Cairo University
Faculty of Engineering
Credit-Hour System
Communication and Computer Engineering
CCEN 481: Graduation Project-2
Spring 2019

SumSim
For Text Summarization and Simplification

Prepared by:

Amira Ahmed Amer
Dina Adib Fouad
Rana Amr Afifi
Salma Hanafy Ahmed

Supervised by:

Dr. Ahmed S. Kaseb

Acknowledgment

We would like to express our sincere gratitude to our supervisor **Dr. Ahmed S. Kaseb** for his continuous guidance and support throughout the entire graduation project. We would like to thank **Dr. Nevin Darwish** and **Dr. Mohsen Rashwan** for discussing the feasibility of the project and how to approach it. Finally, we would like to thank **Dr. Ali Ali Fahmy** for his consultation and suggestions to solve problems we faced throughout the project

Abstract

With the increasing rates of illiteracy and ignorance in our world today comes the need of providing a method to make it easier to read. Our project aims to help children, people with low intellectual abilities or even non-native English speakers read and comprehend any paragraph they desire. Therefore, our project provides them with a language tool that summarizes and simplifies English text. By using our tool, complex sentences can be made easier to read and comprehend by simplifying their structure and replacing unfamiliar words. While the summarizer reduces text size by only keeping the most important sentences in a given paragraph. Through our project, we intend to promote literacy and to remove the obstacle of text complexity from the way of aspiring readers.

If the abstract is too long to read, a summarized version generated from our too can be found below.

> By using our tool, complex sentences can be made easier to read and comprehend by simplifying their structure and replacing unfamiliar words. While the summarizer reduces text size by only keeping the most important sentences in a given paragraph.

Table of Contents

List of Figures

List of Tables

Abbreviations

TF	Term Frequency
IDF	Inverse Document frequency
TF-IDF	Term Frequency-Inverse Document frequency
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
GloVe	Global Vectors for Word Representation
UNK	Unknown
PMW	Per Million Word
DRESS	Deep Reinforcement Sentence Simplification
WSD	Word Sense Disambiguation
SVM	Support Vector Machines
Seq2Seq	Sequence to Sequence

Chapter 1
Introduction

1.1 Project Idea

"*It is my ambition to say in ten sentences what others say in a whole book*." ―Friedrich Nietzsche, philosopher and poet.

Simplicity and brevity are two elementary traits of any readable piece of text. Not only they convey the meaning in the minimum possible time and effort, but they also encourage people with reading difficulties to read more. In our project, we present a language tool that provides both *Text Simplification syntactically and lexically* and *Text Summarization* for its users. In the next paragraph, we will be explaining the formal definition of Simplification and Summarization in Natural Language Processing.

Both *Text Simplification* and *Summarization* are research topics that have been under exploration in NLP since the 1990s. At the beginning, they were approached by statistical means, but with the rise of deep learning and neural networks, new methods have been developed that have shown to yield more accurate and more fluent method [9]. It is important to mention that while simplification is performed on the sentence level, summarization is applied on the document level.

Simplification is the process of making text more readable and easier to comprehend by replacing complicated words with simpler ones and modifying syntactic structures to make the text easier to follow up with while maintaining necessary information. The following example illustrates the effect of simplification on sentences.

Original Text: *The population of Afghanistan is divided into a wide variety of ethnic groups of which Tajik, Pashtun, Hazara, Uzbek and Baloch.*

Simplified Text: *The population of Afghanistan is divided into many ethnic groups.*

Simplification is mainly done on two levels: *syntactic simplification* where long and complex sentences are transformed into syntactic equivalents which could be easier to understand and *lexical simplification* where complex words and phrases are replaced with more common ones. Our language tool provides both approaches to any piece of text provided by the user where they can also specify their preferences in selecting one or more of the mentioned techniques. For syntactic simplification, we have implemented a *Seq2Seq* model which works with *Beam Search* and *LSTM*, and for *lexical simplification* we have built a *Neural Network* for ranking the word's substitutes.

Meanwhile, summarization is the process of shortening a text document. There are two major notions of summarization: *extractive* and *abstractive*. In extractive summarization, the most relevant and necessary sentences are selected to be included in the summary while the others are omitted. Whereas, abstractive summarization involves generating entirely new phrases and

sentences that capture the meaning of the source document. Our language tool provides extractive summarization and we defer abstractive summarization for future work. We have implemented extractive summarization by applying a *Text Rank* algorithm that gives scores for sentences and selects only those with the highest scores. The example below demonstrates the application of extractive summarization on a given piece of text.

Original Text: *'Alcohol' is taken in almost all cool and cold climates, and to a very much less extent in hot ones. Thus, it is taken by people who live in the Himalaya Mountains, but not nearly so much by those who live in the plains of India. Alcohol is not necessary in any way to anybody. The regular use of alcohol, even in small quantities, tends to cause mischief in many ways to various organs of the body. It affects the liver, it weakens the mental powers, and lessens the general energy of the body. In addition, damage to the central nervous system and peripheral nervous system can occur from chronic alcohol abuse.*

Summary: *'Alcohol' is taken in almost all cool and cold climates, and to a very much less extent in hot ones. Alcohol is not necessary in any way to anybody. The regular use of alcohol, even in small quantities, tends to cause mischief in many ways to various organs of the body.*

1.2 Motivation

There were actually quite a number of motivations that inspired us to pursue this project. The major motivation behind it is to encourage and promote literacy; people are reading less every day and simplifying and summarizing text would encourage them to read documents that they perceive as long and boring. The following Figure 1 [29] illustrates literacy rate in the world in 2015. Unfortunately, the lowest literacy rates (60%-80%) are found in the Middle East and North Africa regions.

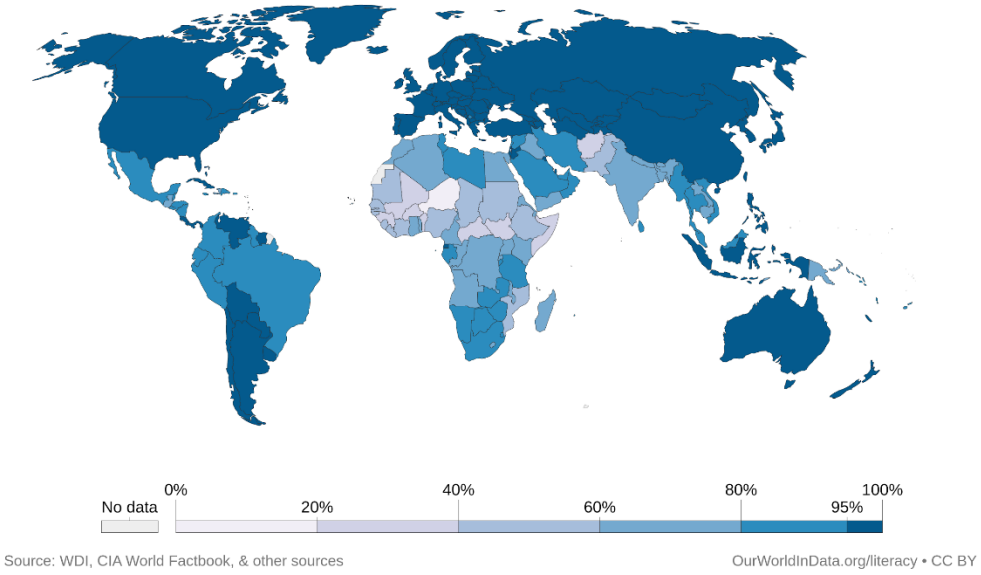


Figure 1-1: Literacy Rates in the World

Secondly, there is a vast majority of audience who lack the skills that enable them to understand complex text. Some segments of such audience include: children, non-native speakers, native speakers of low literacy levels, people with autism, and people with language disorders such as Dyslexia or Aphasia. It has been shown that people with language disorder constitute a percentage that ranges between 3 and 8% [36].

Before settling on this project, we have conducted a survey to have more insights about the usefulness and utility of Text Simplification technology. The survey was conducted in November 2018 and was distributed through social media platforms. There were 441 respondents; the majority of them were in the range between 18-25 years old. We asked the respondents about the best target audience for such technology. Figure 2 plots the results; there were 235 votes for Second Language Learners, 179 votes for children, 135 for people with intellectual disabilities and 162 for adults with reduced literacy.

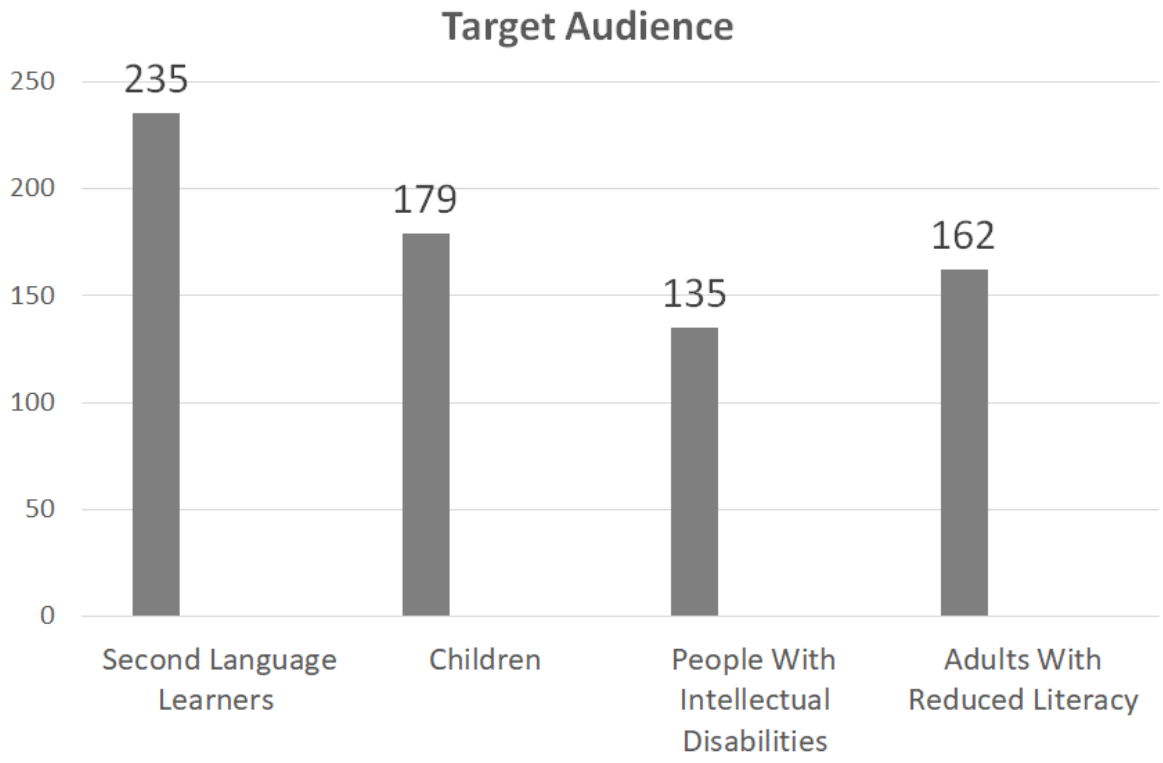


Figure 1-2: Responses from Our Survey on Simplification Target Audience

To have a better insight on how people think Text Simplification can promote and encourage literacy, we asked the respondents to give their opinion on this matter on a scale from 1 to 10 (1 being the least and 10 being the most). The responses are demonstrated in figure 3 below.

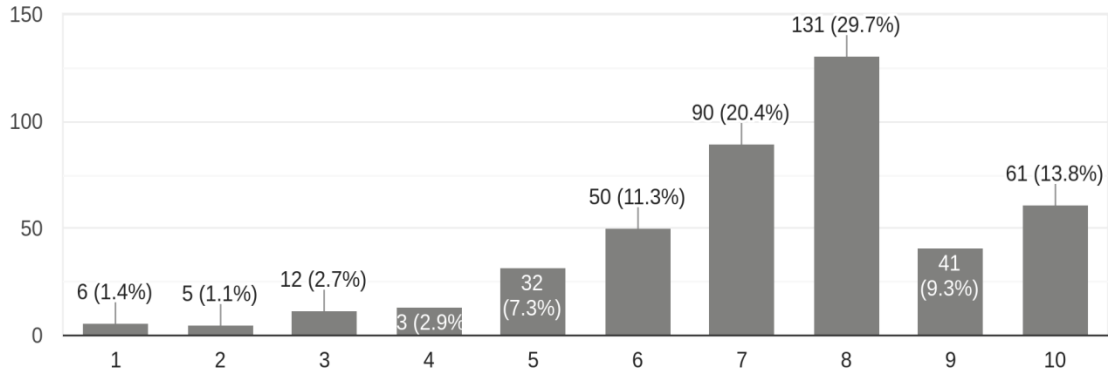


Figure 1-3: Responses from Our Survey on Text Simplification Impact

The results obtained from this survey and the statistics we have found online have encouraged us even more to choose Text Simplification and Summarization for our graduation project.

1.3 Document Overview

In this document we elaborate all the details regarding our language tool. In Chapter 2, we discuss the necessary background for this project. Since our project consists contains both a Text Summarizer and Simplifier, we first explain TF-IDF which is the information retrieval technique used in the summarizer and then we demonstrate important Seq2Seq as well as word embeddings for the simplifier. In Chapter 3, we review on the commonly used approaches for both the syntactic and lexical simplifiers.

In Chapter 4, we demonstrate the system architecture and how all the modules are connected. We also justify the reasons for our design and architecture.

After the big picture is explained, we shed the light on each module separately. In Chapter 5, we discuss the data used, the algorithms and the evaluation metrics for each module. In Section 1, we discuss the Summarizer and elaborate on the Text Rank algorithm that was used. In Section 2, we discuss the syntactic simplifier and raise the curtains off all the magic that occurs in the Recurrent Neural Network that was used. Afterwards, in Section 3 we describe the lexical simplifier and illustrate on the three main tasks of lexical simplification which are word identification, substitution generation and substitution ranking. Last but not least, in Section 4 we demonstrate the complexity assessment module that was used to compare the sentence complexity before and after simplification.

In Chapter 6, we demonstrate on all the experiments that we have tried in order to make our final product. The most notable of them is the fact that our project was first intended to address Arabic Language but due to the lack of datasets and mature technology in Arabic, we had to switch to English. Yet, our trials in English weren't also very successful at the beginning, but we didn't settle until we created a technology that is equivalent to the current state of the art.

In chapter 7, we describe the tools and libraries that were used in our project. Next, in Chapter 8, we give our recommendations for future work and conclude our project. Finally, we refer to all the research papers and references that we used to build our project in chapter 9.

1.4 Contributions

Since our project is research-oriented, we have been keen to implement the state of the art technologies and add contributions to them. The following list summarizes our contributions to the current technologies in both Text Simplification and Summarization.

- *Syntactic Simplification:* We implemented a sequence-to-sequence model with an attention mechanism in flexible way to allow changes, such as number of hidden units and number of layers, to be done easily to allow experimentation. We trained the sequence-to-sequence model on different hyperparameters and architectures. We also further trained the model using reinforcement learning that we implemented and using our proposed reward function. The sequence-to-sequence model we trained for syntactic simplification was tested and compared to the results given by other well-known models in the field.
- *Lexical Simplification:* We have implemented our own features and algorithm for the Substitution Ranking module and have successfully outperformed the state of the art – the Neural Readability Ranker by [15] – by 13%. The accuracy of the whole pipeline of our lexical simplification module has also exceeded the best current model [56] by 9.8%.
- *Text Summarization:* We also made a full implementation of the improved TextRank algorithm for summarization that was proposed by Barrios et.al [5] in their paper. The summarizer was tested using news articles from BBC news and the results were compared to those given by the gensim library implementation that is commonly used for the summarization task.

Chapter 2
Necessary Background

In this chapter, we explain the necessary on the background that is important to comprehend our project. We first demonstrate on TF-IDF which is one of the algorithms used in the summarizer. Then we discuss Seq2Seq models, LSTM and Word Embeddings for the simplifier.

2.1 TF – IDF

TF-IDF is a measure used in Information Retrieval tasks to evaluate the importance of a word to each document in a group of documents.TF-IDF for a word in a certain document is equal to the product of the IDF of this word and the TF of this word in the document.TF for word w_i in document d_j is calculated as follows[1]:

$$\mathrm{TF}(w_i, d_j) = \frac{numberof \times w_i appears \in d_j}{numberof words \in d_j}$$

The IDF for a word w_i is calculated as:

$$\mathrm{IDF}(w_i) = log\left(\frac{numberof documents}{numberof documents containing w_i}\right)$$

TF-IDF of a document is the sum of TF-IDF of all words in the document, which can be expressed as:

$$\text{TF-IDF}(d_j) = \sum_{all\ words\ w_i\ in\ d_j} \mathrm{TF}(w_i, d_j) * \mathrm{IDF}(w_i)$$

2.2 Basics of Simplification Model

2.2.1 LSTM

LSTM is a type of RNN that solves the problem of short-term memory by having gates that learn which data is important to keep and which can be discarded. Similar to RNNs, LSTMs processes the sequence of inputs one by one. An LSTM cell that processes one input produces a hidden state which is passed to the LSTM cell that processes the next step of the sequence. Hidden states act like a memory for the neural network enabling the information from previous steps to flow through future steps.

RNN cell which calculates the output hidden state by concatenating the input and previous hidden state and passing them through a tanh function which squishes the values to be always between -1 and 1, therefore as time passes the effect of inputs at the beginning of the sequence begin to vanish which is referred to as the short-term memory problem.

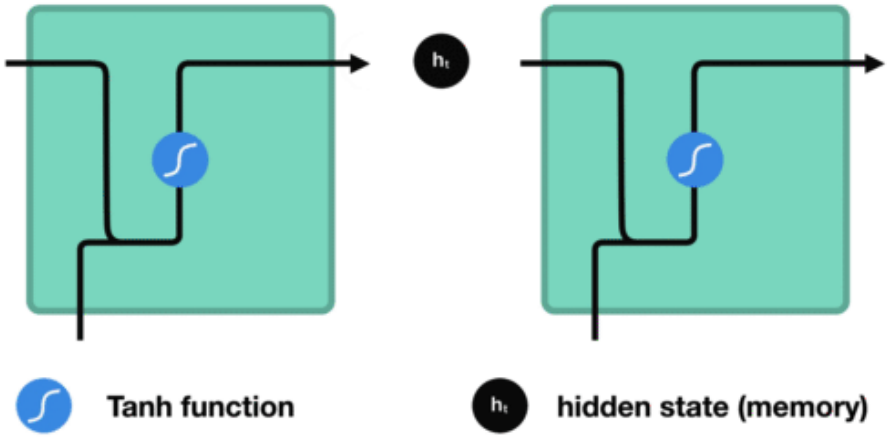


Figure 2-1: Flow of Hidden State through RNN. Retrieved from [2]

To solve the short-term memory problem LSTMs introduces three gates and a cell state to control information flow and allow most important information to be kept rather than information at the end of the sequence. Cell state transfers relative information all the way down the sequence chain. Information is added or removed from the cell state by the gates as it flows through LSTM cells.

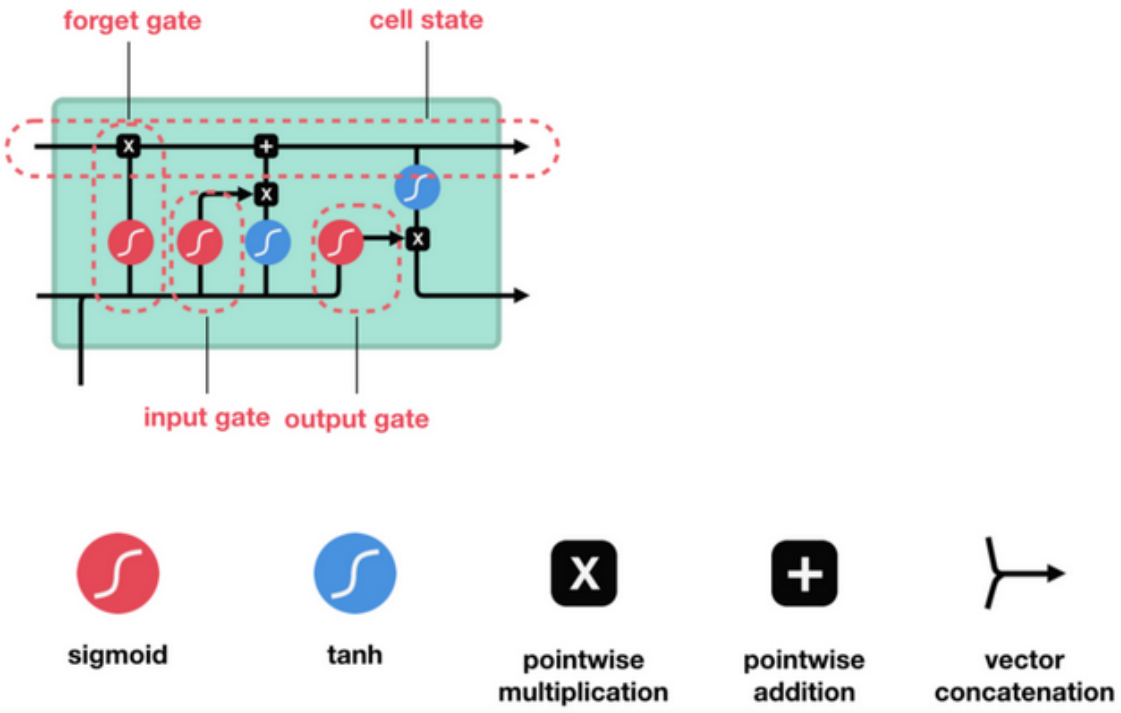


Figure 2-2: LSTM Cell. Retrieved from [2]

Forget gate decides if information should be kept or discarded. The forget gate output is calculated by concatenating the input and previous hidden state and passing them through a sigmoid function, where values come out between 0 and 1. The closer the values are to 0 means to forget, and the closer to 1 means to keep.

Input gate is used to calculate the new cell state along with the forget gate output. First step of input gate is to pass concatenated hidden state and current input into the sigmoid function as with the forget gate. Then, pass concatenated hidden state and current input into a tanh function to help regulate the network. Then you multiply the tanh output with the sigmoid output. The sigmoid output will decide which information is important to keep from the tanh output.

New cell state is calculated by point-wise multiplication of the previous cell state with the forget gate output followed by point-wise addition of the result with the input gate output. This updates the cell state to new values that the neural network finds relevant.

Output gate decides what the next hidden state should be. First, we pass the previous hidden state concatenated with the current input into a sigmoid function. Then we pass the new cell state to a tanh function. We multiply the tanh output with the sigmoid output to decide what information the hidden state should carry.

2.2.2 Seq2Seq Model

A Seq2Seq model is a model that aims to map an input sequence into another output sequence. Seq2Seq models consist of two RNNs, an Encoder and a Decoder. The Encoder provides an encoding of the source sentence. This encoding is used to provide the initial hidden state of the Decoder RNN. The Decoder produces the target sentence conditioned on the encoding provided by the Encoder. Given a (complex) source sentence$X =$, the model learns to predict the (simplified) target sentence $=$. The encoder transforms the source sentence X into a series of hidden states using LSTM while the decoder uses another LSTM to generate one word y_{t+1} at a time of the target sentence. The actual output of the Decoder is the probability distribution of what the output word at the current step should be. The generation is conditioned on all the previous outputs $y_{1:t}$ and a dynamically created context vector c_t:

$$P(Y|X) = \prod_{t=1}^{|Y|} P(y_t|y_{1:t-1}, X)$$

$$P(y_{t+1}|y_{1:t}, X) = \text{softmax}(g(\mathbf{h}_t^T, \mathbf{c}_t))$$

where g(.) is a one hidden layer neural network with the following parameterization:

$$g(\mathbf{h}_t^T, \mathbf{c}_t) = \mathbf{W}_o \tanh(\mathbf{U}_h \mathbf{h}_t^T + \mathbf{W}_h \mathbf{c}_t)$$

where $\mathbf{W}_o \in \mathbb{R}^{|V| \times d}$, $\mathbf{U}_h \in \mathbb{R}^{d \times d}$, and $\mathbf{W}_h \in \mathbb{R}^{d \times d}$; $|V|$ is the output vocabulary size and d the hidden unit size. h_t^T is the hidden state of the decoder LSTM which summarizes $y_{1:t}$:

$$\mathbf{h}_t^T = \text{LSTM}(y_t, \mathbf{h}_{t-1}^T)$$

We perform an argmax to get the word with the highest probability which is known as greedy decoding. The output word at step t is fed to the RNN to get the output at step t+1. The decoding process is stopped when the Decoder produces an end of sequence token.

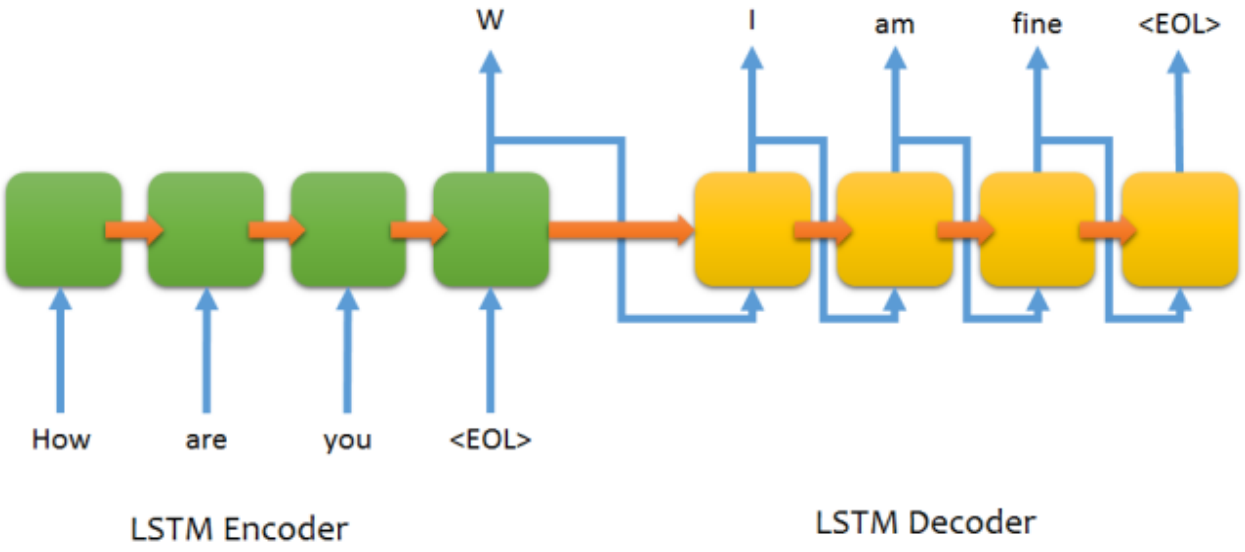


Figure 2-3: Seq2Seq Model

2.2.2.1 Beam Search Decoding

The goal of the decoder is to maximize the probability of the output sequence for the given input sequence. The problem with greedy decoding is that choosing the word with the highest probability at each time step does not guarantee the maximum probability over the whole sequence. In order to find the optimum solution, we should generate all the possible sequence combinations and choose the sequence with the highest probability, but this is very expensive as the search space is very large. To reach a better solution for the decoding problem beam search technique was introduced.

Beam search keeps track of the k most probable partial translations. The constant k is called the beam size which defines the number of alternatives we keep track of simultaneously. Beam search avoids being totally greedy while keeping the search space smaller than exhaustive search. A typical value of k ranges between 5 to 10 [3].

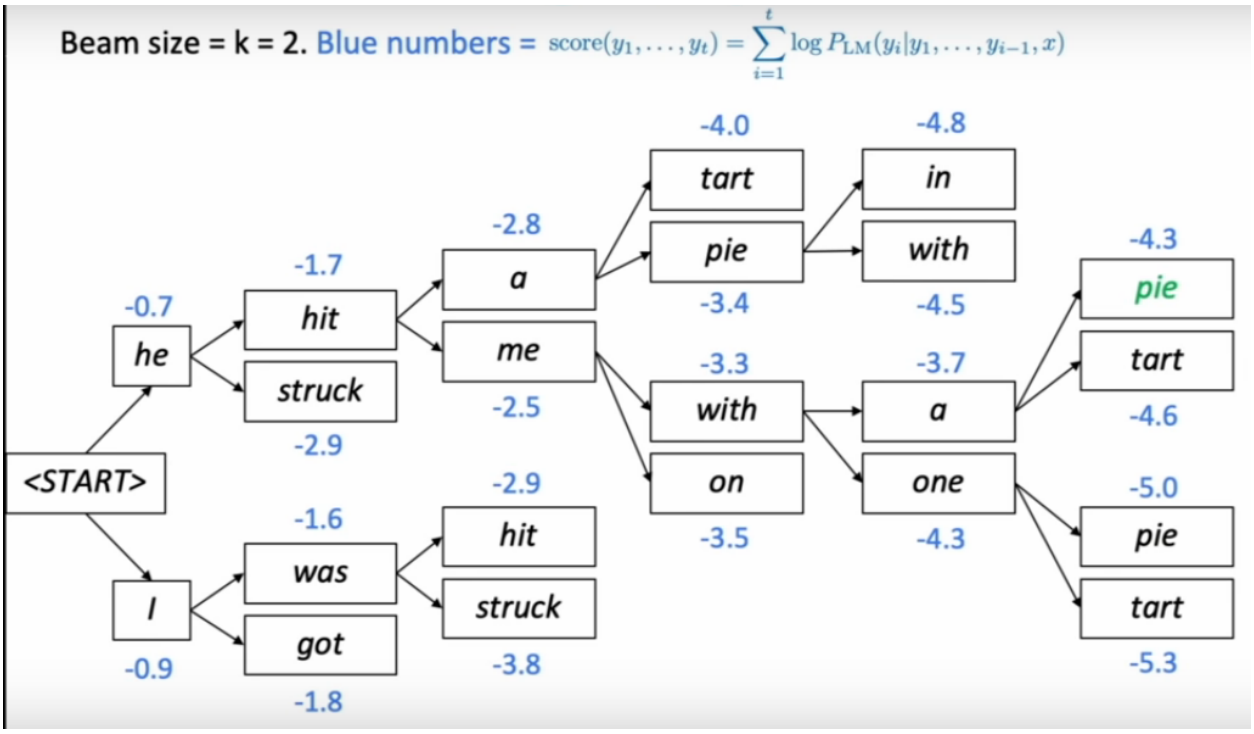
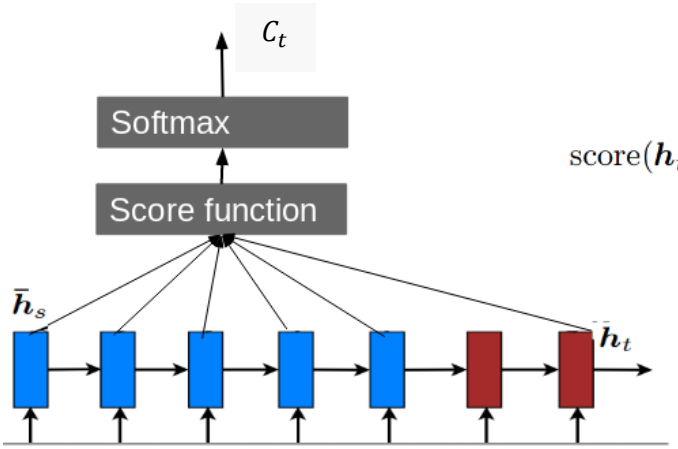


Figure 2-4: Example of Beam Search with k = 2

2.2.2.2 Attention Mechanism

In Seq2Seq model the Encoder tries to capture the features in the input and encode it in one vector and give it to the Decoder to generate the output. This could give acceptable results for shorter sentences, but as the sentence gets longer it gets harder to memorize all the features. For example, a human trying to translate a sentence will find it difficult to memorize the sentence and try to translate it from memory. Instead humans focus on a phrase, translate it and then move to the next one. This is similar to what attention mechanism does. It tells the decoder which input words it should focus on while generating the output by assigning weights to each input word.

Attention passes the hidden state of Decoder at the current step along with all the hidden states of the Encoder to a scoring function. This scoring function can be dot, general or concat(additive). The output of the scoring function is passed to a softmax function to give the attention weights.



$$\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \begin{cases} \boldsymbol{h}_t^\top \bar{\boldsymbol{h}}_s & \textit{dot} \\ \boldsymbol{h}_t^\top \boldsymbol{W}_a \bar{\boldsymbol{h}}_s & \textit{general} \\ \boldsymbol{v}_a^\top \tanh\left(\boldsymbol{W}_a [\boldsymbol{h}_t; \bar{\boldsymbol{h}}_s]\right) & \textit{concat} \end{cases}$$

Figure 2-5: Attention Weight Calculation

The weighted sum of the Encoder hidden states using the attention weights is calculated to give the context vector. The context vector and the current state are passed through a neural network layer to give the Decoder output.

2.3 Word Embeddings

Word Embeddings are vectors of numbers that represent words. The two most popular word embedding techniques are Word2Vec [32] and GloVe [28]. They are capable of capturing words' meaning, their context in a document, and how words are related to each other. One example that shows the effectiveness of these word vectors, is its ability to encode the analogy "king is to queen as man is to woman" in vector space. Thus, the vector (king- man + woman) maps most closely to the word vector for queen.

There are two main techniques used in learning word vectors, namely Global Matrix Factorization methods and Local Context Window methods. Each of the two models has its own drawbacks that make the model behave poorly when used alone.

2.3.1 Global Matrix Factorization

Global matrix factorization is applying matrix factorization on term-document frequencies, in which the rows are words and columns represent documents, or term-term frequencies that represent co-occurrences between words. These frequencies are obtained from a large corpus.

The techniques applied on term-document frequencies matrices are commonly known as Latent Semantic Analysis (LSA) [30]. They are based on the assumption that words that are close in meaning tend to appear in similar documents. The similarity between two documents is usually measured as the cosine similarity (or the dot product) between the normalized vectors of the two

columns representing these two documents. LSA applies Singular Value Decomposition (SVD), which is a matrix factorization technique used in linear algebra, on the term-document frequencies matrix to reduce its row dimensionality while keeping the similarity structure among columns. The output matrix is then a representation of the global corpus statistics.

2.3.2 Local Context Methods

Local Context methods learn word embeddings by passing a window over the corpus line-by-line. It has two main types of models; the skip-gram model [33], which is used to predict the surroundings of a given word, and the continuous bag-of-words model (CBOW) models of Mikolov et al. (2013a) [31] which predicts the word given its surroundings.

While local context methods behave relatively better than global matrix factorization on word analogy tasks, they lack the ability to capture the global corpus statistics since they use separate local windows instead of global co-occurrence counts.

2.3.3 GloVe Vectors

GloVe model is an open-source project that was developed at Stanford [28]. It is an unsupervised learning algorithm that is used to obtain vector representations for words by combining the previous two methods.

GloVe uses matrix factorization of term-term frequency matrices, which represent co-occurrences between words as a large two-dimensional matrix where rows and columns are enumerated unique tokens in the corpus, and each entry represents how often the column term appears in the context of the row term. However, this matrix should be symmetric since the relation goes both ways, meaning that if word i appears in the context of word j, then word j must appear in the context of word i. The authors of GloVe found that using raw co-occurrences was flawed, so they used co-occurrence probability ratios instead to remove noise terms that were not related to both words. They explained this in a more detailed example in their paper [28]. Then, they attempted to design a function that maps word vectors to ratios of co-occurrence probabilities. The purpose of this function is to discriminate any two given word vectors with the help of their context vectors. The authors then incorporate this into a least-squares regression problem with the following objective function to be minimized:

$$J = \sum_{i,j=1}^{V} f(X_{ij})(w_i^T \tilde{w}_k + b_i + \tilde{b}_k - \log X_{ik})^2,$$

Where V is the size of the vocabulary, Xij tabulate the number of times word j occurs in the context of word i, Wi is the vector of center word i, Wk is the vector of context word k, bi is the

bias term for word i, bk is the bias term for word k and Xik is the number of times word k appears in the context of word i.

f is a weighting function that is used to tune our objective function so as to obey three main properties; to have a limit of 0 as x goes to 0, to be non-decreasing so as not to overweight rare co-occurrences, and to be relatively small for large values of x to not overweight frequent co-occurrences.

$$f(x) = \begin{cases} \left(\frac{x}{x_{\max}}\right)^{\alpha} & \text{if } x < x_{\max} \\ 1 & \text{otherwise} \end{cases}$$

This expression ensures that f has values between 0 and 1, α is a tuned parameter that was chosen to be equal ¾ with no real intuition behind it.

GloVe outperforms word2vec and SVD, which are local context methods, in several tasks as word analogy, word similarity and named entity recognition since it captures the global statistics of a corpus using its global objective function in addition to obtaining co-occurrence statistics using a context window over the corpus.

Chapter 3
Literature Review

Automatic text simplification first appeared in research in the late nineties as a preprocessing step for other NLP tasks such as summarization and information extraction. Most of the research was directed towards building rule-based systems to provide lexical and syntactic simplification until Simple English Wikipedia aligned with English Wikipedia offered a large parallel corpus to train data-driven systems. One of the proposed ideas was treating automatic text simplification as monolingual machine translation model [12].

3.1 Syntactic Simplification Approaches

Zhang and Lapata [9], Vu et al. [10], Narayan and Gardent [8] and Wubben et al. [7] all considered simplification to be a machine translation problem, where the source language is English and target language is Simple English.

Both Narayan and Gardent [8] and Wubben et al. [7] use a phrase based machine translation (PBMT) model trained on Simple English and English pairs.

To achieve the intended output Wubben et al. [7] produces 10 hypotheses from the system, reranks them according to Levenshtein distance with the original sentence and picking the closest hypothesis to the original sentence. Narayan and Gardent [8] produced better results by passing the complex sentence to a probabilistic model that performs sentence splitting and deletion. Advancements in machine translation introduced sequence-to-sequence models which provided an end-to-end solution for text simplification which neither need extra modules to perform deletion and splitting tasks nor do they need to restrict input to be lowered cased to achieve reasonable results.

Zhang and Lapata [9] introduced a Seq2Seq model using a 2-layer unidirectional LSTM Encoder-Decoder with attention called EncDecA. Furthermore, they viewed their model as an agent and applied a reinforcement learning using REINFORCE algorithm to achieve even better results with their model and they called it DRESS. Reward function used accounts for simplicity, relevance and fluency. To further enhance lexical simplification they used an encoder-decoder model pertained on parallel corpus of complex and simple sentences that encourages lexical substitution. They integrated lexical simplification with the reinforcement learning trained model using linear interpolation. The final model is called DRESS-LS.

Vu et al. [10] argues that Seq2Seq are designed to capture long-term dependencies in a sequence but their memories are not enough to accommodate for dependencies in long complicated sentences. They see that encoder should consider not only the previous words, but also the future sequence of words which might contain useful information for the simplification task. Therefore, they replaced the LSTM cells in the encoder with a Neural Semantic Encoders which have extra memory that allows the encoder to have unrestricted access to the entire source sequence stored

in the memory so that the encoder may attend to relevant words when encoding each word. Vu et al. [10] suggested that using bidirectional encoders instead of unidirectional ones might offer a solution to the long dependency problem.

For the text simplification task there are two datasets mainly used by most researches using data driven text simplification. The first dataset is PWKP that was built by Zhu et al. [13] using 65,133 articles from English Wikipedia and Simple English Wikipedia. They began by pairing the articles through following the "language link" using the dump files in Wikimedia. After preprocessing they aligned the articles sentence-wise using sentence-level TF*IDF as a similarity measure, where if TF*IDF is above a certain score they are aligned together. PWKP dataset offers more than 108k sentence pairs of which 100 are reserved for the test set. In their paper Xu et al. [11] argue that Wikipedia is suboptimal as a simplification data resource because it contains a large portion of poor quality simplification, has errors that are introduced by its automatic alignment and it does not generalize well on texts from other genres. By manually inspecting 200 randomly sampled sentences, they found that only 50% of the sentences are truly simplified while the others are either partially overlapping or not simpler. To overcome the problems of Wikipedia Xu et al. [11] introduced the second dataset which consists of 1,130 news articles that are re-written 4 times for children at different grade levels by editors at Newsela. Sentences are aligned based on Jaccard similarity between the sentences lemmas and sentences having similarities smaller than 0.40 are discarded.

3.2 Lexical Simplification Approaches

The goal of a lexical simplification system is to simplify text on a word level by replacing complex words with simpler ones, while maintaining the meaning of sentences and correct grammar. While various approaches have been addressed for the lexical simplification task specifically, they often have the same pipeline with different methods proposed in each step of the pipeline. Following is the lexical simplification pipeline used.

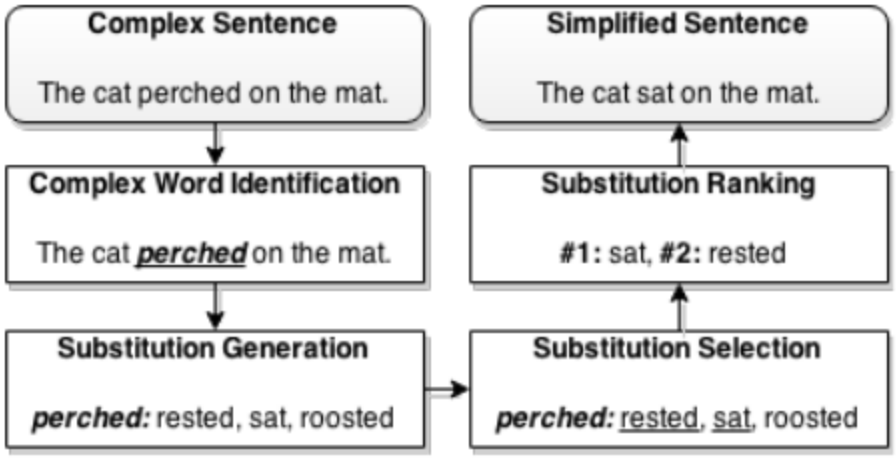


Figure 3-1: Illustration of Lexical Simplifier Pipeline. Retrieved from [16]

In the first notable approach for automatic lexical simplification, Devlin and Tait [43] rank synonyms from the semantic thesaurus WordNet 3.0 [44] using Kučera-Francis frequency [45], which is the counts of words from the Brown corpus. The brown corpus consists of just over one million words from 50 sources which are intended to be representative of the English language. However, recent approaches involve much larger corpora of text to better estimate the frequency counts of words.

This work, by Delvin and Tait, introduced a base for Lexical Simplification that greatly influenced later LS systems. Since each step of the pipeline is independent and has its own methods, we will discuss proposed approaches of each one separately.

For the substitution generation task, the approach that was mostly used is extracting synonyms from linguistic databases, such as WordNet as in Delvin and Tait, 1998 [43] and Carroll et al., 1999 [46], or the UMLS database for medical content used in Ong et al., 2007 [47] and Leroy et al., 2013 [48]. In recent systems, however, other approaches are used such as using aligned complex-to-simple parallel corpora as in Paetzold, 2013 [49]; Paetzold and Specia, 2013 [50], and Horn et al., 2014 [54]. Also, using word embedding models for generating synonyms by Glavaš and Štajner, 2015 [55]; Paetzold and Specia, 2016 [52].

For substitution selection, the goal is to discard any candidates that do not fit the context of a target complex word. Sedding and Kazakov, 2004 and Nunes et al., 2013 treat this step as a disambiguation task in which sense labeling tools are used to discard any candidates that do not have the same context sense as that of the target complex word. However, modern approaches proved better performance by using co-occurrence models as in Biran et al., 2011 [59]. While Horn et al. and Glavaš and Štajner combined substitution selection with ranking.

The most widely used ranking strategy in the literature is metric-based ranking, that is, using manually crafted combination of features; such as word frequency and length. This was used in papers by Devlin and Tait, 1998; Carroll et al., 1998; Carroll et al., 1999; Biran et al., 2011; Bott and Saggion, 2011 [57]. More sophisticated approaches are recently introduced, however, in which the use of machine learning is explored to learn the ranking task, such as SVM rankers by Horn et al., and boundary ranking [56], which is building a binary classifier trained on manually annotated data, and the use of a neural network by Gustavo Paetzold and Lucia Specia [56] and Mounica Maddela and Wei Xu [15] in which the combination of more sophisticated features, other than frequency and length of words, are used.

Chapter 4
System Architecture

4.1 Overview

In our project, we provide a web-based language tool that can be used to simplify and summarize English texts. The user inserts the required text to be simplified to the tool and chooses the operation to apply on the text. The processed text then appears on the left side of the screen with a complexity assessment score before and after the process. In the following Figure 10, we demonstrate a screenshot taken from our tool.

4.2 Block diagram

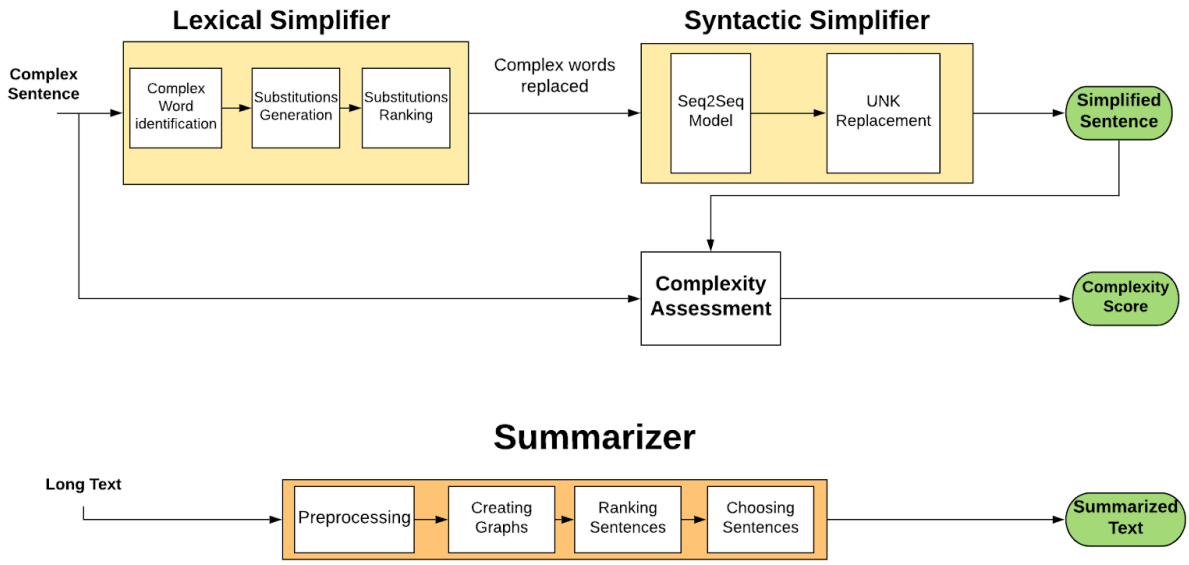


Figure 4-1: Block Diagram of Our System

The summarizer and simplification are presented in as two separate modules to provide the user with the freedom of choosing the operations they desire to perform on their texts. As for the simplifier, it consists of two modules, which are the lexical simplifier and syntactic simplifier, which run consecutively to provide the final simplified output. The lexical simplifier runs first to replace complex words in the text and therefore provides a less complex version of the sentence for the Seq2Seq model to work with and therefore increases the quality of the syntactic simplifier output. A sample of sentences given by the different ordering of modules is given Table 4-1 below.

	Sentence 1	Sentence 2
Source	The popcorn moves like a gymnast doing a somersault .	Teams with third-graders -- even second-graders -- are popping up .
Lexical then Syntactic Simplification	The corn moves like an athlete doing a flip .	Teams with third-grades are popping up .
Syntactic then Lexical Simplification	The ice moves like an athlete .	Teams with third-binders are popping up .

Table 4-1: Simplifier Output with Different Modules Ordering

We compare our results with other state-of-the art simplifiers on the Newsela data set. Table 4-2 below compares the BLEU and SARI scores of our suggested simplifier versus other systems.

Model	BLEU	SARI
DRESS	23.21	27.37
DRESS -LS	24.30	26.63
HYBRID	14.46	**30.00**
Our Model	**24.84**	28.04

Table 4-2: Scores of Our Model Vs. Other Simplifiers

We also evaluated random sentences against the output of other systems to see the advantages and disadvantages of our system compared to the other systems.

Complex: Twelve percent of the world 's land is currently protected in national parks and wildlife preserves , while in comparison , just about 1 percent of the high seas is protected .
HYBRID: twelve percent is protected about 1 percent is protected .
DRESS: Nearly **half** of the world 's land is **now** protected .
DRESS -LS: Nearly **half** of the world 's land is **now** currently protected .
Our Model: There is about 1 percent of the high **waters** is protected .

Table 4-3: Our System Output Vs. Other Systems

Chapter 5
Modules

5.1 Summarizer

Summarizer is responsible for producing a shorter version of the document by removing the details from the document and selecting the most important sentences in the documents. The summarizer module is classified as an extractive summarizer and it follows the TextRank algorithm introduced by Mihalcea and Tarau [4].

5.1.1 Inputs

The summarizer takes as an input the raw text that needs to be summarized and the desired compression ratio. The compression ratio is defined as Summary length/Original text length.

5.1.2 Preprocessing

In this step we prepare the input document for the TextRank algorithm over four steps.

1. The text is split into a list of sentences.
2. All stop words, punctuation marks, numbers, tags and extra white spaces are removed from the sentences.
3. All words are stemmed using Porter Stemmer
4. Number of occurrences of a stem in each sentence is counted

5.1.3 TextRank Algorithm

TextRank is an unsupervised graph-based ranking algorithm that decides the importance of each vertex (sentence) within the graph recursively based on votes from other vertices. TextRank uses weighted undirected graphs for calculating the importance of each vertex.

5.1.3.1 Creating the Graphs

Each sentence is treated as a vertex in the graph. Each vertex is connected to all vertices in the graph by a weighted undirected edge. Unlike the originally proposed algorithm which uses the conventional IDF discussed in Chapter 2 we use the modified IDF as proposed by Barrios, et.al [5] to calculate the graph weights. Edge weights are calculated on two steps:

1. Calculate the modified IDF for each word as the following equation:

$$IDF(s_i) = \begin{cases} log(N - n(s_i) + 0.5) - log(n(s_i) + 0.5) & \text{if } n(s_i) > N/2 \\ \varepsilon \cdot avgIDF & \text{if } n(s_i) \leq N/2 \end{cases}$$

Where N is the number of sentences in the document, n(s_i) is the number of sentences containing the word s_i, avgIDF is the average IDF of all words in the document and Ɛ is a constant set to 0.25 as suggested by Barrios,et.al[5].

2. Calculate the weight between sentence 1 and sentence 2 using the following ranking function:

$$Score(sentence1, sentence2) = \frac{IDF(s_i) * f(s_i, sentence1) * (k+1)}{f(s_i, sentence1) + k * \left(1 - b + \frac{b * L(sentence1)}{avgL}\right)}$$

Where $f(s_i, sentence1)$ is the frequency of word s_i in sentence 1, L(sentence 1) is the length of sentence 1, avgL is the average length of all sentences in the document and k and b are constants set to 1.2 and 0.75 respectively.

This function is known as BM25 which the state-of-the-art variation from TF-IDF used in information retrieval [5].

5.1.3.2 Ranking Sentences

Importance scores of sentences is calculated recursively by passing over the graph multiple times and updating the graph. Calculation is stopped if the scores of all sentences converge or the maximum number of iterations over the graph is reached. Initially all sentences have the same importance score which is $\frac{1}{numberofsentences}$ .Scores for each sentence is calculated recursively using the following equation:

$$score(i)=(1-d)+d * score(i) * \sum_{j=1, j\neq i}^{n} \frac{w(j,i)}{\sum_{k=1, k\neq j}^{n} w(j,k)}$$

Where w (j, i) is the edge weight between sentence i and sentence j, score(i) is the importance score given to sentence i and d is the damping ratio set to 0.85. Convergence of a node is achieved if the difference between the previously calculated score and the current score is less than or equal the convergence threshold which is set to 0.0001.

5.1.3.3 Choosing Sentences

Sentences are sorted according to their scores. Top sentences are added to the summary until the length of the summary complies with the compression ratio desired. To maintain logical order sentences are not placed in the final summary according to their scores order, but rather according to their order in the original text. Scores are only used for deciding which sentences to include in the summary.

5.1.4 Evaluation

5.1.4.1 ROUGE

ROUGE stands for Recall-Oriented Understudy for Gisting Evaluation. It includes measures to automatically determine the quality of a summary by comparing it to other (ideal) summaries created by humans [27]. It ranges from 0 to 1 where 0 is the lowest score and 1 the highest. There are four major variations of ROUGE Score: ROUGE-N, ROUGE-L, ROUGE-W, ROUGE-S. In our work, we used both ROUGE-N and ROUGE-L for assessment.

ROUGE-N measures the n-gram recall between the system summary and the reference summary. This is done by dividing the clipped count of the matching n-grams between the system and the reference summary by the total count of n-grams in the reference summary. The clipped count means that if the n-gram occurs in both the system and the reference, only the minimum count of this n-gram in either the reference or the system is included in the following equation.

$$Rouge{-}N = \frac{\sum_{m}^{matching\ ngrams} Count_{clipped}(ngrams)}{\sum_{n}^{reference\ ngrams} Count(ngrams)}$$

ROUGE-L accounts for the similarity between the reference and the system's summary using the longest common subsequence between them.

$$Rouge - L = \frac{(1+\beta^2)R_{lcs}P_{lcs}}{R_{lcs} + \beta^2 P_{lcs}}$$

Where LCS is the longest common subsequence and

$$R_{lcs} = \frac{LCS(reference\ summary, system\ summary)}{length\ of\ reference\ summay}$$

$$P_{lcs} = \frac{LCS(reference\ summary, system\ summary)}{length\ of\ system\ summay}$$

5.1.4.2 Results

To test the summarization module, we used the publicly available BBC news summary dataset which contains 2225 articles and their handcrafted extractive summaries. Articles are divided into five categories which are sports, tech, business, entertainment and politics. To create our test set we chose 100 articles from each category at random. We produced ROUGE scores at two compression factors (0.4 and 0.2) and compared our results with that produced by the gensim library implementation of [5].

Compression Factor	Score	Our system	Gensim
0.2	ROUGE-1	0.70.36	0.6019
	ROUGE-2	0.6174	0.5041
	ROUGE-L	0.6590	0.5214
0.4	ROUGE-1	0.7924	0.7420
	ROUGE-2	0.7219	0.6610
	ROUGE-L	0.7812	0.7293

Table 5-1: Comparison between Our System and Gensim

Table 5-1 shows that the average ROUGE scores for our implementation exceeds that of gensim library implementation for the same input. It is also noted that that scores did not fall below 0.6 even for a small compression factor (i.e. a small summary) which indicates that the system picks the most representative sentences within the given summary length.

A sample of the system output at 0.4 compression rate is shown below.

Original Text:

Godzilla gets Hollywood fame star

Movie monster Godzilla has received a star on Hollywood's Walk of Fame, honouring both his 50th birthday and the launch of his 28th film.

An actor dressed as the giant creature breathed smoke over photographers on Monday as Godzilla received the 2,271st star on Hollywood Boulevard. "Godzilla should thank you for this historical and monumental star," said Final Wars producer Shogo Tomiyama. "But unfortunately, he cannot speak English," he added. Hollywood's honorary mayor, Johnny Grant, said: "I do hereby proclaim this Godzilla Day in Hollywood.

"He's loose, he's wild, and I'm getting the hell out of here," he added. The premiere of Godzilla: Final Wars at Grauman's Chinese Theatre followed the ceremony on Hollywood Boulevard. The monster was joined by co-stars including Japanese pop star and actor Masahiro Matsuoka. Director Ryuhei Kitamura said it may not be Godzilla's final outing, as it has been billed. "That's what the producers say. But the producer's a liar," he said. "[Godzilla's] been working for the

last 50 years. So, I think Godzilla just deserves a vacation." And producer Shogo Tomiyama added: "So long as Godzilla can fascinate people, I believe he will be resurrected by new generations of filmmakers in the future." Godzilla first appeared in 1954 as a prehistoric lizard woken by atomic bomb tests.

Summary:

Godzilla gets Hollywood fame star

Movie monster Godzilla has received a star on Hollywood's Walk of Fame, honouring both his 50th birthday and the launch of his 28th film. An actor dressed as the giant creature breathed smoke over photographers on Monday as Godzilla received the 2,271st star on Hollywood Boulevard. "Godzilla should thank you for this historical and monumental star," said Final Wars producer Shogo Tomiyama. Hollywood's honorary mayor, Johnny Grant, said: "I do hereby proclaim this Godzilla Day in Hollywood. The premiere of Godzilla: Final Wars at Grauman's Chinese Theatre followed the ceremony on Hollywood Boulevard. Director Ryuhei Kitamura said it may not be Godzilla's final outing, as it has been billed. And producer Shogo Tomiyama added: "So long as Godzilla can fascinate people, I believe he will be resurrected by new generations of filmmakers in the future."

5.2 Syntactic Simplifier

This module performs simplification operations on the sentence level with the aim to reduce the linguistic complexity of the text while retaining the original information and meaning. These operations include substituting complex words with less complex words or phrases, deleting parts of the original text and coping parts of the original sentence.

5.2.1 Model Architecture

For syntactic simplification we used an attention based Seq2Seq model with a reinforcement framework. Both encoder and decoder use 2 layers of LSTMs with 400 hidden units. The architecture contains 2 word embedding layers that are used to represent the word/token in a 300 dimensional vector. It also contains a single layer neural network to map the 400 dimensional vector produced by the decoder into a vector whose dimension is equal to target (simple) vocab size.

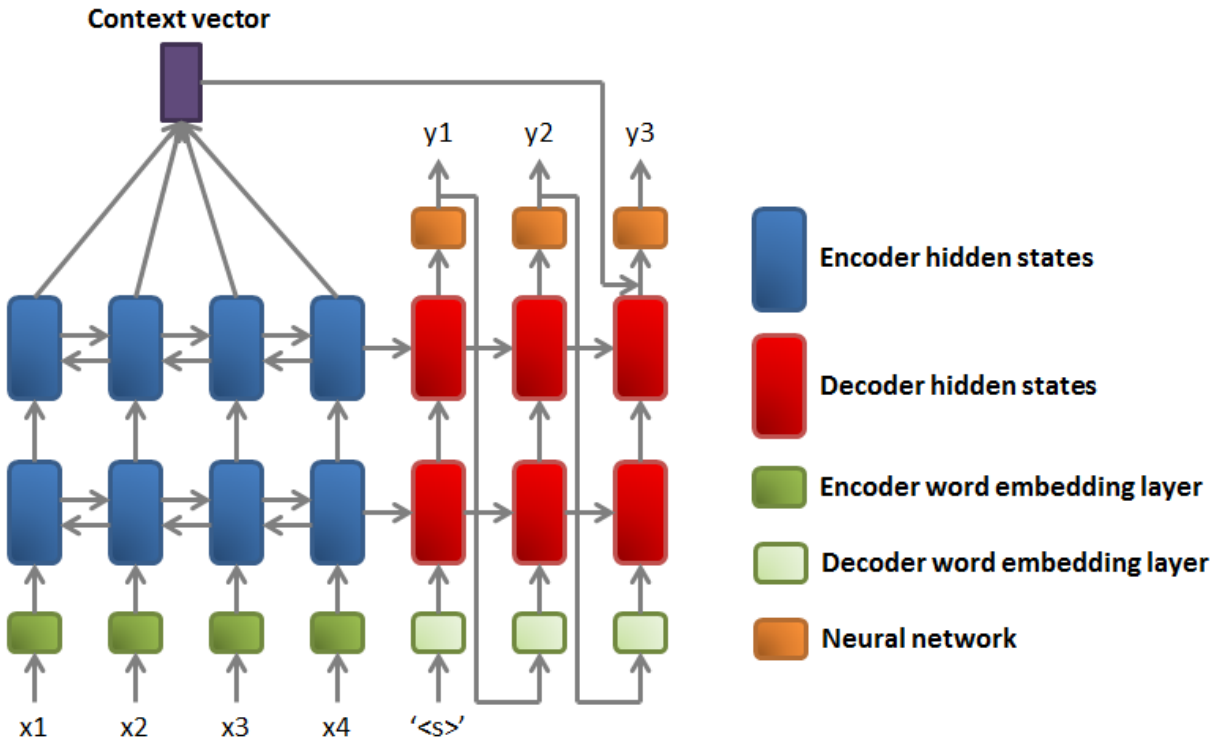


Figure 5-1: Proposed Model Architecture for Syntactic Simplification

The encoder is bidirectional; where each layer of the encoder is divided into 2 separate sub-layers each of 200 hidden units, the first one is fed the input in a natural order while the second one is fed in a reverse order. The final output of an encoder layer is the concatenation of the output of the 2 sub-layers. This structure allows the network to have forward and backward information about the input sequence to provide it with context.

The decoder applies input feed, where the output word of the last time step $y_{t-1}$is used as an input to generate the next output word y_t. During inference only, beam search is used with beam size 10.

For the attention mechanism, we applied the concat attention stated in Luong et al. (2015) where the score function is given by the equation in Figure 8.

To handle the problem of out-of-vocab words, we used UNK replacement where we replace the UNK token in the generated output with the word that had the highest attention score at the time step in which the UNK token was generated.

5.2.2 Data set

To train our model, we used monolingual parallel corpora of complex and simple sentences. We conducted our experiments on Newsela dataset [20], a high-quality simplification corpus of

1,130 news articles composed by professional editors at Newsela, a company that produces reading materials for pre-college classroom use [19]. Each article has been re-written 4 times for children at different grade levels where V0 is the original sentence, V1 the least simplified version and V4 the most simplified one. This data forms a parallel corpus, where sentences are aligned at different reading levels, as shown in Table 5-2.

Grade Level	Version	Text
12	V0	**Slightly more** fourth-graders **nationwide are reading proficiently compared with** a decade ago, but only a third of them are **now reading** well, according to a new report.
7	V1	Fourth-graders **in most states are better readers than they were** a decade ago. **But** only a third of them **actually are able to read** well, according to a new report.
6	V2	Fourth-graders **in most states** are better readers than they were a decade ago. But only a third of them actually are able to read well, according to a new report.
4	V3	**Most** fourth-graders are better readers than they were **10 years** ago. **But few** of them **can actually** read well.
3	V4	Fourth-graders are better readers than 10 years ago. But few of them read well.

Table 5-2: Example of Sentences Written at Multiple Levels of Complexities from Newsela Data Set. Words in Bold Highlights the Differences Between the Sentence and its Adjacent Version(s).

We followed the same split as Zhang and Lapata [9] where the first 1,070 documents were used for training (94,208 sentence pairs), the next 30 documents for development (1,129 sentence pairs) and the last 30 documents for testing (1,076 sentence pairs). In addition, we removed sentence pairs corresponding to levels 0–1, 1–2, 2–3 and 3-4, since they were too similar to each other.

5.2.3 Model Training

Typically, reinforcement learning starts with a random policy which could be challenging in our case due to the large number of trainable parameters and the large vocabularies used. To address this issue we pre-trained the Seq2Seq model with a negative log-likelihood objective function to guarantee that the model can produce good simplifications which is considered better than starting with a random policy.

5.2.3.1 Inputs

In addition to the training parameters, this module takes as an input:

1. Complex and simple training sentences
2. Complex and simple validation sentences
3. GloVe vectors to initialize the two word embedding layers

5.2.3.2 Preprocessing

We start by building 2 datasets one for training and the other for validation. Each dataset consists of pairs of complex and simple sentences. Since we treat the simplification process as a machine translation problem, we build two different vocabs from the source and target sentences of the training dataset. In order for the model to learn handling words not seen before, we restrict the vocab size to the top 20500 word based on the word frequency and removed words with frequency less than 3. Finally, we added 2 extra tokens to both vocabs: '<unk>' which is used to replace all out-of-vocab words (unknown) and '<blank>' which is used in padding the sentences to create a batch. In addition to the previous to tokens, we added 2 more to the target vocab only which are: '<s>' to indicate the beginning of a sequence and '</s>' to indicate the end of a sequence.

5.2.3.3 Training Details

We started by training the Seq2Seq model and then continued with the reinforcement learning training. The parameters of the encoder and decoder were initialized uniformly from [-0.1, 0.1]. We used Adam to optimize these parameters with learning rate 0.001, first momentum coefficient $\beta_1 = 0.9$and second momentum coefficient$\beta_2 = 0.999$. To avoid exploding gradients, we rescaled when the norm exceeded 5. We initialized both embedding layers with 300d GloVe vectors. The training step was applied on a batch of size 32. As for the regularization, we applied 0.3 dropout rate between LSTM layers of both encoder and decoder.

As for the reinforcement learning, we used SGD optimizer with learning rate 0.01. The reward function $r(\hat{Y})$ is the weighted sum of SARI score $r^S$and ROUGE scorer^R :

$$r(\hat{Y}) = \lambda^S r^S(X, \hat{Y}, Y) + \lambda^R r^R(\hat{Y}, Y)$$

where X is the complex sentence, Y the reference (simple) sentence and $\hat{Y}$ the action sequence (simplification) produced by the model. The ROUGE score is calculated using both ROUGE-2 and ROUGE-L using the following equation:

$$r^R = \beta_1 \text{ROUGE-2}(\hat{Y}, Y) + (1 - \beta_1)\text{ROUGE-L}(\hat{Y}, Y)$$

We set the weights as follows: $\lambda^S = 0.5$, $\lambda^R = 0.9$ and $\beta_1 = 0.15$.

In order to pre-train the model using supervised learning, we applied early stopping with patience set to 2 based on the validation set perplexity; early stopping is a technique applied to avoid overfitting by stopping the training before the weights converge and patience is the number of epochs to wait before early stop if no progress on the validation set. We pre-trained the model for 5 epochs and then added 7 more epochs with reinforcement learning.

The final model parameters are

- Complex vocab size: 20502 token
- Simple vocab size: 20229 token
- Total trainable parameters: 25,785,129 parameter
    - Encoder: 7,917,000 parameter
    - Decoder: 17,868,129 parameter

5.2.4 Evaluation Metrics

5.2.4.1 SARI

SARI is a metric that was specifically designed for Text Simplification. It ranges from 0 to 1 where 0 is the lowest score and 1 the highest. It does so by comparing **s**ystem output **a**gainst **r**eferences and against the **i**nput sentence [29]. This way it evaluates the three major text simplification operations: addition of, deletion of and keeping words. For each operation, both precision and recall are called. This concept is illustrated in the following diagram.

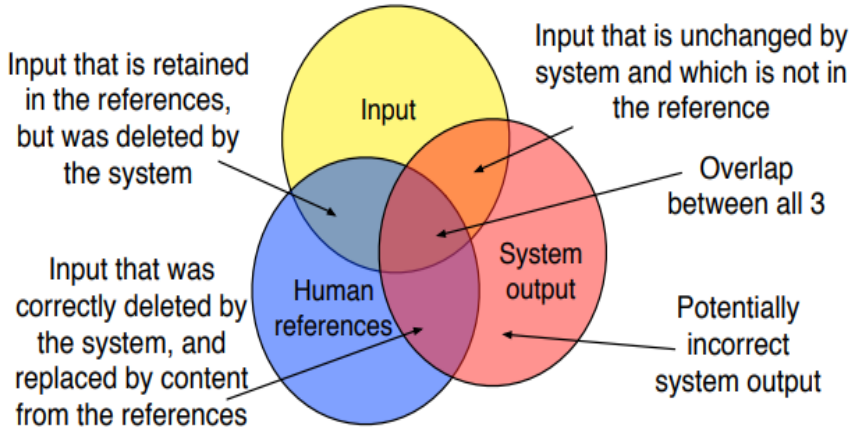


Figure 5-2: Illustration of the Concept Behind SARI. Retrieved from [29]

The first operation, addition, is scored according to the following equations:

$$p_{add}(n) = \frac{\sum_{g \in O} \min\left(\#_g(O \cap \bar{I}), \#_g(R)\right)}{\sum_{g \in O} \#_g(O \cap \bar{I})}$$

$$r_{add}(n) = \frac{\sum_{g \in O} \min\left(\#_g(O \cap \bar{I}), \#_g(R)\right)}{\sum_{g \in O} \#_g(R \cap \bar{I})}$$

O: System Output, I: Input, R: Reference, #g(.): a binary indicator of occurrence of n-grams g in a given set and

$$\#_g(O \cap \bar{I}) = \max(\#_g(O) - \#_g(I), 0)$$

$$\#_g(R \cap \bar{I}) = \max(\#_g(R) - \#_g(I), 0)$$

In other words, for each n-gram in the system's output the precision numerator counts the minimum of the difference between the count of the ngram in the output and the input and the count of the ngram in the reference. While the denominator sums all the counts of the n-grams that were present in the output but not the input. The recall calculates the same numerator but divides it with the sum of all the counts of the n-grams that were present in the reference but not the input.

The second operation, keeping words that were also kept in the reference, is rewarded using the formulas below.

$$p_{keep}(n) = \frac{\sum_{g \in I} \min\left(\#_g(I \cap O), \#_g(I \cap R')\right)}{\sum_{g \in I} \#_g(I \cap O)}$$

$$r_{keep}(n) = \frac{\sum_{g \in I} \min\left(\#_g(I \cap O), \#_g(I \cap R')\right)}{\sum_{g \in I} \#_g(I \cap R')}$$

Where R' represents the ratio between the number of the references that the n-gram has appeared in over the total number of references (r) and

$$\#_g(I \cap O) = \min\left(\#_g(I), \#_g(O)\right)$$

$$\#_g(I \cap R') = \min\left(\#_g(I), \#_g(R)/r\right)$$

The deletion operation is only scored using precision because over-deletion negatively affect readability. The same R' notion is used in its calculation.

$$p_{del}(n) = \frac{\sum_{g \in I} \min\left(\#_g(I \cap \overline{O}), \#_g(I \cap \overline{R'})\right)}{\sum_{g \in I} \#_g(I \cap \overline{O})}$$

where

$$\#_g(I \cap \overline{O}) = \max\left(\#_g(I) - \#_g(O), 0\right)$$

$$\#_g(I \cap \overline{R'}) = \max\left(\#_g(I) - \#_g(R)/r, 0\right)$$

Finally, the equation to calculate SARI is as follows:

$$SARI = d_1 F_{add} + d_2 F_{keep} + d_3 P_{del}$$

where $d_1 = d_2 = d_3 = 1/3$ and

$$P_{operation} = \frac{1}{k} \sum_{n=[1,\ldots,k]} p_{operation}(n)$$

$$R_{operation} = \frac{1}{k} \sum_{n=[1,\ldots,k]} r_{operation}(n)$$

$$F_{operation} = \frac{2 \times P_{operation} \times R_{operation}}{P_{operation} + R_{operation}}$$

$$operation \in [del, keep, add]$$

Where K is the highest n-gram order.

5.2.4.2 BLEU

Originally developed by Papineni et al [26], BLEU (Bilingual Evaluation Understanding) is the most widely used metric for machine translation, and thus it has been used by several text simplification systems for evaluation. It ranges from 0 to 1 where 0 is the lowest score and 1 the highest. f It aims to measure the systems' precision and the output's fluency by counting the number of matches between the n-grams of the system's output and the n-grams of a reference output. [26] This reference output is the human-generated simplified text that we compare our system against. The way of doing so is by summing up the number of all matching n-grams and dividing them by the total number of n-grams in the reference simplification. To avoid scoring system's dummy output of repeated words; the number of matching n-grams is clipped to the maximum number of its matching n-grams in the reference. This is illustrated in the following equation.

$$P_n = \frac{\sum_m^{matching\ ngrams} Count_{clipped}\ (ngrams)}{\sum_n^{system\ nrgrams} Count\ (ngrams)}$$

Then, the Bleu score is calculated as

$$\text{BLEU} = \text{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right)$$

Where N is the maximum number of N-grams we are considering, Wn are positive weights for each n-gram that sum to 1 and BP is the Brevity Penalty, which penalizes short system output, is computed as

$$\mathrm{BP} = \begin{cases} 1 & \text{if } c > r \\ e^{(1-r/c)} & \text{if } c \leq r \end{cases}$$

Where r is the reference output length and c is the system output length.

5.2.5 Results

Following previous work (Xu et al., 2016) we evaluated system output automatically adopting metrics widely used in the simplification literature [14]. Specifically, we used BLEU and SARI that are measured at corpus level. We compared our proposed model against DRESS, the model proposed by Zhang and Lapata [9]. Our model performs better by 2.58 according to BLEU while DRESS scores a little higher by 0.4 according to SARI yet our model without reinforcement learning has the highest SARI score, as shown in Table 5-3. Table 5-4 illustrates the system outputs versus each other.

Model	**BLEU**	**SARI**
EncDecA	21.70	24.12
Our Model without reinforcement	24.95	**27.79**
DRESS	23.21	27.37
Our Model	**25.79**	26.97

Table 5-3: Our Model Vs. DRESS According to BLEU and SARI Scores

Type	**Sentence**
Source	A succession of steadily more powerful and flexible computing devices were constructed in the 1930s and 1940s , gradually adding the key features that are seen in modern computers .
Reference	Computing devices became more powerful and flexible during the 1930s and 1940s .

EncDecA	A period of rapidly more powerful and flexible computing devices were constructed in the 1930s and 1940s .
Our Model without reinforcement	More powerful and advanced computers were constructed in the 1930s and 1940s.
DRESS	A period of rapidly more powerful and flexible computing devices were constructed in the 1930s and 1940s .
Our Model	More powerful and simpler computers were made in the 1930s and 1940s .

Type	**Sentence**
Source	Being much more resistant to cold and moist weather conditions , brick enabled the construction of permanent buildings in regions where the harsher climate precluded the use of mud bricks .
Reference	Fired bricks were much more resistant to cold and moist weather conditions. Fired bricks enabled the construction of permanent buildings in regions where the harsher climate precluded the use of mud bricks .
EncDecA	As much more resistant to cold and wet weather conditions , brick changed the construction of permanent buildings in regions where the harsher climate precluded the use of mud bricks .
Our Model without reinforcement	People used to make milk more resistant .
DRESS	Being much more resistant to cold and wet weather conditions , brick changed the construction of permanent buildings in regions where the harsher climate precluded the use of mud bricks .
Our Model	The construction of permanent buildings in regions where the tougher climate precluded the use of mud bricks .

Table 5-4: Sample of Our Model's Output Vs. DRESS

To determine the beam size, we tested our proposed model with different beam sizes on Newsela test set. As illustrated by Figure 14, as the beam size increases both BLEU and SARI scores increase but on the other hand the computation time increases which will cause a significant delay in the overall process. Table 5-5 shows a sample of the output using these beam sizes.

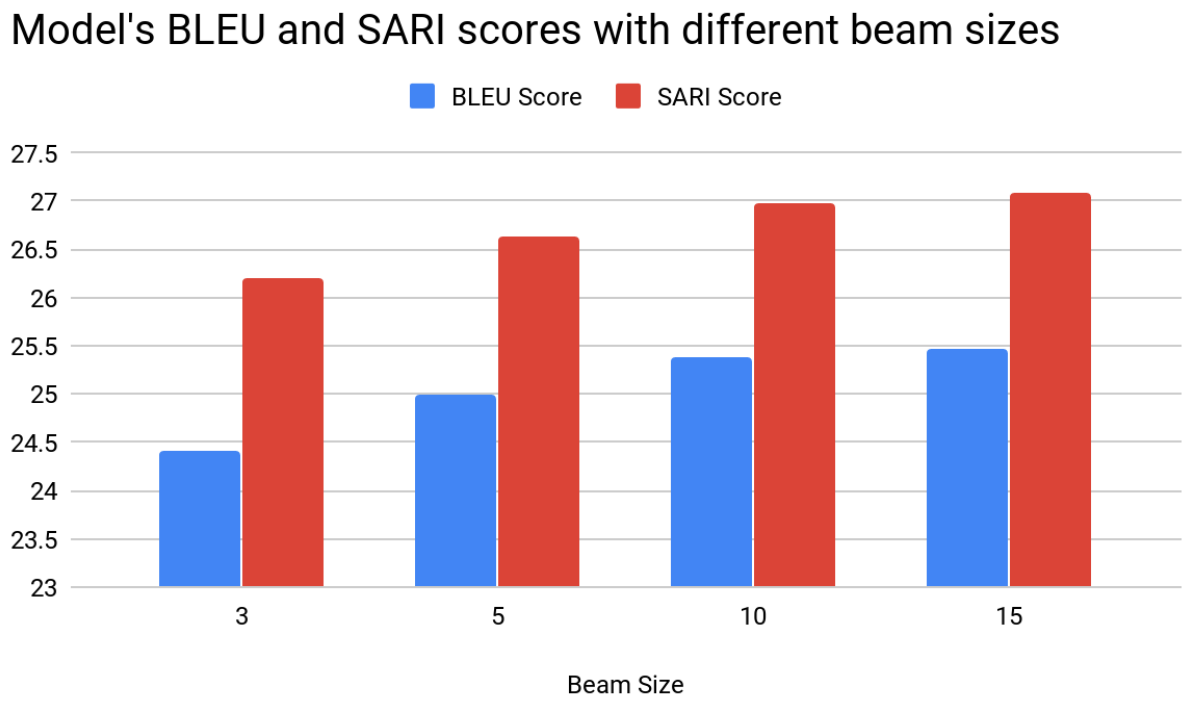


Figure 5-3: Our Model's BLEU and SARI Scores for Different Beam Sizes

Type	Sentence
Source	The city had plans to build the Clinton library in a decaying warehouse district on the edge of downtown .
Reference	Little Rock , Arkansas , wanted to build the Clinton library .
Beam Size 5	The city had plans to build the library library in a decaying city district on the edge of downtown .
Beam Size 10	The city had plans to build the Clinton library .
Beam Size 15	The city had plans to build the Clinton library .

Table 5-5: The Simplified Output at Different Beam Sizes

Our model also has its flaws that are observed through the output sentences. The most observed problem is replacing named entities with other named entities (usually the replaced entity is of the same type as the original, i.e. a person name is changed with another name) or replacing words with its antonym. A sample of the faulty outputs is given in Table 5-6 below.

Original Sentence	**System Output**
Gary Goddard is the founder of Gary Goddard Entertainment .	It is the founder of the **British** Goddard Entertainment .
They sounded more like the Scottish chimps .	They sound more like the **Navarro** chimps .
He felt joy and happiness .	He felt joy and **awful** .
He watched the lava in 1991 as it devoured the house where he was born and raised and where his father still lived when the lava made its gradual way to the coast .	He saw the **mud** in 1991 .

Table 5-6: System Faulty Output. Words Wrongly Replaced are Bolded

5.3 Lexical Simplifier

The aim of the lexical simplifier is to replace complex words with simpler ones. Such module works by completing three major modules: i) Complex Word Identification: which involves identifying complex words in the sentence; ii) Substitution Generation: which involves finding alternatives to complex words; iii) Substitution Ranking: which works on ranking the generated alternatives by simplicity [15].

Our work on the lexical simplifier has been influenced by the work of Mounica Maddela and Wei Xu [15] although there are major modules that have been greatly modified.

5.3.1 Data sets

Each module of the lexical simplifier required a separate set of datasets. For the Complex Word Identification module, we used lexicon; a dataset of 15,000 English words with their complexity rankings by 11 non-native but fluent human annotators [15]. We have also used Wordfreq [18], a Python library that has calculated frequencies of words from a collection of corpora: Wikipedia, OpenSubtitles 2018, SUBTLEX, NewsCrawl 2014, GlobalVoices,Google Books Ngrams 2012. As for the Substitution Generation Module, we used PPDB [17], and the thesaurus corpus provided by OpenOffice.org [23]. For the Substitution Ranking module, we used BenchLS; a

lexical simplification dataset consisting of 929 instances each containing a sentence, a target complex word and several substitutions ranked by their simplicity by English speakers [16]. In our experiments, we have also used SubIMDB [25] which is a corpus collected from family movies and series subtitles and has been collected specifically for lexical simplification.

5.3.2 Complex Word Identification

To identify complex words, we have tried two approaches. The first is the machine learning module proposed in [15] which is elaborated in the experiments section. The second approach and the one that was implemented in our project is based on thresholding. Any word that has a lexicon score above 3.0 or that has zipf frequency more than or equal to 4, which indicates that it has a frequency of 10 per million words. We also skip named entities when identifying complex words.

5.3.3 Substitution Generation

For generating the candidates, we tried different datasets and thesaurus that are mentioned in the experiments section. The best source used was the API provided by datamuse [24]. However, we replaced it with the thesaurus provided by OpenOffice.org [23] and PPDB [17] to get instantaneous results. The example below compares the results before and after replacing the API. The final step in Substitution Generation is to transform all the generated candidates to the same POS form as the complex word and then pass them to the substitution ranking module.

Original: These functions also use scheduling priority to decide which thread gets to execute when there is contention.
API-generated Substitutes: These functions also use scheduling priority to decide which thread gets to run when there is argument.
Thesaurus-generated Substitutes: These functions also use programming focuses to decide which thread gets to continued when there is controversy
Original: The productivity of crops and pastures as well as the health of other vegetation declines as the saline watertable reaches their root zones.
API-generated Substitutes: The output of crops and crops as well as the health of other species falls as the salty lowering reaches their root zones.
Thesaurus-generated Substitutes: The factor of crops and pastures , as well as the health of other plant , declines as the salty watertable reaches their root zones .

Table 5-7: Substitution Generation Comparison Results for API and Thesaurus-generated Methods

5.3.4 Substitution Ranking

The final step in lexical simplification is to rank the generated substitutions according to their simplicity. This is done by training a neural network on a labeled set of features until the network learns how to rank a given set of words. We have also tried two implementations for this module; the first was based on the network proposed in [15] which trains the network with all possible pairs of a set of simplified candidates for a given word with the difference in the pairs' ranks as their labels [15].

5.3.4.1 Features

Alternatively, our approach trains the neural network with the ranked set of candidates for every complex word. To add the relativeness between candidates to the neural network; we scale the features of each set of candidates to their max absolute feature of every complex word before feeding them to the network. By doing so, we aim to imply to the network how the top-ranked candidate relates to the rest of the candidates in terms of features. The eight features that we use in this implementation are: the scores of both three-gram and five-gram language models collected from Newsela corpus [19] using only sentences which have complexity score higher than one, the lexicon score of the candidate, the number of characters in the candidate word, the number of syllables in a candidate word, the logarithmic frequency of the candidate in the aggregated corpus described in section 6.3.2, the cosine similarity between the candidate, the complex word to be replaced and the ratio of the candidate frequency in Simple Wikipedia and English Wikipedia from WikiSmall [13].

5.3.4.2 Training

We divided our dataset BenchLS [16] for testing and training with ratio 20:80 to give a total of 744 sentences for training with 6159 candidate words and 185 sentences for testing with 688 candidate words. All our results are reported on the test set.

5.3.5 Evaluation Metrics

Benchmarking Lexical Simplification Systems [16] presents the first real contribution to comparing lexical simplification systems in a systematic way. They do not only compare the performance of the system as a whole, but also system components individually, such as Substitution Generation, Selection and Ranking modules.

For the evaluation of any of the modules, the predictions are compared against the BenchLS substitutions and their ranks, namely gold standards.

5.3.5.1 Ranking Evaluation Metric

The metric used is TRank-at-n, which measures the proportion of times in which a candidate with a gold-standard rank $r \leq n$ was ranked first.

5.3.5.2 Lexical Simplification Evaluation Metrics

The evaluation metrics used are the following:

- Precision: Ratio with which the highest ranking candidate is either the target word itself or is in the gold standard.
- Accuracy: Ratio with which the highest ranking candidate is not the target word itself and is in the gold standard.
- Changed Proportion: Ratio with which the highest ranking candidate is not the target word itself.

5.3.6 Results

The authors of the paper Benchmarking Lexical Simplification Systems [16] introduced the evaluation dataset BenchLS and compared Lexical Simplification systems on the same data. We will borrow their results and compare them with those of our model.

5.3.6.1 Ranking Evaluation Results

For the ranker task, the authors split BenchLS in training and test sets containing 465 and 464 instances, respectively, which is an almost 50:50 split. For comparison purposes only and to avoid any bias, we split our data here into 55% training and 45% testing. However, the final model used, which we believe has the best performance, is the one with the 80:20 split for the training and testing data respectively.

As for the neural readability ranking (NRR) model proposed in [15], we trained and tested their model on the same training and testing data we used respectively. To ensure a fair comparison between our model and theirs, we also used the same Newsela language model, wiki frequency, lexicon files and wordfreq features for their Google frequency feature.

Our model outperforms all the other models for the ranker task in terms of TRank-at-n for n values 1 and 3, and ranks second for n = 2 with the NRR model [15] ranked first This is mainly due to the idea of scaling the features of each set of candidates of a complex word. The following Table 5-6 shows the TRanks for all the models.

Ranker	N = 1	N = 2	N = 3
Devlin [46]	0.457	0.630	0.665
Biran [59]	0.472	0.617	0.707
Bott [57]	0.519	0.657	0.704
Yamamoto [62]	0.435	0.583	0.674
Horn [54]	0.539	0.694	0.737
Glavas [55]	0.526	0.674	0.746
Paetzold [52]	0.547	0.701	0.743
NRR [15]	0.718	**0.9945**	0.9945
Our Model	**0.841**	0.96	**0.995**

Table 5-8: Substitution Ranking Benchmarking Results

5.3.6.2 Complete Lexical Simplification Evaluation Results

For the evaluation of the complete pipeline, the following table shows the two metrics described in the evaluation metrics section for lexical simplification, namely precision and accuracy on different models. Our model has outperformed all other models and achieved both highest accuracy and precision when using a zipf_frequency equal 4.8. The zipf_frequency variable is used to identify complex words in a sentence, where a higher zipf_frequency allows more words to be identified as complex. Thus, when increased, the complex words identified in the test set are captured and replaced by simpler ones. This proves that our model is better at substitution generation than other models. However, based on our observations and the BLEU and SARI scores obtained when combining lexical and sentence simplifiers, we found that using a zipf_frequency of 4 has better results.

System	Precision	Accuracy	Changed
Devlin [46]	0.309	0.307	0.998
Biran [59]	0.124	0.123	0.999
Yamamoto [62]	0.044	0.041	0.997
Horn [54]	0.546	0.341	0.795
Glavas [55]	0.480	0.252	0.772
Paetzold [52]	0.416	0.416	**1.000**
Our Model (zipf_frequency = 4)	**0.756**	0.357	0.611
Our Model (zipf_frequency= 4.8)	0.584	**0.514**	0.93

Table 5-9: Complete Lexical Simplifier Benchmarking Results

The following Table 5-9 summarizes our experiments with different scaling techniques for the features, and shows that scaling each feature to its max absolute value - the max absolute scaler- has yielded both highest accuracy and Pearson Correlation. In addition, we have tried SubIMDB [25] for the language model and found that Newsela Corpus [19] yielded better results. We have also measured the accuracy over the training set to detect overfitting and found the accuracy to be 91.79%. Accuracy here is measured as the ratio of times our prediction rankings match the gold standards rankings to the total number of predictions.

Scaling Technique	Language Model Corpus	Accuracy	Pearson Correlation
Max Absolute Scaling	Newsela (sentences of complexity score above one)	**93.47%**	0.36

Min Max Scaling	Newsela (sentences of complexity score above two)	84.7%	0.37
Scaling to the mean and standard deviation	Newsela (sentences of complexity score above two)	77.7%	**0.415**
No scaling	Newsela (sentences of complexity score above two)	57.6%	0.33
Max Absolute Scaling	SubIMDB	91.3%	0.375

Table 5-10: Accuracy and Pearson Correlation for Different Scaling Techniques Applied on Our Model

A sample of the results produced by our model:

Original: When the leg is released, the ice moves like a gymnast somersaulting.
After Lexical Simplification: When the leg is released , the ice moves like an athlete flip.

Original: however , people often endured inadequate AT because of the lack of a viable alternative .
After Lexical Simplification: however , people often suffered inappropriate AT because of the lack of a sustainable option .

Original: the emancipation of women can only be completed when a fundamental transformation of living is effected ; and life-styles will change only with the fundamental transformation of all production and the establishment of a communist economy .
After Lexical Simplification: the release of women can only be completed when an essential change of living is affected ; and life-styles will change only with the basic change of all act and the constitution of a communist economy

Table 5-11: A Sample of Results Produced by the Lexical Simplification Model

5.4 Complexity Assessment

Assessment of the complexity level of text is an essential stage before and after simplification. It is used after the simplification process to determine the effect of simplification on the given text. Our module extracts a set of 33 features from a given labeled corpus, feeds them to a neural network that learns how to assess the complexity level of the text; it is based on the model suggested by [38].

5.4.1 Data

We have used Newsela [19] corpus for this module. The Newsela Corpus consists of 100K unique sentences that have complexity levels ranging from 0 to 4, with 0 being the most complex and 4 the most simple. We have performed cleaning on the data, as we observed that some sentences are labelled complex while they are in fact very simple. This noise is because the sentences are all extracted from articles of the same complexity levels ; as a result, there were simple sentences in complex articles that were mistakenly considered as complex. To eliminate their effect, we compared the sentence length of all the sentences of different complexity levels and eliminated all sentences from level 0 and 1 that were had less than 17 words - the average sentence length in level 4-. The resulting dataset had a total of 89K sentences with the following distributions among complexity levels (from 0 to 4): 19K, 13K, 21K 19K, 15K.

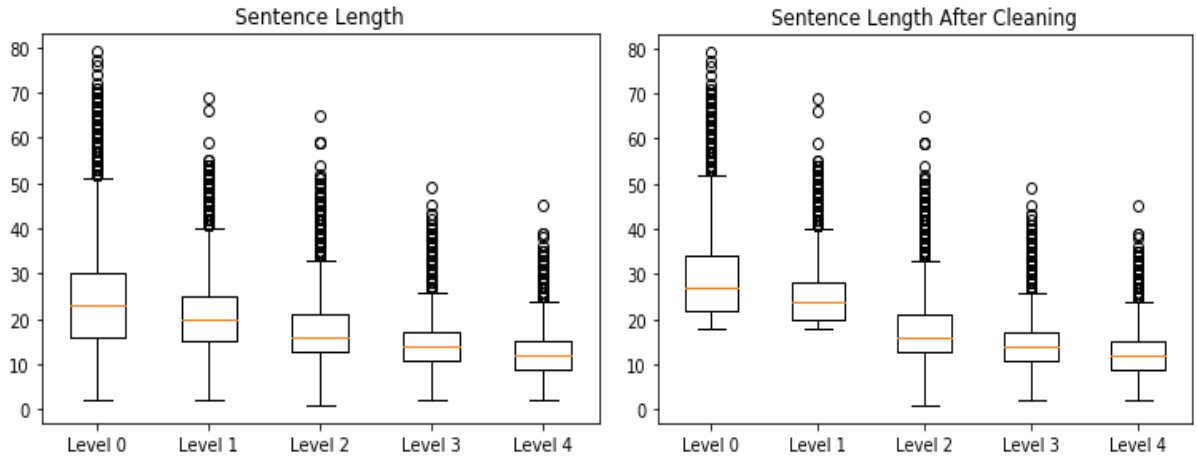


Figure 5-4: Two Box Plots of Sentence Length Before and After Data Cleaning

5.4.2 Features

To build our model we used a collection of different types of features: traditional, lexical, and syntactic features.

5.4.2.1 Traditional Features

The following features have been included in our model: *average number of characters per word*, *average number of syllables per word*, and *sentence length*. Previously, mathematical formulas have been used to measure the readability of text. They are sometimes referred to as "shallow formulas" because they rely on very simplistic factors in the calculations such as the number of words, the number of characters or the number of syllables. Nevertheless, we have plotted these scores across various complexity levels and found that they can have a significant role in complexity assessment; therefore, we used Flesch's Reading Ease, Flesch-Kincaid Grade Level and Coleman Liau [39] Index as features. The formulas for these scores are described below:

Flesch Reading Ease = $$206.835 - 1.015\left(\frac{\text{total words}}{\text{total sentences}}\right) - 84.6\left(\frac{\text{total syllables}}{\text{total words}}\right)$$

Flesch-Kincaid Grade Level = $$0.39\left(\frac{\text{total words}}{\text{total sentences}}\right) + 11.8\left(\frac{\text{total syllables}}{\text{total words}}\right) - 15.59$$

$$CLI = 0.0588L - 0.296S - 15.8$$

Where **L** is the average number of letters per 100 words and **S** is the average number of sentences per 100 words.

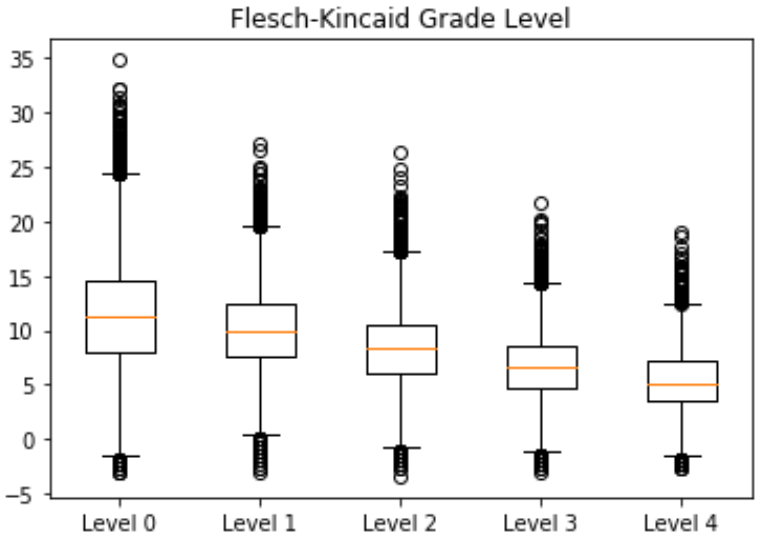


Figure 5-5: Box Plot of One of The Readability Metrics

5.4.2.2 Lexical Features

We have used the *type-token ratio (TTR)* which is the ratio of number of word types (T) to total number word tokens in a text (N). It has been widely used as a measure of lexical diversity or lexical variation in language acquisition studies [37]. There are also other variations of this metric that make it independent of text size such as *Root TTR* = $T/\sqrt{N}$, and *Uber Index TTR* = $Log^2 T / Log(\frac{N}{T})$. We have also included the *Measure of Textual Lexical Diversity (MTLD)* in our features which measures the mean length of string sequences that maintain a default Type-Token Ratio value. That is, the TTR is calculated at each word. When the default TTR value (0.72) is reached, the MTLD count increases by one and TTR evaluations are again reset [38]. Other features that represent the lexical variation in the text are *noun, adjective, modifier, and adverb and verb variations***.** These features are calculated as the ratio of the words of the respective categories to all the words in the sentence. Another feature we used is the *Lexical Density*, which is the ratio of the number of lexical to the total number of words in a text. To measure the level of complexity of the words used in the sentence, we used the *ratio of words in the Academic Word List;* this is a list that contains the most frequent words found in academic texts [41]. We have also included the *average frequency of words* [18] in the sentence, *the average lexicon score [15]* and the *number of complex words in the sentence* as features. We have found the latter to vary significantly among different complexity levels.

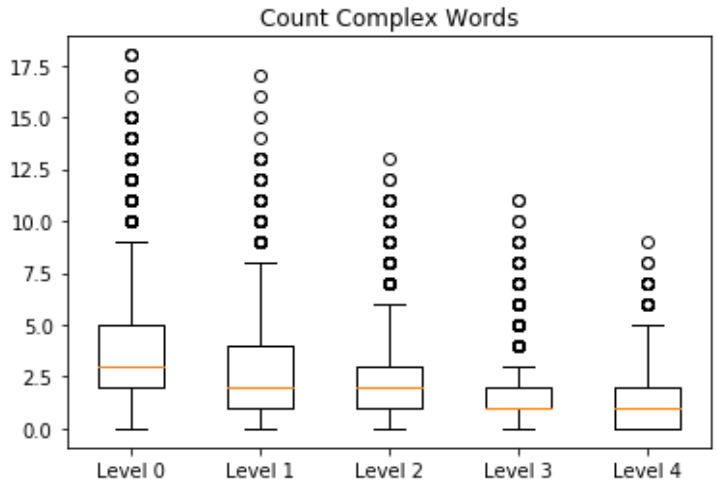


Figure 5-6: Box Plot of One of the Features Used: Count of Complex Words

5.4.2.3 Syntactic Features

We have also included features that relate to the syntactic structure of the sentence such as the parse tree height, average length of clause, the number of clauses per sentence, the number of verb phrases per sentence, the average length of verb phrases, the number of complex noun phrases per sentences, the number of prepositional phrases, the number of the dependent clauses

in the sentence, the number of different syntactic typesand the number of coordinating conjunctions in the sentence.

5.4.3 Results

The Newsela dataset was divided with ratio 80:20 for training and testing respectively. We calculated the accuracy in three different ways. First, we merged the top two levels of complexity and the lowest two level in complexity and calculated the result. Second, we calculated the exact accuracy on the datasets before and after cleaning. Third, since we found that the adjacent levels in Newsela often only differ in a word or two , we gave a range for the accuracy calculation to allow it for 1 level error and reported the accuracy on the dataset.

Description	**Accuracy**	**Approximated Accuracy**
Merging levels 0 ,1 and levels 3,4 to one level to result in only 3 levels of complexity	69%	69%
Using 5 levels without cleaning the data	37.4%	74.2%
After data cleaning	42.7%	83.5%.

Table 5.4 Accuracy of the complexity module

Chapter 6
Experiments

6.1 Syntactic Simplifier Experiments

6.1.1 Google Machine Translation Architecture

Inspired by Google's state-of-the-art neural machine translation we tried to create a model using the concepts introduced by Google neural machine translation (GNMT). We applied two main concepts suggested by Wu et al. [14] The first concept is residual connection between LSTM layers. Wu et al. [14] state that deep stacked LSTMs often give better accuracy over shallower models, but the network becomes too slow and difficult to train, likely due to exploding and vanishing gradient problems. To resolve this issue residual connections are used. Residual connections simply put is adding the output of a lower layer element wise with the input of that layer and use the result of addition as an input to the upper layer. The concept of residual connections is illustrated in Figure 18 below.

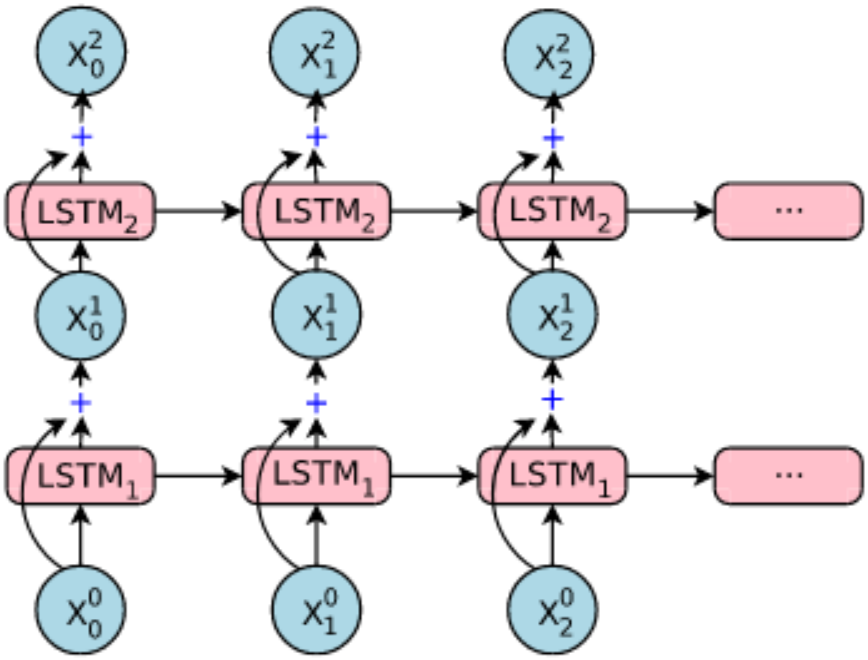


Figure 6-1: Residual Connections. Retrieved from [14]

The second concept we applied is using a bi-directional encoder for first layer. In order to capture information required to translate certain words on the output side can appear anywhere on the source side, Wu et al. [14] suggested using a bidirectional RNN for the first layer of the encoder. Other top layers of the encoder were kept unidirectional. The output of the forward LSTM and backward LSTM are concatenated and then fed to the next layer.

The suggested final architecture for GNMT was 8 layer encoder-decoder with attention and 1024 hidden units. The GNMT architecture is illustrated in Figure 19 below.

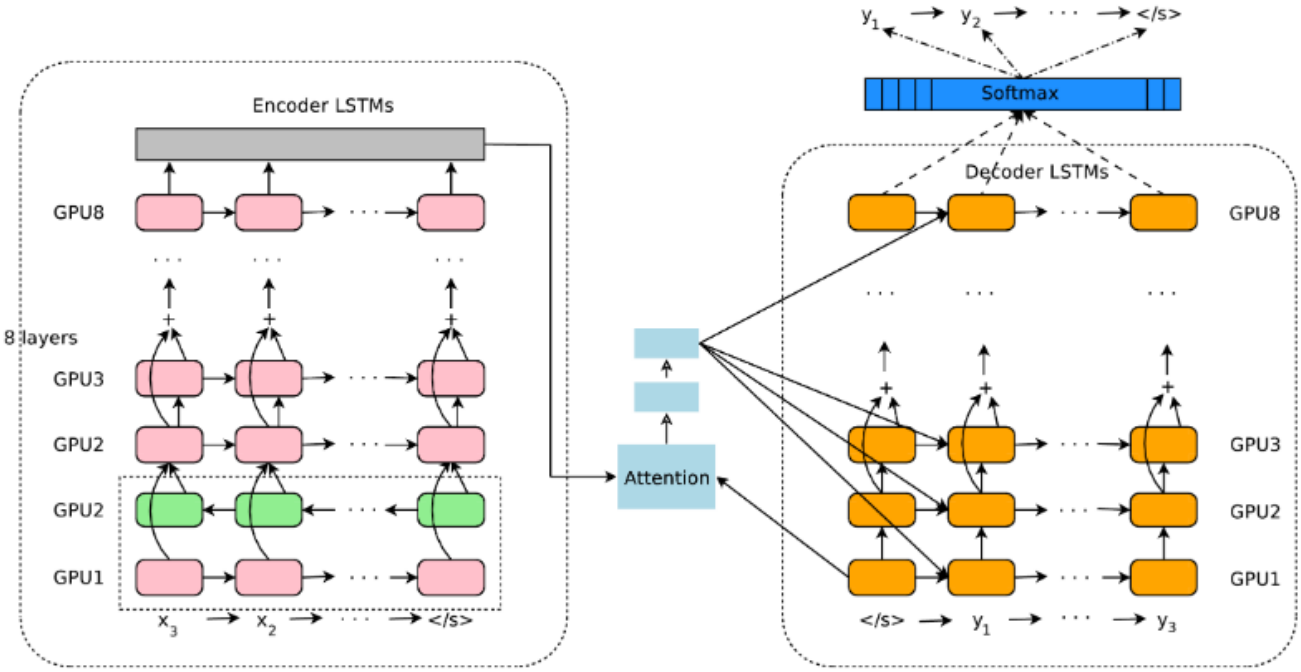


Figure 6-2: GNMT Architecture. Retrieved from [14]

This architecture is not feasible in our case due to the fact that the largest training set for simplification contains 94k example while for machine translation training set sizes can reach 36M sentence pairs. Therefore, we needed to have less trainable parameters. We tried out two models inspired by this architecture. One model was 4 layer encoder-decoder with attention and 256 hidden units and the other was 2 layer encoder-decoder with attention and 300 hidden units. The models were trained for 6 epochs using Adam optimizer with learning rate of 0.0005 and β1 and β2 set to 0.9 and 0.999 respectively and for 4 epochs using SGD with learning rate of 0.005.

The model was trained and tested on Newsela dataset and the resulting scores on the test set are given in Table 6-1 below.

Model	**BLEU**	**SARI**
GNMT 4 layers	25.17	27.11
GNMT 2 layers	25.72	**28.00**
Our Model	**25.79**	26.97

Table 6-1: Results of GNMT Architectures Vs. the Proposed Model

The models give lower BLEU scores than the final syntactic simplification model and they also produce an output which has redundant words or incomplete sentences. A sample of such output is given in Table 6-2 below.

Gary Goddard is the founder of McDonald 's founder of McDonald .
As winner of his school C , Richie , Richie will now move on to play in the recent Louis Louis Spelling Spelling Bee .
There are many variations , Babb , billowy , slabby , slabby , slabby , slabby , slabby , slabby , slabby , slabby , slabby , slabby , .
And Wal-Mart .
She tells girls that if you are good at math .

Table 6-2: GNMT Faulty Outputs

6.1.2 Different Attention Mechanisms

Attention mechanism affects directly the output and therefore we needed to conduct an experiment to reach the best attention mechanism for our architecture. We tested out two types of attention concat (additive) and general (multiplicative) attentions. The tests were conducted on the models produced prior to applying reinforcement learning using the same parameters described in chapter 5. Table 6-3 below shows the BLEU scores of the two models. It is obvious through scores that the additive attention style has a better BLEU score compared to the multiplicative style.

Attention type	**BLEU**
Additive	24.95
Multiplicative	24.38

Table 6-3: Additive Vs. Multiplicative Attention

By manual inspection of the output, we can also see that the additive attention type produces higher quality output. In the models that use multiplicative attention model had a higher probability of replacing the named entities with words that are not in the text, deleting large

important portions of the sentence or replacing words with others that give another meaning. Examples of such sentences are shown in Table 6-4 below.

Source	Multiplicative	Additive
The NASCAR Acceleration Nation project focuses on the laws of motion , or what it calls the three D 's of speed -- downforce , drafting and drag .	NASCAR **Life** is on the laws of motion .	The NASCAR Acceleration Nation focuses on the laws of motion .
With her mentoring group Polished Pebbles , Kelly Fair gives young black girls exposure to female professionals who can offer advice on how to reach their goals .	(no output was produced by the model)	She gives young black girls to offer advice on how to reach their goals .
The first lady is a graduate of Princeton University and Harvard Law School .	The first lady is a **leader** of Princeton University and Harvard Law School .	The first lady is a graduate of Princeton University and Harvard Law School .
He is the creative director of Rethink Leisure & amp ; Entertainment , which is working on several projects in China and elsewhere in Asia .	He is the creative director of Rethink & amp ; **Associates** .	He is the creative director of Rethink Leisure & amp ; Entertainment .

Table 6-4: Example of Output where Multiplicative Attention Fails

6.1.3 Changing Model Hyperparameters

All the following experiments were done on the model prior reinforcement learning with beam size of 5 to choose the most suitable model to further refine. The final model was chosen based on its BLEU score resulting from the test set. Models are not trained for the same number of epochs, but rather trained till they reach the highest accuracy possible on the development set.

6.1.3.1 Changing in Architecture

1. Hidden units
   We tried changing the model hidden units with 300 and 500 hidden units.

2. Encoder directionality
   We tried the exact same architecture explained in chapter 5, but with a unidirectional encoder instead of the bidirectional encoder suggested.

3. Number of layers
   We tried using 3-layered and a single layered architecture for the encoder and the decoder. All the other parameters and training procedure were kept constant.

The BLEU scores for each of the previous explained model are shown in Table 6-5 below.

Model	**BLEU**
Bidirectional Encoder/2 Layers/300 Hidden Unit	23.65
Bidirectional Encoder/2 Layers/500 Hidden Unit	23.32
Unidirectional Encoder/2 Layers/400 Hidden Units	23.81
Bidirectional Encoder/3 Layers/400 Hidden Units	22.48
Bidirectional Encoder/Single Layer/400 Hidden Units	22.56
Our Suggested Model	**24.95**

Table 6-5: BLEU Scores of Different Model Parameters

Both the 3 layered model and the model with 500 hidden units have a large number of trainable parameters, so the model was not able to train well. Single layered model is not deep enough to capture the suitable sentence features. Results produced by the unidirectional model confirms our assumption that bidirectionality is able to capture better dependencies within the system that are

needed in the simplification task. The 2-layered architecture with 400 hidden units and a bidirectional encoder gave the highest BLEU score and hence was chosen for further refinement.

6.1.3.2 Changing Training Parameters

We tried changing two training parameters which are the batch size and learning rate. All the experiments were done on the model architecture explained in chapter 5. Batch sizes that were included in the experiment were 16, 32, 64 and 128. Learning rates used were 0.001 and 0.0005. BLEU scores resulting from the different combinations of training parameters are shown below in Table 6-6.

		Learning rate	
		0.0005	0.001
Batch size	16	24.91	23.20
	32	23.63	**24.95**
	64	24.72	24.02
	128	24.26	24.29

Table 6-6: BLEU Scores Resulting from Changing Training Parameters

6.2 Lexical Simplifier Experiments

6.2.1 Complex Word Identification Module

The first stage in lexical simplification is to identify complex words. Our approach was first based on a machine learning model, which took in five features: lexicon score, number of characters, number of syllables and the word frequency. However, the accuracy reported from this module wasn't promising and it couldn't identify the complex words correctly when it was integrated in the lexical simplifier. Therefore, we replaced it with thresholding techniques to obtain higher results.

6.2.2 Substitutions Generation

For generating the candidates that could replace a complex word, we first tried retrieving candidates from SimplePPDB [62], SimplePPDB++ [15], Wordnet [22] and the API provided by datamuse [24]. It is important to note that the substitutes generated by the API were the most relevant. However, we had to replace it because it was slow since it connected to the internet to fetch the results. We replaced the API with the thesaurus provided by OpenOffice.org [23] and PPDB [17] to get instantaneous results.

6.2.3 Neural Readability Ranker

We tried to implement the model proposed in [15]. The aim of the model is to output a real number indicating the relative complexity between a pair of input words or phrases (wa, wb), where if the output is negative, then wa is simpler than wb and vice versa.

The general architecture of the model is shown in the following Figure 20, highlighting its three main components:

1. An input feature extraction layer that creates single features from each word, and pairwise features.

2. A Gaussian-based feature vectorization layer that converts numerical features along with other features into a vector representation by using a series of Gaussian radial basis functions. In this step, the value range of each feature is divided into k bins, and then a Gaussian function is placed for each bin with mean and variance that are used to calculate the distance to each feature value. All values are then normalized and concatenated together.

3. A feed forward neural network that takes the concatenated vectors from the previous step as input and applies regression with one task-specific output node, indicating the relative complexity of the two input words.

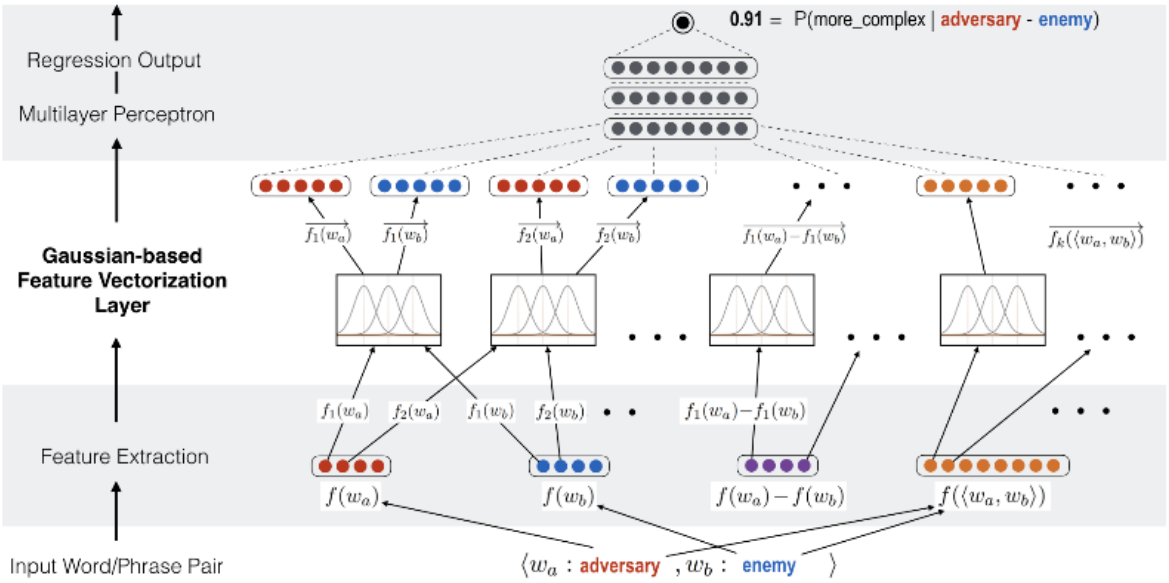


Figure 6-3: The Neural Readability Ranking Model. Retrieved from [15]

The features used in this model are the following: phrase length in terms of words and characters, number of syllables, frequency with respect to Google Ngram corpus [60], the relative frequency in Simple Wikipedia with respect to normal Wikipedia [61] and ngram probabilities from a 5-gram language model trained on the SubIMDB corpus [52], language model probabilities of all the possible n-grams within the context window of 2 to the left and right of w. These individual features, f(wa) and f(wb), are included along with the differences f(wa)−f (wb) and pairwise features including cosine similarity, and the difference between the word2vec embedding [31] of the input words. They use the 300-dimensional embeddings pre-trained on the Google News corpus, which is released as part of the word2vec package.

We implemented the same model. However, we did not get the desired results. For some reason, the model gave the same rank to all candidates. As a workaround, we ignored the pairwise features and inputted individual features for each candidate word to the feedforward neural network instead and then scaled features of candidates for the same complex word. The neural network's output is then a rank for each candidate.

6.3 Sentence Simplification for Arabic language

We had a huge desire to implement our project for Arabic language so that we can promote reading in Arabic. The biggest obstacle we encountered was the unavailability of a suitable dataset, since the state of the art technique in sentence simplification is using a Seq2Seq model which requires monolingual parallel corpora. To overcome that obstacle we tried two different approaches: automatically generating the desired dataset and using basic technique like tree transduction.

6.3.1 Automatically Generating Data Set

Since Wikipedia does not have a simplified version in Arabic, it is not possible to apply the same techniques used for English. To overcome that we tried different work arounds with the help of translation tools:

1. Translating Wikipedia and Simple English Wikipedia articles to Arabic
2. Translating parallel English corpora to Arabic
3. Translating Arabic Wikipedia articles to English and back to Arabic and using the output as the simplified version

This approach did not yield good results due to inaccuracies in the output of the translation tools as shown in Table 6.7.

	English	**Arabic**
Complex	Genetic engineering has expanded the genes available to breeders to utilize in creating desired germlines for new crops.	وسعت الهندسة الوراثية الجينات المتاحة للمربين للاستفادة منها في إنشاء سلالات الجرثومية المطلوبة للمحاصيل الجديدة.
Simple	New plants were created with genetic engineering.	تم إنشاء نباتات جديدة مع الهندسة الوراثية.
Complex	The print collections are further supported by extensive microform holdings.	يتم دعم مجموعات الطباعة بشكل أكبر بواسطة مقتنيات microform الواسعة.
Simple	The print collections include large collections of microform.	تتضمن مجموعات الطباعة مجموعات كبيرة من microform.

Table 6-7:Example of Translating Parallel English Corpora to Arabic

6.3.2 Tree Transduction

Tree transduction is a method that compares two parse trees to extract rules that transform one sentence to another. Using complex and sentence sentences we can extract how to change the syntax of a complex sentence to be simpler. Figure 6.4 shows the pipeline used in training and inference of the tree transduction model.

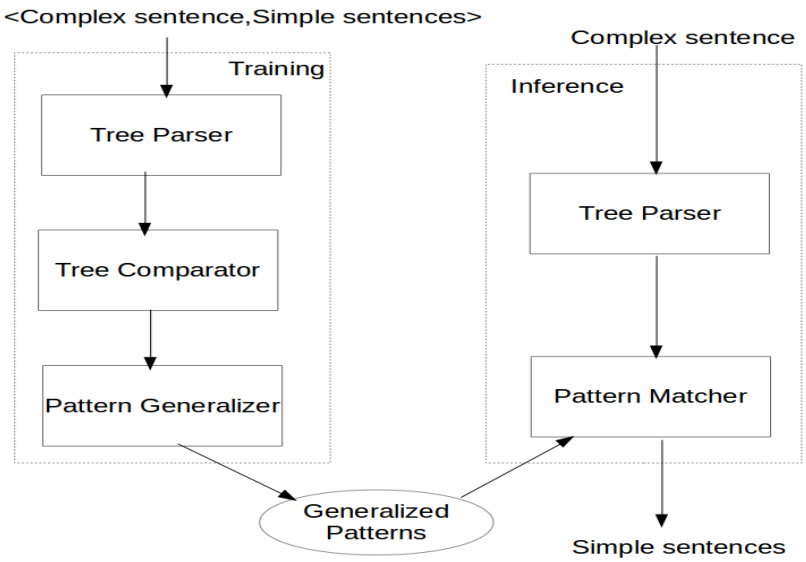


Figure 6-4: Tree Transduction Model Pipeline

First step of both training and inference is parsing sentence into a parse tree to be able to perform the necessary transformations. We used the Stanford Parser for Arabic language for this task. To extract the syntax transformation rules the parse tree of the complex sentence and the simple sentences pass through two modules, the tree comparator and the pattern generalizer. The tree comparator compares the complex sentence tree with the simple sentences trees and see where they match. The match is done bottom up where we first compare the leaves and if they match we compare the parents till we reach the largest matching subtrees. The pattern generalizer takes the string format of the tree and replace the matching subtrees in the tree with a generalized format of what the complex sentence and simple sentence. The matching subtree are replaced with .*? In the complex sentence parse tree to enable regex matching of the same format but with different words. As for the simple sentences we replace the match subtree with the index of the this subtree in the complex sentence to allow copying words from the complex sentence to the

simple sentence during inference. An example of the rules the system outputs is given Table 6.8 Below where simple sentence rules given in an array form indicate that sentence splitting is involved.

Complex Sentence Rules	Simple Sentence Rules
(VP (VBP.*?) (NP.*?) (NP (NN .*?) (JJ .*?)))	(VP [(0,)] [(1,)])
(ROOT= (NP= (NP.*?)= (SBAR= (WHNP (WP .*?))= (S (VP.*?)))))	['(ROOT (S [(0, 0)] [(0, 1, 1, 0)]))', '(ROOT (S [(0, 0)] [(0, 1, 1, 0, 2)]))']

Table 6-8: Example of Rules Given by the System

During inference time the sentence parse tree is matched against the extracted rules and matches found, whether the match is part of the original sentence or the whole sentence, are replaced with its simpler equivalents.

This technique needs the training set to be carefully written by experts to include the most important complex syntax and its simple equivalent syntax. Also this technique does not account for word replacement in the simple sentence focusing mainly on word recording and sentence splitting. Given that the Arabic language is a rich language that contains many syntactic formats that are all equivalent, the dataset has to be very large to account for all the possible variations. Obtaining such a dataset was not possible given the time constraints and if such a large dataset could be crafted it will be more practical to use the state-of-the-art Seq2Seq technique to approach the problem.

Chapter 7
Languages, Tools and Libraries

7.1 Languages

Language	Description	Usage
Python	Python is an interpreted, high-level, object oriented programming language. Python has built-in high level data structures, a syntax that emphasizes readability and supports writing scripts.	Used in data preprocessing , implementing machine learning models and evaluation metrics
Shell Script	Shell Script is a command line interpreted language designed to be run by the Unix/Linux shell. Typical operations performed by shell scripts include file manipulation, program execution, and printing text	Used to run testing scripts on the output of different models and write commands that run the system modules

7.2 Tools

Tool	Description	Usage
PyCharm	PyCharm is an integrated development environment (IDE) used in computer programming, specifically for the Python language. It contains a Python terminal, a debugger.	Summarizer and the syntactic simplifier modules are both implemented using PyCharm.
Google Colaboratory	Google Colab, Google's free cloud service for AI developers. Google Colab is a free cloud service that supports free GPU. It can support developing deep learning applications using popular libraries such as Keras, TensorFlow and PyTorch. It runs Python Notebooks.	Used to train the models we implemented, as they need a GPU to run the training in a short time and a large processing power to support holding the training and dev sets

Github	Github is a web-based hosting service that offers version control using Git	Used for version control of the modules. Syntactic Simplification training files were uploaded on github and cloned from github to Google Colaboratory for training in order to have a more readable code.
MIT Language Modeling Toolkit	The MIT Language Modeling (MITLM) toolkit is a set of tools designed for the efficient estimation of statistical n-gram language models. [38]	Used for building both the three-gram and five-gram language models that were used in the lexical simplifier.
Django	A Python-based free and open-source web framework.	It was used to build the interface for the summarizer, lexical and syntactic simplifiers.

7.3 Libraries

Library	Description	Usage
NLTK	NLTK is Python library used with human language data. It offers many text processing techniques such as tokenization, stemming, tagging and parsing	It is used in text preprocessing for the summarizer using the tokenizing and stemming functionalities.
PyTorch	PyTorch is an open source deep learning framework built to be flexible and modular for research, with the stability and support needed for production deployment. It enables fast, flexible experimentation through a tape-based autograd system designed for immediate and python-like	It is used in building the syntactic simplification model and all the syntactic simplification module experiments.

	execution.[21]	
NumPy	NumPy is the fundamental package for scientific computing with Python. Among its capabilities are creating and manipulating(sum, mean, matrix multiplication, etc.) multidimensional arrays and generating random numbers	NumPy is used in all project phases that contain matrices and matrix operations.
Stanford Parser	The Stanford parser is a library that works out the grammatical structure of sentences, for instance, which groups of words go together (as "phrases") and which words are the subject or object of a verb. [42]	It was used in the Complexity Assessment module and Lexical Simplification module to get pos tags and parse trees

Chapter 8
Conclusion and Future Work

8.1 Conclusion

Our project provides a language tool for simplifying and summarizing English text that could be used in the learning process. The simplifier consists of two main modules which are lexical and sentence simplifiers. First, the lexical simplification module is used to replace complex words with less complex ones. Our lexical simplifier outperformed all other models in both substitution generation and ranking tasks according to the benchmark proposed in [16]. Second, the sentence simplification module that aims to reduce the linguistic complexity of the sentence. It has a better BLEU score than the state-of-the-art model, DRESS, and has higher SARI score when used without reinforcement. As for the summarizer, which uses TextRank algorithm to provide extractive summarization, the average ROUGE scores have exceeded that of the Gensim library for the same input. We also provide a complexity assessment module to help with the evaluation by extracting features from a given sentence and rating it. It has also achieved highest accuracy among all other models used for complexity assessment. As shown, each of the four models proposed has proved outperforming results over all other models provided.

8.2 Future Work

8.2.1 Enhancing Google Machine Translation Model

The 2-layer Google machine translation model showed promising results, but suffered from repetition and incomplete sentence problems. Such problems could be enhanced using reinforcement learning such as that proposed in chapter 5 or that proposed by Zhang and Lapata [9]. Reinforcement learning technique and parameters best to be used can be discovered by further experimentation.

8.2.2 Using anonymized training set

Dataset includes a large number of named entities that increase the number of vocabulary. To decrease the vocabulary size and therefore the trainable parameters, Zhang and Lapata [9] suggested to use an anonymized data set, where every named entity is replaced with an anonymized version. This is done using a Named Entity Recognition tagger that identifies named entities and categorizes them into one of the following categories {ORGANIZATION, PERSON, LOCATION, MISC}. The text is anonymized into NE@N token, where NE belongs to the categories mentioned and N indicates NE@N is the N-th distinct NE typed entity. For example, “John and Bob are . . .” becomes “PERSON@1 and PERSON@2 are . . .”. This technique will reduce the target vocabulary from 30481 to 17500 and the source vocabulary from 41380 to 24545.Decreasing learning parameters may allow the model to train better.

8.2.3 Abstractive summarization

Currently used summarization technique is the extractive technique which chooses the most relevant sentences and presents them as the new summary. Alternatively, abstractive summarization tries to mimic how humans summarize where it captures the document meaning and summaries it in a different style. Abstractive summarization is currently limited to short news articles and with the current accuracies of text simplification and abstractive summarization using them in a pipeline might not be the best solution currently. Another suggestion is to fuse both summarization and simplification task and use one end-to-end model to do both tasks, but this will require a larger document to document corpus than that provided by Newsela.

8.2.4 Word Sense Disambiguation (WSD)

Words can have multiple meanings when used in different contexts and while it can be a relatively easier task for humans to disambiguate words, it can be very difficult for machines to catch the right sense in a context. This makes it harder to select the correct candidates in lexical simplification. Choosing an irrelevant substitute and replacing the target word with it can have disastrous results since it can completely change the meaning of a sentence. Some lexical simplification systems have applied WSD by using latent words language models. While others use WordNet synsets and hypernyms along with the Lesk algorithm to get the target word sense. We have tried this algorithm ourselves and found that it was too simple to accurately get the correct sense. WSD is also carried out by using context vectors and vector similarity measures. Recent approaches have started using sense-annotated corpora to train from, as the SemCor Corpus [58]. However, this is still an open problem that needs further research to get satisfying results.

Chapter 9
Bibliography

[1] Shivangi Sareen.(Aug 7, 2018).TFIDF.[Online].Available: https://medium.com/@shivangisareen/tfidf-7b29017dcdd .[Last visit 8 May 2019]

[2] Michael Nguyen.(Sep 24, 2018). Illustrated Guide to LSTM's and GRU's: A step by step explanation.[Online].Available: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21 .[Last visit 8 May 2019]

[3] Stanford CS224N: NLP with Deep Learning | Winter 2019 | Lecture 8 – Translation, Seq2Seq, Attention.(2 May 2019).[Video file]. Available: https://www.youtube.com/watch?v=XXtpJxZBa2c .[Last visit 8 May 2019]

[4] Rada Mihalcea and Paul Tarau.(2004). TextRank: Bringing Order into Texts. Retrieved from https://www.aclweb.org/anthology/W04-3252

[5] Federico Barrios, Federico L´opez, Luis Argerich, Rosita Wachenchauzer.(2016).Variations of the Similarity Function of TextRank for Automated Summarization. Retrieved from https://arxiv.org/pdf/1602.03606.pdf

[6] Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In Proceedings of the 2015 EMNLP, pages 1412–1421, Lisbon, Portugal.

[7] Sander Wubben,Antal van den Bosch and Emiel Krahmer.(2012) .Sentence Simplification by Monolingual Machine Translation .Retrieved from: https://www.aclweb.org/anthology/P12-1107

[8] Sashi Narayan and Claire Gardent.(2014).Hybrid Simplification using Deep Semantics and Machine Translation. Retrieved from: https://www.aclweb.org/anthology/P14-1041

[9] Xingxing Zhang and Mirella Lapata.(2017).Sentence Simplification with Deep Reinforcement Learning. Retrieved from: https://aclweb.org/anthology/D17-1062

[10] Tu Vu, Baotian Hu,Tsendsuren Munkhdalai and Hong Yu.(2018).Sentence Simplification with Memory-Augmented Neural Networks. Retrieved from: https://aclweb.org/anthology/N18-2013

[11] Wei Xu,Chris Callison-Burch and Courtney Napoles.(2015).Retrieved from: Problems in Current Text Simplification Research: New Data Can Help.https://cocoxu.github.io/publications/tacl2015-text-simplification-opinion.pdf

[12] Sanja Stajner and Horacio Saggion.(2018).Data-Driven Text Simplification. Retrieved from: https://www.aclweb.org/anthology/C18-3005

[13] Zhemin Zhu,Delphine Bernhard and Iryna Gurevych.(2010).A Monolingual Tree-based Translation Model for Sentence Simplification.Retrieved from: https://aclweb.org/anthology/C10-1152

[14]Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi,Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, Jeffrey Dean.(2016).Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation.Retrieved from:https://arxiv.org/pdf/1609.08144.pdf

[15] Mounica Maddela and Wei Xu (2018) , A Word-Complexity Lexicon and A Neural Readability Ranking Model for Lexical Simplification

[16] Gustavo H. Paetzold and Lucia Specia (2016) , Benchmarking Lexical Simplification Systems. Retrieved from: https://pdfs.semanticscholar.org/c811/2c63ab80cbdb23b5ad8a1d3a6d22a9112020.pdf

[17] Pavlick, Ellie & Rastogi, Pushpendre & Ganitkevitch, Juri & Van Durme, Benjamin & Callison-Burch, Chris. (2015). PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. 425-430. 10.3115/v1/P15-2070.

[18] LuminosoInsight. (2019, April 16). LuminosoInsight/wordfreq. Retrieved from https://github.com/LuminosoInsight/wordfreq

[19] Instructional Content Platform. Retrieved from https://newsela.com/

[20] 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.

[21] PyTorch Official Facebook Artificial Intelligence webpage. https://ai.facebook.com/tools/pytorch/ [Accessed in 20 May 2019]

[22] George A. Miller (1995). WordNet: A Lexical Database for English. Communications of the ACM Vol. 38, No. 11: 39-41.

[23] Kevin B. Hendricks (2003). Retrieved from https://wiki.openoffice.org/wiki/Dictionaries

[24] Retrieved from http://www.datamuse.com/api/

[25] Gustavo Henrique Paetzold and Lucia Specia (2016). Collecting and Exploring Everyday Language for Predicting Psycholinguistic Properties of Words. Proceedings of the 26th COLING. 2016

[26] Papineni, Kishore & Roukos, Salim & Ward, Todd & Zhu, Wei-jing. (2002). BLEU: a Method for Automatic Evaluation of Machine Translation. 10.3115/1073083.1073135

[27] Lin, Chin-Yew. (2004). ROUGE: A Package for Automatic Evaluation of summaries. Proceedings of the ACL Workshop: Text Summarization Braches Out 2004. 10.

[28] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation.

[29] Sulem, Elior & Abend, Omri & Rappoport, Ari. (2018). Simple and Effective Text Simplification Using Semantic and Neural Methods.

[30] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. Journal of the American Society for Information Science, 41.

[31] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient Estimation of Word Representations in Vector Space. In ICLR Workshop Papers.

[32] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In NIPS, pages 3111–3119.

[33] Tomas Mikolov, Wen tau Yih, and Geoffrey Zweig. 2013c. Linguistic regularities in continuous space word representations. In HLTNAACL.

[34] 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.

[35] Roser, M., & Ortiz-Ospina, E. (2016, August 13). Literacy. Retrieved from https://ourworldindata.org/literacy

[36] Marrus, N., & Hall, L. (2017, July). Intellectual Disability and Language Disorder. Retrieved from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5801738/

[37] Bo-June (Paul) Hsu and James Glass. Iterative Language Model Estimation: Efficient Data Structure & Algorithms. In Proc. Interspeech, 2008. Toolkit can be found at https://github.com/mitlm/mitlm

[38] V, Sowmya & Meurers, Detmar. (2012). On Improving the Accuracy of Readability Classification using Insights from Second Language Acquisition. 163–-173.

[39] Meri Coleman and T.L. Liau. 1975. A computer readability formula designed for machine scoring. Journal of Applied Psychology, 60:283–284.

[40] J. P. Kincaid, R. P. Jr. Fishburne, R. L. Rogers, and B. S Chissom. 1975. Derivation of new readability formulas (Automated Readability Index, Fog Count and Flesch Reading Ease formula) for Navy enlisted personnel. Research Branch Report 8-75, Naval Technical Training Command, Millington, TN.

[41] Averil Coxhead. 2000. A new academic word list. Teachers of English to Speakers of Other Languages, 34(2):213–238.

[42]Peng Qi, Timothy Dozat, Yuhao Zhang and Christopher D. Manning. 2018. Universal Dependency Parsing from Scratch In Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, pp. 160-170. [pdf] [bib]

[43] S. Devlin and J. Tait, "The use of a psycholinguistic database in the simplification of text for aphasic readers." Linguistic Databases, pp. 161–173, 1998.

[44] C. Fellbaum, WordNet: An Electronic Lexical Database. Bradford Books, 1998.

[45] H. Kučera and W. N. Francis, Computational analysis of present-day American English. Providence, RI: Brown University Press, 1967.

[46]Carroll, J., Minnen, G., Pearce, D., Canning, Y., Devlin, S., and Tait, J. (1999). Simplifying text for language impaired readers. In Proceedings of the 9th EACL, pages 269–270.

[47] Ong, E., Damay, J., Lojico, G., Lu, K., and Tarantan, D. (2007). Simplifying text in medical literature. volume 4, pages 37–47.

[48] Leroy, G., Endicott, J. E., Kauchak, D., Mouradi, O., and Just, M. (2013). User evaluation of the effects of a text simplification algorithm using term familiarity on perception, understanding, learning, and information retention. Journal of Medical Internet Research, 15.

[49] Paetzold, G. H. (2013). Um Sistema de Simplificação Automática de Textos escritos em Inglês por meio de Transduçao de Árvores. Western Parana State University.

[50] Paetzold, G. H. and Specia, L. (2013). Text simplification as tree transduction. In Proceedings of the 9th STIL.

[51] Paetzold, G. H. and Specia, L. (2015). Lexenstein: A framework for lexical simplification. In Proceedings of The 53rd ACL.

[52] Paetzold, G. H. and Specia, L. (2016). Unsupervised lexical simplification for non-native speakers. In Proceedings of The 30th AAAI.

[53] Paetzold, G. H. (2015). Reliable lexical simplification for non-native speakers. In Proceedings of the 2015 NAACL Student Research Workshop.

[54] Horn, C., Manduca, C., and Kauchak, D. (2014). Learning a Lexical Simplifier Using Wikipedia. In Proceedings of the 52nd ACL, pages 458–463.

[55] Glavaš, G. and Štajner, S. (2015). Simplifying lexical simplification: Do we need simplified corpora? In Proceedings of the 53rd ACL, page 63.

[56] 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: Volume 2, Short Papers.

[57] Bott, S. and Saggion, H. (2011). An unsupervised alignment algorithm for text simplification corpus construction. pages 20–26.

[58] https://www.sketchengine.eu/semcor-annotated-corpus/

[59] Biran, O., Brody, S., and Elhadad, N. (2011). Putting it simply: a context-aware approach to lexical simplification. In Proceedings of the 49th ACL, pages 496–501.

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

[61] 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).

[62] Kajiwara, T., Matsumoto, H., and Yamamoto, K. (2013). Selecting Proper Lexical Paraphrase for Children. Proceedings of the 25th Rocling, pages 59–73.

Appendix A
User Manual

➢ General Information

- The website has only one web page and provides the user with four main options:
    1. Summarization
    2. Lexical Simplification, that is simplifying sentences on a word level
    3. Sentence Simplification; simplifying the structure of a sentence
    4. Simplifying both the structure and words of a sentence.
- Application name: SumSim
- Operational status: under development. It is currently a demo that runs on a local host.
- The website can be used by second-language learners, adults with reduced literacy, children or any other people who would like to summarize or simplify a piece of text for any purpose.

➢ Getting Started

When you open the website the following page will appear.

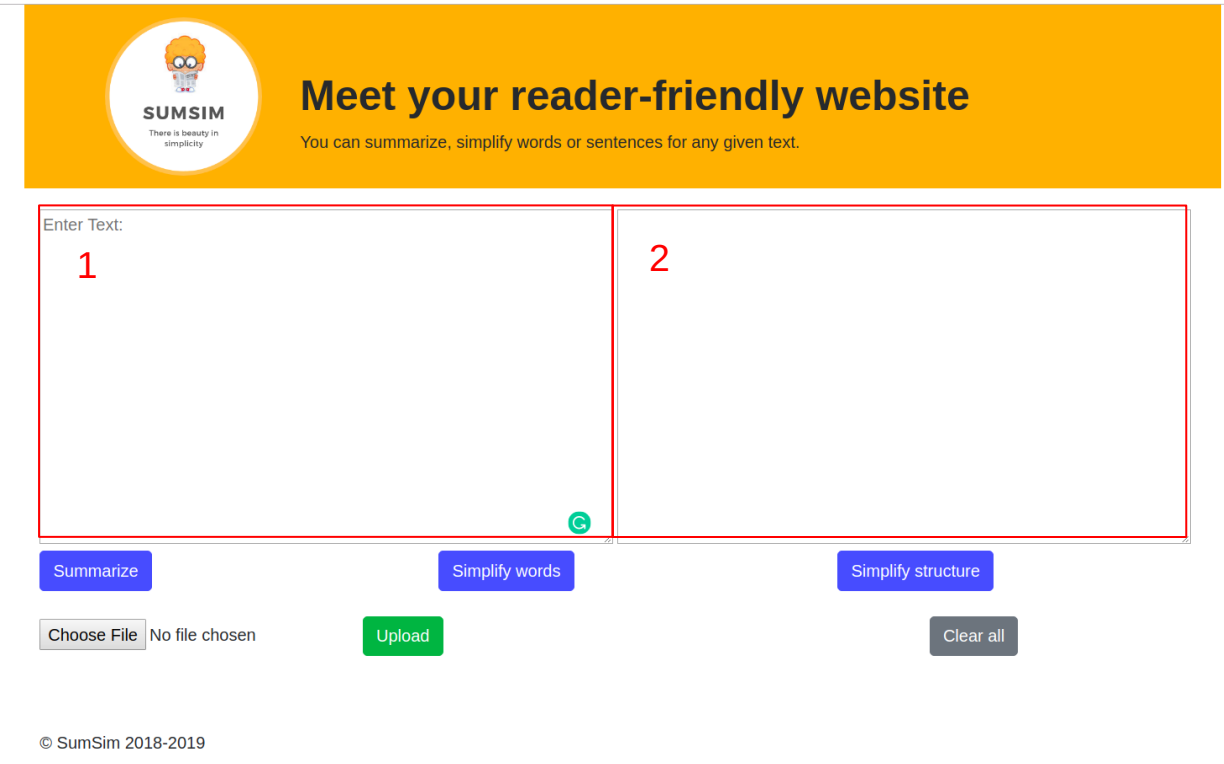


Figure A-1: Interface Preview

1. You can enter your input text either by copying and pasting it in the area labeled 1 or by uploading it from a file.
2. To upload a file, click on the “Choose File” button.
3. After clicking, a window will appear. You can then navigate to the file that you want to upload and click open.

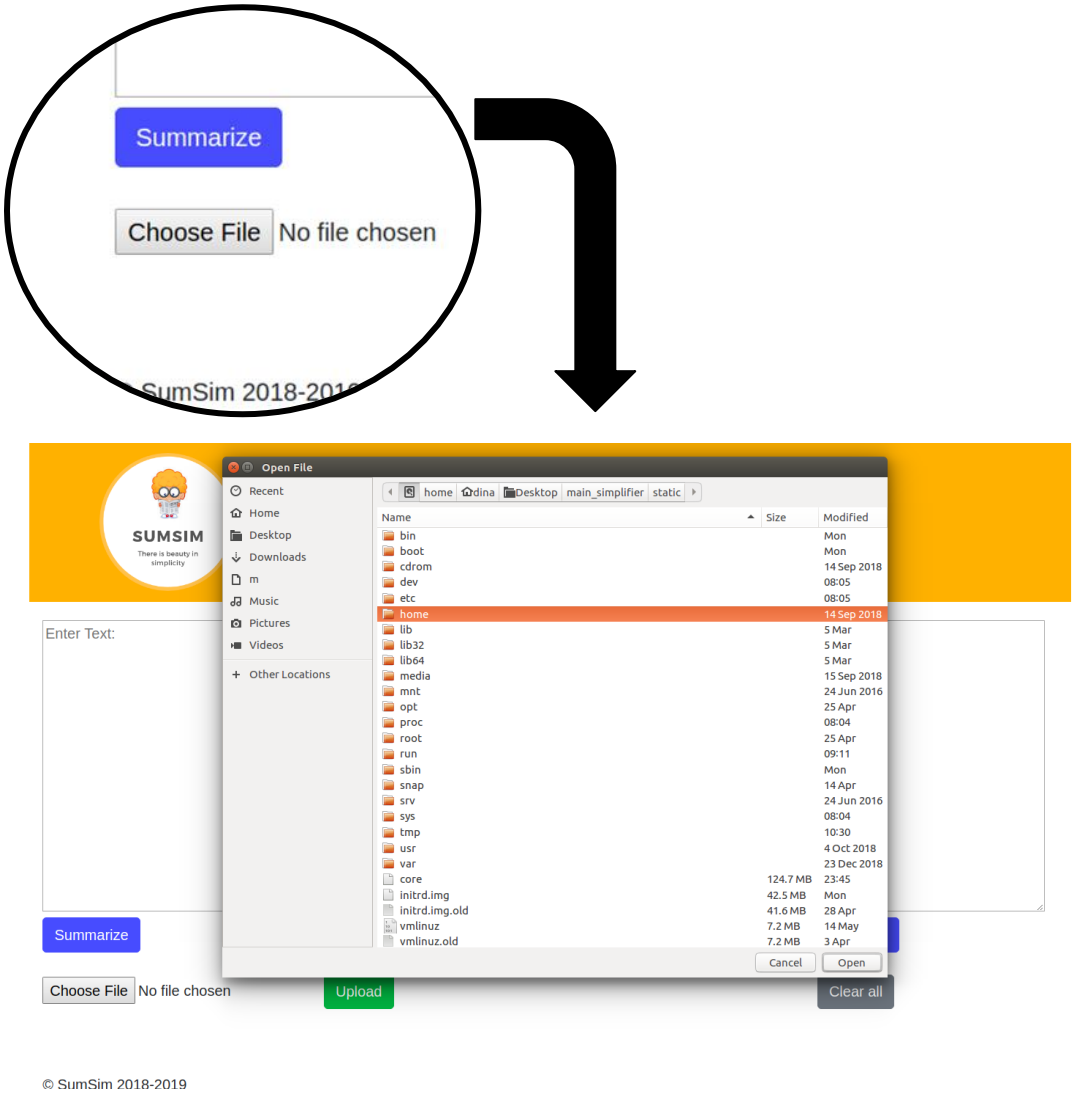


Figure A-2: Choosing File to Upload

4. Then, by clicking the upload button, your file content will appear in the input text area.

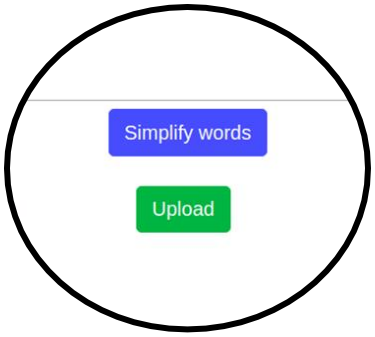


Figure A-3: Upload Feature Button

5. Once you have filled the input text area, you can now use any of the provided options, namely summarization, simplifying words, and simplifying structure of sentences.

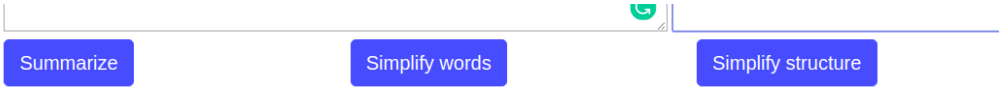


Figure A-4: Interface Buttons

6. Use the “Summarize” button to get a summary of your text.
7. Use the “Simplify words” button if you have an input text with complex words.
8. Use the “Simplify structure” button if you want to simplify the structure of sentences.
9. After clicking any of the buttons in steps 6 to 8, the output text will appear in the textbox labeled 2 in the first screenshot.
10. To reset all and clear text, click on the “Clear all” button.