: Lexical simplification for spanish. *Proceedings of COLING 2012* (2012), 357–374.

[2] Sabine Nicole Buchholz. 2002. Memory-based grammatical relation finding. (2002).

[3] 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. 7–10.
[4] Lewis Carroll. 2000. *Alice's Adventures in Wonderland*. Broadview Press, Peterborough, Ontario.
[5] Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*. Association for Computational Linguistics, 1–8.
[6] William H DuBay. 2004. The Principles of Readability. *Online Submission* (2004).
[7] Rudolf Franz Flesch et al. 1949. Art of readable writing. (1949).
[8] Flocabulary. 2019. https://www.flocabulary.com/
[9] Albert Gatt and Emiel Krahmer. 2018. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research* 61 (2018), 65–170.
[10] Raghav Goyal, Marc Dymetman, and Eric Gaussier. 2016. Natural language generation through character-based rnns with finite-state prior knowledge. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. 1083–1092.
[11] Robert Gunning. 1952. The technique of clear writing. (1952).
[12] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.
[13] J Peter Kincaid, Robert P Fishburne Jr, Richard L Rogers, and Brad S Chissom. 1975. Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel. (1975).
[14] Stephen King. 1997. *The Green Mile: A novel in six parts*. Plume, New York, NY.
[15] Zachary C Lipton, Sharad Vikram, and Julian McAuley. 2015. Generative concatenative nets jointly learn to write and classify reviews. *arXiv preprint arXiv:1511.03683* (2015).
[16] G Harry Mc Laughlin. 1969. SMOG grading-a new readability formula. *Journal of reading* 12, 8 (1969), 639–646.
[17] My Byline Media. 2019. http://www.readabilityformulas.com/flesch-grade-level-results.php
[18] David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Lingvisticae Investigationes* 30, 1 (2007), 3–26.
[19] Gustavo Paetzold and Lucia Specia. 2015. Lexenstein: A framework for lexical simplification. *Proceedings of ACL-IJCNLP 2015 System Demonstrations* (2015), 85–90.
[20] Ehud Reiter and Robert Dale. 2000. *Building natural language generation systems*. Cambridge university press.
[21] J.K. Rowling. 1999. *Harry Potter and the Sorcerer's Stone*. Scholastic, New York, NY.
[22] Horacio Saggion. 2017. Automatic text simplification. *Synthesis Lectures on Human Language Technologies* 10, 1 (2017), 1–137.
[23] Amy Sancetta. 2011. Wal-Mart reveals overseas operations investigation. https://usatoday30.usatoday.com/money/industries/retail/story/2011-12-09/wal-mart-investigation/51762402/1?loc=interstitialskip
[24] Matthew Shardlow. 2013. A Comparison of Techniques to Automatically Identify Complex Words.. In *51st Annual Meeting of the Association for Computational Linguistics Proceedings of the Student Research Workshop*. 103–109.
[25] Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. *arXiv preprint arXiv:1506.06714* (2015).
[26] CNN Wire Staff. 2011. Midnight deadline passes for Occupy Boston protesters to clear out. https://www.cnn.com/2011/12/09/us/massachusetts-occupy-deadline/index.html
[27] Ilya Sutskever, James Martens, and Geoffrey E Hinton. 2011. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*. 1017–1024.
[28] Mariët Theune, Esther Klabbers, Jan-Roelof de Pijper, Emiel Krahmer, and Jan Odijk. 2001. From data to speech: a general approach. *Natural Language Engineering* 7, 1 (2001), 47–86.
[29] Anthony Tommasini. 2011. Emboldened Orchestras Are Embracing the New. https://www.nytimes.com/2011/12/11/arts/music/emboldened-orchestras-embracing-the-new-music.html?_r=1&ref=arts#h[]
[30] Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. *arXiv preprint arXiv:1508.01745* (2015).