Enhancing Text Readability using Deep Learning Techniques

by

Wejdan Alkaldi

Thesis submitted to the

University of Ottawa

In partial fulfillment of the requirements

For the Ph.D. degree in

Computer Science

School of Electrical Engineering and Computer Science

Faculty of Engineering

University of Ottawa

Abstract

In the information era, reading becomes more important to keep up with the growing amount of knowledge. The ability to read a document varies from person to person depending on their skills and knowledge. It also depends on the readability level of the text, whether it matches the reader's level or not. In this thesis, we propose a system that uses state-of-the-art technology in machine learning and deep learning to classify and simplify a text taking into consideration the reader's level of reading. The system classifies any text to its equivalent readability level. If the text readability level is higher than the reader's level, i.e. too difficult to read, the system performs text simplification to meet the desired readability level. The classification and simplification models are trained on data annotated with readability levels from in the Newsela corpus[1]. The trained simplification model performs at sentence level, to simplify a given text to match a specific readability level. Moreover, the trained classification model is used to classify more unlabelled sentences using Wikipedia Corpus and Mechanical Turk Corpus. in order to enrich the text simplification dataset. The augmented dataset is then used to improve the quality of the simplified sentences. The system generates simplified versions of a text based on the desired readability levels. This can help people with low literacy to read and understand any documents they need. It can also be beneficial to educators who assist readers with different reading levels.

[1] https://newsela.com/data/

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Overview

The ultimate goal of writing a text is to communicate. Therefore, any written text must be readable and understandable to its targeted audience. However, there are many readers with poor literacy not able to fully comprehend the texts they read. These individuals might have a low level of reading and understanding a given text. In this case, the text must be explained in order to allow them to grasp the ideas behind it. Children who are developing their literacy skills would have to go through the same process.

The organization of the text and the vocabularies used affect the text readability level. Determining the readability level for a given text is not an easy task. Many metrics were developed to measure the readability level of a written text. One of the most popular metric is *Flesch-Kincaid* (Kincaid et al., 1975), which uses the number of sentences, words, and syllables in an equation that determines how easy or difficult a given text is to comprehend.

In addition to the readability metrics, there are many features that determine the readability level of a text. In (Aluisio et al., 2010), there are 59 features listed that could be used for readability assessment in machine learning methods, in order to classify a given text based on its readability level. The nature of these features varies in the readability metrics, from words and textual information, to syntactic information, and to logical operations.

Manipulating some of these features could increase the readability of the text. For instance, splitting long sentences into shorter ones, converting passive voice sentences into active voice, or substituting unusual vocabularies with common terms, would definitely have an impact on the readability level of the original text. The readability level of the text would increase to a certain level that would allow poor literacy readers or children to read and understand the written text.

It would be useful to automatically determine the current readability level of a text and then simplify it, if needed, to the desired readability level. This would serve users with poor literacy to be able to access and understand all the information they need. Also, educators could help their students improve their reading skills by providing class materials that match the desired reading level.

1.2 Motivation

Reading is one of the important keys to maintain a healthy mind and brain. It is an essential skill that needs to be cultivated from a very young age and practiced through

lifetime, as a platform for lifelong learning, enjoyment, and improvement.

However, readers sometimes struggle to read a text that does not match their reading level. This would discourage the reader to continue reading. Studies show that struggling readers are not motivated to read, and their opportunities to learn decrease significantly (Baker et al., 2000). This can lead to strong negative feelings about reading, and this means poor readers will remain poor readers.

Studies proved that children who read for enjoyment every day will have a better educational performance than those who do not. Also, they develop a broader vocabulary, increased general knowledge, and build a better understanding of other cultures (Clark and Rumbold, 2006). So, it is very important to find the suitable text to encourage reading more.

1.2.1 Difficult Sentences

Many aspects could make a sentence difficult to read and understand. For example, using uncommon vocabularies or phrases would require time and effort from the reader in order to comprehend the meaning behind the text. Therefore, substituting these words and phrases with common ones would make the text more readable.

Also, long sentences could express several ideas, making it difficult to follow the main topic. Therefore, it is better to split long sentences into several short sentences. This could express the ideas better and make the text more clear and simple for readers to understand. In addition, complex syntax in the sentences can make it hard to follow and read the whole text fluently. In this case, we could reorder the structure of the sentence to make it more

direct and clear. For instance, sentences with passive voice could be simplified into active voice.

1.2.2 Readability Levels

Every written text has a certain readability level. This level can help when deciding whether a text is suitable for a certain reader or not. The level of the text must match its intended audience level, to understand and comprehend the message behind the text. For example, writing a report about "*Energy*" for grade one students would not be the same as writing a report for scientists. Both would handle the same topic and discuss the same issues, but one might have more paragraphs than the other, use more scientific terms, contain longer complicated sentences, and require a scientific background to be able to follow and understand the text. So, the two reports can have the same topic, yet different readability levels.

If the text is too difficult for the reader's level, the reader would not fully understand the text and that will discourage him/her to read more. On the other side, reading a text that has a different readability level than the reader's reading level would not be useful to the reader if the text is too simple.

Therefore, finding a reading material about a certain topic must match the readability level of the audience. Otherwise, the readers would not benefit from reading and could build a negative feelings towards reading in the future.

1.3 Tasks

1.3.1 Finding the Readability Level of a Text

In many situations, finding the readability level of a text is essential to determine whether the text is suitable for the reader or not. Many formulas were proposed in order to measure the readability of a text, like *Flesch-Kincaid Grade Level* (Kincaid et al., 1975) and *Gunning Fog Index* (Gunning, 1969). There are also some models that categorize a text using machine learning methods (Aluisio et al., 2010; Bessou and Chenni, 2021; Marvin Imperial and Ong, 2021). However, deep learning proved to be a very powerful machine learning method solving many NLP problems as mentioned in Giovanelli et al. (2017). Yet, we could not find a model that classifies a text into its rightful readability level using deep neural networks.

Therefore, we propose a classifier that uses deep learning approaches to classify documents and sentences to their right readability level, ranging from 0 to 4, using semantic and syntactic features. Then we used this model to classify more data to different readability levels and create a new augmented dataset that we will use for training a text simplification system.

1.3.2 Simplifying a Text to a Given Readability Level

We noticed that text simplification techniques available now do not use the readability level as a required feature for the output text. They basically simplify the given text to whatever readability level it can reach.

For instance, consider readability levels from 1 to 4, where 1 represents very difficult text and 4 very easy text to read. If a reader with reading level 3 reads a level 1 document, the text must be simplified to the reader's level to be comprehended. So, using the available simplification techniques, the original text could be simplified to a readability level that cannot be controlled. If the level is 2, this will result in a text that is still difficult for the reader to read and understand. Even through this simplified text is easier than the original text, it is not enough for the reader to grasp and comprehend the ideas behind the text. So, the original text must be simplified to represent the readability level of at least 3 or 4. Unfortunately, this scenario cannot be executed with the available techniques since the readability level of a text does not play a roll in the present simplification models.

To fill this gap, we created and trained a state-of-the-art simplification model using aligned sentences from the Newsela dataset (Newsela Inc., 2019) The model takes a complex text with low readability level, and produces the simplified version of the text taking into consideration the required readability level.

1.4 Hypothesis

We have two hypothesis we are trying to prove in this thesis:

- We hypothesise that it is possible to build a classifier that classifies a text into a readability level scale using deep learning techniques.
- We also hypothesise that it is possible to build a simplification system that generates simpler text at a desired readability level.

1.5 Contributions

We built a system that uses deep neural network mechanisms as a central component to determine the readability level of a given text. Then, it preforms a text simplification to the required readability level. This system improves the readability of a given text to the level desired. It also classifies sentences and longer texts to provide their readability levels. Our contributions can be described as follows:

- Classify a given sentence or a whole document to its readability level. This has not been performed before using deep learning techniques.
- Use syntactic, semantic, and readability metrics features to classify text into five classes labeled from *0* to *4* where 0 represents low readability, *difficult text*, and 4 represents high readability, *simple text*. In previous work, only some of these features were uses, not all together.
- Determine the user's readability level by classifying a given document by the user employing our pretrained document classifier. This will help decide the target readability level for text simplification.
- Generate a novel dataset of classified sentences based on readability level that could be employed by researchers who need a larger corpus for text simplification with sentences categorized into five readability levels.
- Simplify a text using state-of-the-art deep learning techniques, then improve the simplification performance by using text readability level as part of the simplification

process and by using the new (augmented dataset) that was generated. Incorporating the readability level classifier in the simplification algorithm through reinforcement learning reward function is a novel work and has not been done before in previous research.

- Allowing users to specify the readability level wanted for the output text. This feature was not available in the previous text simplification systems.

1.6 Thesis Organisation

In the rest of this thesis, we will discuss the problem of enhancing text readability in more details, with focus on text classification and text simplification. Then we will describe and test our proposed system in enhancing text readability level. The reminder of this document is organized as follows:

- **Chapter 2** provides background on enhancing text readability, including text classification and simplification, followed by an overview of the state-of-the-art deep learning techniques used in Natural Language Generation.

- **Chapter 3** highlights the available works related to text classification based on readability level of the text, and text simplification projects used in different languages. It also mentions simplification systems that use deep learning techniques as their main component.

- **Chapter 4** reviews the datasets that could be used for text classification and simplification. It includes datasets with aligned sentences and others with aligned articles.

Only one of them is labeled with readability levels.

- **Chapter 5** explains the methodology we used for text classification and simplification. It also discusses the available readability metrics and list the scores commonly used for simplification evaluation.

- **Chapter 6** presents our text classification experiments by readability level using deep neural network. We explain the work and outline the main features we used for the classification and analyzes the results. The classification is performed based on the readability of the text at document level and sentence level. In the last part of this chapter, the classifier is applied to produce a large augmented dataset for simplification (pairs of complex sentences and simplified sentence at readability levels 1 to 4).

- **Chapter 7** explains our text simplification system with and without a reinforcement learning loop. We present and compare several sentence simplification results using aligned sentences from the Newsela dataset and using the augmented dataset we produced automatically with our classification model. At the end of this chapter, we analyse the results.

- **Chapter 8** concludes our work and suggests future work that could help improve the readability system further.

Chapter 2

Background

In this chapter, we provide a background knowledge on text readability and ways to enhance it, including text classification and simplification, followed by an overview of the state-of-the-art deep learning techniques used in Natural Language Generation (NLG) since they are used in our system.

2.1 Text Classification

According to IBM (Schneider, 2016), around 80% of all data available is unstructured, with text being one of the most common types of unstructured data such as emails, legal documents, web pages, chat conversations, and social media messages. It is not that easy to analyze, understand, organize, and sort a text. It takes time and effort to extract information using text data, and that is due to the unstructured nature of text. This is where automatic text classification can be useful, with its fast performance and efficient

cost.

Text classification is one of the fundamental problems in Natural Language Processing (NLP). It is the task of determining a suitable label for a sentence or a document to make it more structured and useful. Defining the classes and choosing the labels depends on the dataset used. There could be two or more classes for a dataset.

A text classification system can be applied on different levels of text: document level, paragraph level, sentence level, and even sub-sentence level (Kowsari et al., 2019). Each level is a subset of the other. Document level has categories relevant to full documents, while paragraph level deals with a single paragraph, or a portion of a document. Sentence level handles categories of a single sentence, or a portion of a paragraph. Sub-sentence level obtains relevant categories of sub-expressions within a sentence, or a portion of a sentence. Therefore, text classification systems could run through several phases to assign a label to a given text.

2.1.1 Text Classification Pipeline

Most text classification systems share a basic pipeline with five phases. The pipeline includes: data cleaning, extracting features, selecting a classifier model, validating and testing, and an optional phase called reducing dimensions. The pipeline phases are shown in Figure 2.1.

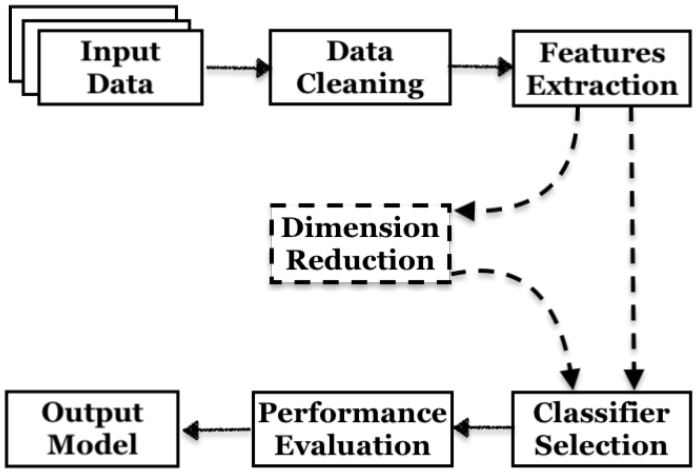

Figure 2.1: Text Classification Basic Pipeline

Data cleaning

This is the first phase where the outdated, in-completed, duplicated, or incorrect data would be removed. The remaining data should have high quality and will be used for processing. This phase improves the quality of the input data which later could increases overall productivity and accuracy.

Feature extraction

Because the input data is unstructured text, it is very important to convert the data into structured features. These features will be used in mathematical modeling as part of a classifier. There are many feature extraction methods that can be applied. Some of the common techniques of feature extractions include: Term Frequency (TF) (Salton and Buckley, 1988), Term Frequency-Inverse Document Frequency (TF-IDF), Word2Vec (Goldberg and Levy, 2014), Global Vectors for Word Representation (GloVe) (Pennington

et al., 2014).

Dimension reductions

This phase is an optional stage in case data processing was not going well due to time and memory cost. In this case, using inexpensive algorithms will not perform as well as expected. Therefore, to avoid the downside of the performance, we use dimensional reduction. Using fewer dimensions will reduce the time and memory complexity and improve the overall performance.

Classifier selection

The most significant phase in a text classification pipeline is choosing the best classification algorithm. There are many algorithms found for classification like logistic regression (LR) (Harrell, 2001), Random Forest (RF) (Tin Kam Ho, 1995), Support Vector Machine (SVM) (Han and Karypis, 2000), and Convolutional Neural Network (CNN) (Chellapilla et al., 2006). Understanding each algorithm and learning its strength and weaknesses will result in choosing the proper classifier for the dataset available. This could deliver an efficient model for the text classification application.

Performance evaluation

This is the final phase of Text Classification pipeline. Evaluating the model requires understanding how the model perform. There are many measures available for evaluation like Accuracy, Root Mean Square Error (RMSE), Loss, and others. Each has its benefits

and drawbacks. For example, calculating the *Accuracy* is the simplest evaluation method that can be used. It is easy and fast, but on the other hand, it does not work well on unbalanced datasets (Jin Huang and Ling, 2005).

2.1.2 Text Classification Algorithms

Learning about classification algorithms is very important, in order to choose a suitable algorithm for Text Classification system. There are many classification algorithms that could be used to classify a given text. Each algorithm has its benefits and limitations that must be acknowledged before being used. The following are some of the popular classifiers available, along with their main characteristics.

Logistic Regression

Logistic Regression (LR) is one of the earliest classification algorithms developed (Cox and Snell, 1989). LR is a linear classifier that estimates discrete values, i.e., binary values like 0/1 or true/false, based on a given set of independent variables. LR predicts the probabilities rather than the classes (Fan et al., 2008). Consider the probability function $P(Y = 0|x_i) = y_i$, LR model will train by calculating the probability y_i of the variable Y being 0 given the input value of x_i where $Y \in \{0,1\}$. The probabilities describe the possible outcomes of a single trial. The values of y_i would always be between 0 and 1 since they represent probabilities.

In classification, LR is very useful for understanding the influence of several independent variables on a single result variable. However, LR works only when the predicted labels

are binary. It also assumes that all variables used for prediction are independent of each other (Huang, 2015).

Naïve Bayes Classifier

The Naïve Bayes Classifier (NBC) is a classification algorithm that has been widely used for document categorization. The theory behind NBC is based on the Bayes theorem, which was found by Thomas Bayes between 1701–1761, with the assumption of independence between every pair of features (Pearson, 1925). In other words, NBC assumes that the presence of a particular feature is not related to the presence of any other feature. Given n the documents $d_1, d_2, ..., d_n$ to be classified, and $C = \{c_1, c_2, ..., c_k\}$ as a set of classes, where k is the total number of classes, the predicted class of document d will be c where $c \in C$.

The NBC algorithm can be used when the dataset is small. It requires very little amount of training data to estimate the parameters. It is computationally inexpensive, compared with sophisticated methods, and requires a very small amount of memory (Larson, 2010). On the other hand, NBC does not always give good estimations.

K Nearest Neighbors

The K-Nearest Neighbours algorithm (KNN) is a simple non-parametric algorithm widely used for text classification applications (Jiang et al., 2012). KNN can be used for both classification and regression problems (Kowsari et al., 2019). It retains all the instances results from training data and use them to classify a new instance. The new instance

is assigned to the most common class amongst its K nearest neighbours. Choosing the number of neighbours K is very critical. If $K = 1$, then the case is simply assigned to the class of its nearest neighbour.

The KNN algorithm works well against noisy training data, because it consults the closest neighbours of a given instance. KNN is more effective with large training data, more data means more close neighbours. However, KNN is computationally expensive (Sanjay and Nagori, 2018). It needs to compute the distance of each instance against all the training samples using a distance function, like the hamming distance.

Decision Trees

One of the early classification algorithms introduced to text classification problems is the decision tree (Quinlan, 1986). Given a data (attributes) and its classes, a decision tree algorithm produces a sequence of rules that can be used to classify the data. It splits the data into two or more similar sets based on the most significant attributes that makes the sets as distinct as possible (Aggarwal and Zhai, 2012). The goal is to create a tree based on the attributes that categorize the data into subsets. Choosing which attribute could be in parents' level and which one should be in child level is challenging.

The decision tree algorithm is simple and performs very fast in learning and predicting the results (Kowsari et al., 2019). It requires little data preparation, and can handle both numerical and categorical data. The most important advantage is that the learned decision tree is interpretable for the human users. However, it has some limitations as well. It is extremely sensitive to small disturbance in the data (Giovanelli et al., 2017), and can easily

be over fitted (Quinlan, 1987), tough adding a validation method and pruning the tree can help avoid those limitations (Giovanelli et al., 2017).

Random Forest

Random Forest is an algorithm used for text classification that fits a number of decision trees on different sub-samples of the dataset (Tin Kam Ho, 1995). The sub-sample size is the same as the original input sample size, but the samples are drawn with replacement. The idea behind the Random Forest classifier is to generate random decision trees. After training all trees as a forest, the algorithm uses average to improve the accuracy of the predictions (Wu et al., 2004).

The random forests algorithm gives a very fast model to train for text datasets, yet the model is slow in prediction once it is trained (Bansal et al., 2018). The more trees found in random forest, the more time it cost in the prediction stage. Therefore, reducing the number of trees could make the prediction faster.

Support Vector Machine

Support Vector Machine (SVM) (Manevitz and Yousef, 2002) is a popular technique which employs a discriminative classifier for document categorization. SVM was originally designed for binary classification tasks (Han and Karypis, 2000). However, it was adapted to solve multi-classes problems (Bo and Xianwu, 2006). This type of classification would be accomplished by breaking down the multi-classification problem into smaller sub-problems, all of which are binary classification problems.

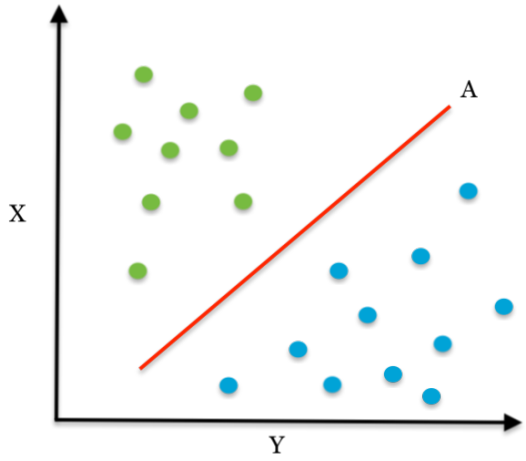

Figure 2.2: A simple SVM: line A separates a set of data into two categories

A simple description of SVM algorithm is a representation of the training data as points in a plane separated into categories by a clear gap that is as wide as possible. After that, test data are mapped into that space. Then based on which side of the gap they fell, the category they belong to is predicted. Figure 2.2 shows a simple SVM line A that separates a set of data into two categories.

2.2 Text Simplification

Text Simplification is a process used in Natural Language Processing to enhance the readability and comprehensibility of a given text. It reduces the lexical and structural complexity of a text, while maintaining the semantic meaning. There are many approaches used for Text Simplification such as; lexical simplification, syntactic simplification, explanation generation, statistical machine translation, and hybrid techniques. Some of these approaches are automated and some needs the involvement of manual processing to get

the best results.

Natural Language Generators (NLG) techniques such as machine translation, text summarization, and text simplification have some similarities; even if they are different tasks. For instance, machine translation is the process of automatically converting a natural language text into another language text. The process must preserve the meaning of the input text and generate a fluent text as the output. This output text does not care about the size of the text as long as the text is fluent in reading and preserves the same meaning of the input text.

On the other hand, text summarization is the task of reducing the size of the text by removing redundant or insignificant sentences and possibly shorten some sentences. The resulted short summary does not necessarily mean it is simpler or easier to read. To improve the readability of the final summary, text simplification could be applied. Text simplification's main task is to make the input text simpler and easier to read. This would be achieved by using easier more common synonyms, splitting long ambiguous sentences into two or more clear sentences, and adding explanations if needed. This could result in a longer output text, yet easier to read and understand than the original input text.

2.2.1 Beginning of Text Simplification

The first effort towards automated Text Simplification was made by *Boeing* for writers of Simplified English (SE) (Shardlow, 2014; Hoard et al., 1992). Boeing developed *a grammar and style checker* for the writers of their commercial aircraft manuals to help them keep

in accordance with the **ASD-STE100**[1] standards for Simplified English (SE) (Shardlow, 2014). For this reason, Boeing was adding more projects and work to the field of automated Text Simplification.

In 1994, the **SECC** project was found by Adriaens (1994) to develop a tool for technical writers who produce documents in Simplified English (SE). This tool checks if the document complies with the syntactic and lexical rules, and automatic correction is attempted wherever possible to reduce the need of human manual correction (Adriaens, 1994). Since then, there has been work on extending Text Simplification techniques to work on Natural Language in many areas like *syntactic* simplification by Chandrasekar and Srinivas (1997) and *lexical* simplification by Devlin and Tait (1998). As a result, several methods were developed to improve syntactic simplification and lexical simplification (Shardlow, 2014).

2.2.2 Text Simplification Approaches

Text Simplification is one of the NLP tasks that performs text-to-text generation. The purpose of simplifying a text is to increase its level of readability. A text simplification tool could be developed using one approach or combining several simplification approaches. These approaches could focus on lexical simplification, syntactic simplification, explanation generation, statistical machine translation, and text simplification approaches in languages other than English.

[1] ASD Simplified Technical English specification (**ASD-STE100**), an international specification for the preparation of technical documentation in a controlled language: http://asd-ste100.org

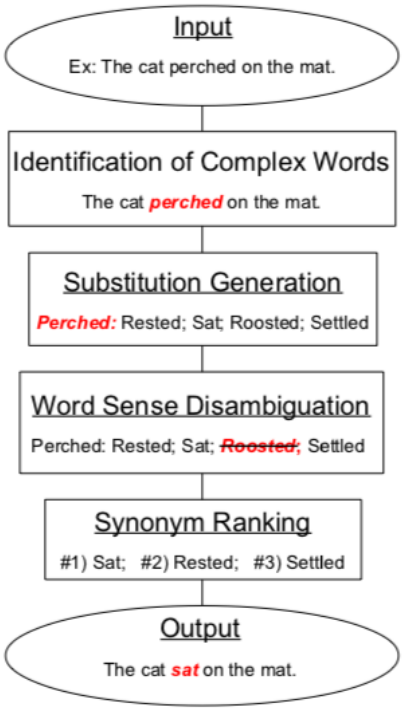

Figure 2.3: Lexical Simplification phases from (Shardlow, 2014)

Lexical Simplification Approach

Lexical Simplification works mainly on the lexical side of the text. This approach uses standard spelling, a standard format for numbers and dates, and specifies a dictionary to identify and replace complex words with simpler proper substitutions. It also excludes specific acronyms, synonyms, ambiguous pronouns, and includes relative pronouns. Moreover, lexical simplification rules out ambiguous anaphoric references, ambiguous conjunctions like "*as*", double negations, and limits words that signal negation (Siddharthan, 2014). The main task is to simplify complex and ambiguous aspects of vocabularies (terms/words). This approach does not involve any modification to simplify the grammar of the document text nor changing the structure of the sentences.

There are four main phases followed in lexical simplification, as explained in (Shardlow, 2014). These phases could be expressed as the following:

- **Phases 1:** Identify the complex and ambiguous vocabulary terms in a given document.
- **Phases 2:** Generate a list of substitutions for each vocabulary term found in the previous phase.
- **Phases 3:** Filter the substitution list and keep only the substitution words that would be meaningful in the given context.
- **Phases 4:** Rank the remaining substitution words based on their simplicity. Then the most simple substitution word would be used as a replacement for the original complex word.

The four phases are shown in Figure 2.3 with an example. These phases do not explicitly treat metaphors, slangs, or idioms. So, when applying those four phases, the text's authors have to be as specific as possible to avoid ambiguity.

Syntactic Simplification Approach

The Syntactic Simplification approach identifies the complex syntactic structure of a given sentence and then rephrase it into a simpler structure. The structure of a complex syntax could be identified as long sentences, sentences with passive voice, or poorly written sentences. Some readers may struggle to follow the text or lose interest at some point in

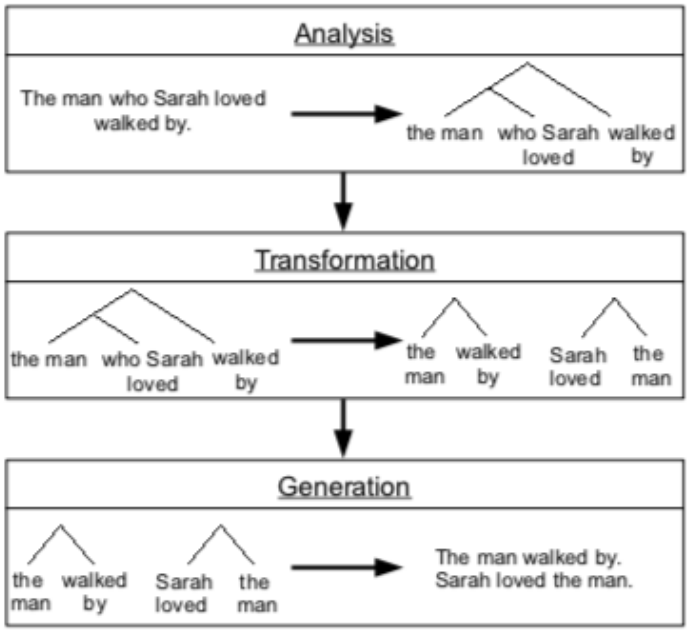

Figure 2.4: Syntactic Simplification phases from (Shardlow, 2014)

a sentence and eventually give up trying to read it or understand it. So, splitting long sentences into several clauses might help readers to continue reading the text (Shardlow, 2014). In some cases, it is hard to distinguish between subject and object when the passive voice is used in the sentence. Therefore, these sentences have to be rewritten in a simpler way.

The Syntactic simplification approach consists of three phases as explained in (Shardlow, 2014). The first phase includes, analyzing the text and identifying its structure and parse tree. At this phase, words and phrases are grouped together into *super-tags* (Shardlow, 2014). These super-tags can be joined together, with respect to grammar rules, to provide a structured version of the text. Also, during this phase, the complexity of a sentence is determined to decide whether it will require simplification or not.

The second phase involves modifying the parse tree according to a set of rewrite rules.

These rewrite rules perform the simplification operations such as sentence splitting, clause rearranging, and clause removing. Although there are techniques for automatically inducing these rules, most systems use hand-written rewrite rules to accomplish a syntactic text simplification (Shardlow, 2014).

The last phase of syntactic simplification is modifying the sentences to improve their coherence, relevance, and readability level. The three phases are shown in Figure 2.4 with an example from (Shardlow, 2014).

Explanation Generation Approach

Explanation generation is the approach that takes a difficult term or concept in a text and associates it with an additional information, in order to make it more readable and understandable. This approach is more appropriate for scientific documents that involve scientific terms, where the reader is expected to learn and understand those terms(Shardlow, 2014). These terms are categorized by their semantic type, such as disease name, anatomical structure, device type, etc. Explanations are then generated by finding an easier description that best expresses the complex term. This explanation is then added as a short connecting phrase to explain the complex term (Shardlow, 2014).

Statistical Machine Translation Approach

Statistical machine translation is an established approach in machine translation that is characterized by the use of machine learning methods (Lopez, 2008). It involves automatic techniques that convert the lexicon and syntax of one language into another language. This

process results in a translated text.

A recent development in the traditional translation applies simplification to improve the readability in the target language, which makes it useful for language learners. It is also useful to transform source and target texts to a common format to improve their alignment, thus improving the translation accuracy (Shardlow, 2014). This approach is very convenient for simplification where we could treat complex text and simple text as two different languages. Then using translation approaches, complex text can be translated to simple text.

Simplification for Other Languages

In natural language processing applications, most of text simplification research is done for the English language. However, text simplification is also applied across many other languages. Every language has its own specific characteristics. It is non-trivial to re-implement existing text simplification techniques into other languages. Every language has different characteristics that need to be handled in a different way. Some languages, like Latin and Swedish, use complex verb conjugations. For example, specific forms of the verbs express passive voice sentences. Other languages, like Mandarin Chinese, have unchangeable form of verbs when expressing passive voice sentences. This means their verbs do not have any tenses. Several projects focus on re-implementing existing techniques and adapting them to their own language.

2.3 Deep Learning and Text Processing

Deep Learning (DL), also known as Deep Neural Networks (DNN), is a subset field of machine learning (ML) and artificial intelligence (AI) that gained popularity in the past few years. According to (Schmidhuber, 2015), the concept *Deep Learning* was introduced to the machine learning community for the first time in 1986, by Rina Dechter. This system involves many pattern recognition and multiple layers of neural networks. These deep neural networks are capable of learning unsupervised using an unstructured data to extract and process its characteristics. The layers in this network imitate the work of the human brain in processing data and creating patterns for use in decision making (Bengio et al., 2009; Goodfellow et al., 2016).

There three types of layers: an *Input Layer*, an *Output Layer*, and several *Hidden Layers*. Each layer uses the output of the previous layer as an input to learn from. The input layer is usually embedding vectors as shown in Figure 2.5. The output layer contains a number of neurons (or nodes) equal to the number of classes needed for classification (Mittal, 2018).

2.3.1 Deep Learning Distributed Representations

The term Distributed Representation refers to the process of features creation, in which similar inputs have similar features. It is a technique used in Deep Learning to expressed words as vectors that represent pieces of data. Using distributed representation values gives us the ability to capture meaningful semantic similarity between data. There are two kinds of distributed representations; Word Embeddings and Character Embeddings.

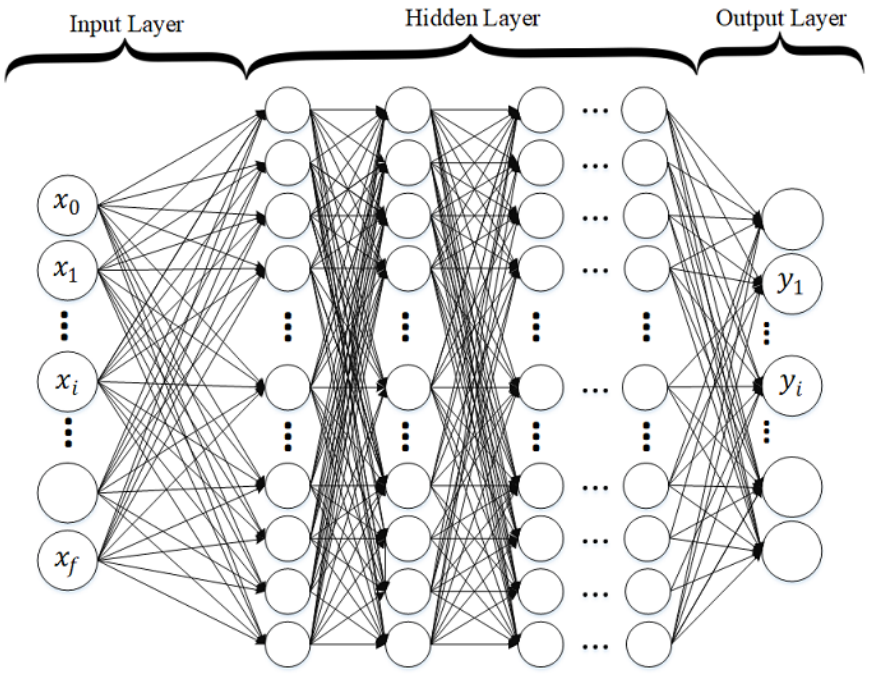

Figure 2.5: Layers of a Deep Neural Network.

Word Embeddings

Word Embedding (WE) is a natural language modelling technique that uses a mapping function to map words or phrases from a vocabulary word to a corresponding vector in a n-dimensional space. The main idea of using WE is to transform a text into a vector of numbers. The WE distributional hypothesis is that words with similar meanings tend to occur in similar contexts. Therefore, the vectors for those words try to capture the characteristics of the neighbouring words. The main advantage of distributional vectors is that they capture the similarity between words. Measuring similarity between vectors is then possible using measures such as cosine similarity (Young et al., 2017). WE is used as a pre-processing layer for the text data in DL models. There are two advantages of using WE: dimensional reduction which gives more efficient representations, and contextual similarity

that shows more expressive representation.

- **GloVe**, proposed by Pennington et al. (2014), is a famous word embedding method that uses unsupervised learning algorithm to obtaining vector representations for words. It is a count-based model that counts accumulating word co-occurrences to produce a matrix. This matrix is then factorized to get lower dimensional representations which is done by minimizing a reconstruction loss (Young et al., 2017). The code of the algorithm and the pre-trained word vectors are both made available for public by *The Stanford NLP Group*[2].

- **Word2vec** is another method used to produce word embedding. This method was proposed by (Mikolov et al., 2013). It takes a large corpus of text as its input, and generates a vector space with several hundred dimensions as its output. Each word in the corpus is assigned to a corresponding vector in the space. There are two ways to implement word2vec: CBOW (Continuous Bag-Of-Words) and Skip-gram. The two models work on opposite ways of each other. **CBOW** computes the conditional probability of a target word given the context words surrounding it across a window of size k. On the other hand, the **skip-gram** model predicts the surrounding context words given the central target word (Young et al., 2017).

Character Embeddings

Character embeddings are designed for helping text classification. Some languages' text is not composed of separated words, but individual characters that carry the semantic

[2]https://nlp.stanford.edu/projects/glove/

meaning of words, such as in the Chinese language. With large number of vocabularies, or characters in this case, a phenomenon called the unknown word phenomenon is common, also known as out-of-vocabulary (OOV) words.

Character embeddings naturally deals with OOV, since each word is considered as no more than a composition of individual letters (Young et al., 2017). When word embeddings are able to capture syntactic and semantic information only, character embedding can build any word as long as those characters are included (Likhomanenko et al., 2019). Many researches were done in this field by Facebook AI Research (FAIR)[3] and some of them made their data publicly available.

2.3.2 Deep Learning Models

Deep learning models has proven to be the state-of-the-art models across many domains. They provide high accuracy results with less features needed. There are many models used for deep learning. Here we will briefly explain two popular models that could be used in text classification and simplification, Convolutional Neural Networks and Recurrent Neural Networks.

Convolutional Neural Networks

Convolutional Neural Network (CNN) is a deep learning model that is popularly used for text classification (Lai et al., 2015). It was originally designed for image processing, but also found to be effective for text classification (Lecun et al., 1998). CNN is composed

[3]https://ai.facebook.com/

of three types of layers that considered to be the model building blocks: convolutional, pooling, and fully-connected layers. The convolution and pooling layers perform feature extraction, while the fully-connected layer maps the extracted features into a final output.

The convolution layer is always the first layer and could include multiple layers that can be stacked to provide multiple filters for the input. A pooling layer reduces the size of the output from one layer to the next, while preserving important features. This could reduce the computational complexity in the network (Kowsari et al., 2019). These layers are not fully connected. The neurons from one layer do not connect to every single neuron in the following layer. The final output of the convolution and pooling layers is the input to the last layer, the fully-connected layer. CNN learns the filters automatically, without explicitly specifying them. A single filter is applied and shared across different parts of an input. These filters help extracting relevant features from the input data.

Recurrent Neural Networks

Another deep learning model used for text classification is recurrent neural network (RNN) (Sutskever et al., 2011). RNN was initially designed to work with text sequence prediction problems due to its ability to memorize long-term dependencies (Fu et al., 2016). But with time lags increase, the gradients of RNN may vanish.

To overcome this problem, certain structures of RNN, like Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) and Gated Recurrent Unit (GRU) (Cho et al., 2014), were proposed to include "forget gates". A "forget gate" was designed to give the memory cells the ability to determine when to forget certain information (Fu et al., 2016).

This will determine the optimal time lags. Because of this feature, short-term memory is not an issue for RNN. GRU and LSTM have a similar architecture, with GRU having a simpler computation and implementation (Fu et al., 2016). LSTM uses mainly three gates to regulate the amount of information allowed into each node, whereas GRU has only two gates. This means that GRU consumes less memory than LSTM, which means it trains faster. Both structures could be helpful with different datasets as explained in a study by Yang et al. (2020). The same study suggests GRU performance surpass LSTM when using long input text and small datasets; while LSTM works better with shorter input text and larger datasets.

2.3.3 Transformers and Text-to-Text Generators

A transformer (Vaswani et al., 2017) is a deep learning models that has been popularly used in solving NLG problems. It was originally proposed as a Seq2Seq model for machine translation, but transformer-based pre-trained models proved to be state-of-the-art method (Qiu et al., 2020) in solving many problems in NLG tasks like caption generation, machine translation, document summarization, text simplification, and even question answering. As mentioned, DL uses neural networks as the central component to process and analyze written text and then produce the output results. It processes large amount of data automatically without the need of manual human work. A special data training and text mining is used to discover patterns in a given text (customer complaints, physician notes, or news reports, etc.) to get a useful output. Some NLP problems are known as text-to-text generators, like summarization, chat-bots, machine translation, and simplification.

The Transformer architecture can be used in several ways. Usually in seq2seq models, the full Transformer architecture (encoder-decoder) is applied as we did in this work for text simplification (Lin and Wan, 2021). The encoder could be used alone as well, and the outputs of the encoder will represent the input sequence such as BERT (Devlin et al., 2018). Also, the decoder only can be applied without using the Encoder or the attention mechanism. This is often used for text generation, such as in the GPT-3 model, described in the next section.

GPT-3 Model

The GPT-3 model, created by OpenAI [4], is a famous very recent transformer that solves NLP problems using the decoder part. It is a text-to-text generator that is able to perform several NLP tasks including machine translation, question answering, and summarization in a zero-shot setting (Dale, 2021). This means that the model trained to maximize the likelihood of a sufficiently varied text corpus begins to learn how to perform other tasks without the need for explicit supervision (Radford et al., 2019). GPT-3 has about 175 billion parameters and it is trained on over 45 Terabyte of text data from different datasets. However, GPT-3 is difficult to use (in terms of resources required) and does not perform text simplification , this is why we did not use it in this work.

Seq2Seq Model with Attention Layer

Sequence-to-sequence (Seq2Seq) model is a transformer that uses encoder and a decoder units to solve NLG problems. Seq2Seq models have excelled at solving text-to-text genera-

[4] https://openai.com/blog/better-language-models/

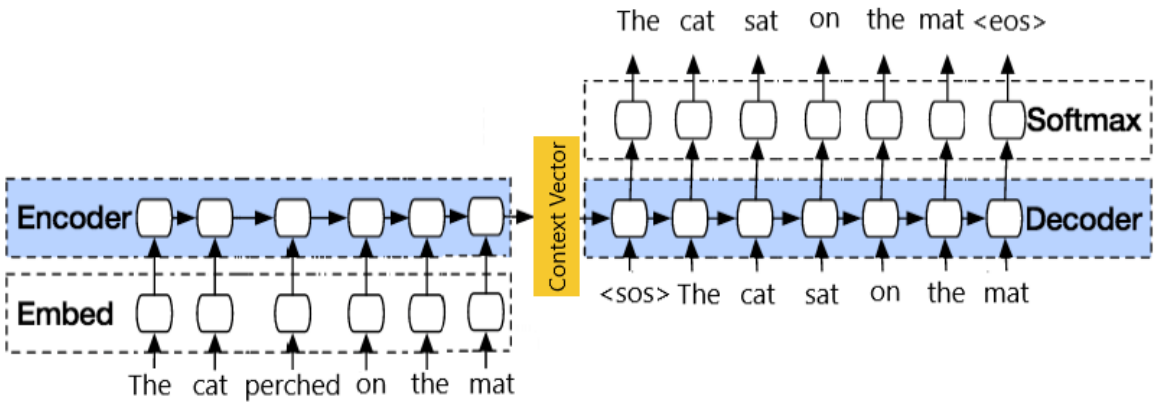

Figure 2.6: Seq2Seq model architecture

tion problems which involve generating natural language sentences Sriram et al. (2017). A Seq2Seq model should contain at least two RNNs: an Encoder and a Decoder Sojasingarayar (2020). The RNN could be LSTM or GRU, depending on the task. The Encoder takes a sequence (sentence) as an input and processes one token (word) at each time step. The purpose of using the Encoder is to convert a sequence of tokens into a fixed size context vector that encodes the important features in the input sequence. Each hidden state influences the next hidden state as shown in the Figure 2.6. The Decoder generates another sequence, one token (word) at a time. At each time step, the decoder is influenced by the context vector and the previously generated tokens.

When dealing with long input sequences, some of the features at the beginning of the sequence could be forgotten in the context vector. To solve this issue, an attention layer (Luong et al., 2015) was introduced to keep track of all the input sequence stages. This layer takes the output of every hidden layer into consideration. It allows the context vector to preserve the input sequence from the beginning to the end.

Using the methods and approaches described in this chapter, many researches and

experiments were performed in text classification based on readability level and text simplification. These projects are explained and discussed in the next chapter.

Chapter 3

Related Work

In this chapter, we highlight the work related to text classification based on the readability level of the text, followed by a description of the available text simplification projects established for different languages. At the end, we will mention some simplification systems that used deep learning techniques as their main component.

3.1 Available Libraries

One of the reasons machine learning and deep learning are becoming mainstream is because of their open source libraries available for developers interested in applying them to their projects. These libraries offer functions with an acceptable level of abstraction and simplification, mostly using *Python*.

Scikit-learn [1] is a general purpose library for machine learning. It provides many simple

[1] https://scikit-learn.org/

and efficient algorithms with features that could help with working on text classification. It is a very friendly library for beginners to start text classification.

NLTK [2] is a popular library specialized in Natural Language Processing (NLP). Many people contributed to this library, making it more practical and efficient in handling text processing. It is very helpful when performing text classification. It provides many tools for the machine to manipulate the text. For instance, it facilitates splitting paragraphs into sentences, or even words.

SpaCy [3] is a newer NLP library that represent an updated version of NLTK. It has more minimal and straightforward functions than NLTK. For instance, SpaCy only implements a single stemmer while NLTK has nine different options. Also, SpaCy integrated word embeddings into its library which can be helpful to improve accuracy in text classification.

Deep Learning Libraries like Keras[4], TensorFlow [5], and PyTorch [6] are all specialized libraries. *Keras* provides a simple start to build a recurrent neural networks and convolutional neural networks. However, *TensorFlow* is a more popular open source library among developers for implementing deep learning algorithms. Although Keras is easier to learn, TensorFlow is the leader in the deep learning field. Finally, *PyTorch* is the alternative to TensorFlow which provides an extensive DL library. It allows the operations to be carried out and processed on the GPU.

[2] https://www.nltk.org/

[3] https://spacy.io

[4] https://keras.io

[5] https://www.tensorflow.org

[6] https://pytorch.org

3.2 Classifying Text by Readability Levels

Finding the readability level of a text was an essential mission since the beginning of text simplification. Many formulas were put together to find the readability level of a text like *Flesch-Kincaid Grade Level* (Kincaid et al., 1975) and *Gunning Fog Index* (Gunning, 1969). Recently, several research works have been done in text classification using machine learning techniques in general to classify a text to its readability level (Aluisio et al., 2010; Bessou and Chenni, 2021; Marvin Imperial and Ong, 2021) (but not using deep learning techniques).

In (Scarton et al., 2018), the authors used sentences from Newsela Corpus to preform text classification. They used basic machine learning models to classify sentences into two categories, complex vs. simple, without specifying the level of simplicity. Yeakel and Tzeng (2019) also tried to classify documents into students grades, 2 to 12. However, they used too many classes over small number of documents as training data. This resulted into poor performance. Another work, found in (Štajner et al., 2017), classified documents into five levels based on their simplicity. The paper mentioned using basic machine learning classifiers only like SVM and Random Forest. Also in (Larson, 2010), SVM was employed to improve the performance of text classification based on readability levels for Swedish input text.

Even though SVM is a good model for classification, deep learning methods proved to show a significant improvements in solving NLP problems, as Giovanelli et al. (2017) and Li (2017) mentioned in their work. This makes deep learning methods the perfect method to use for classifying a text to its readability level. We could not find papers that classifies

documents into several readability levels using deep neural network and we are filling in this gap in our work.

3.3 Text Simplification in Different Languages

Text simplification is a major challenge in all languages. Many projects and tools were found for text simplification over different languages. Most of them were developed to assist people with disabilities or learning difficulties.

KURA

One of the earliest projects found in text simplification was KURA. The KURA project (Takahashi et al., 2001) is a Japanese project that aims to simplify Japanese language text for deaf students. It developed a lexico-structural paraphrasing engine. KURA introduced the concept of phrase-based simplification which identifies then simplifies complex terms (Inui et al., 2003).

PorSimples

Similarly, the PorSimples project (Aluisio and Gasperin, 2010) developed text adaptations tools for Brazilian Portuguese. The tools developed serve both people with poor literacy levels and authors who produce texts for these audience. It is one of the largest text simplification projects with three main systems and many types of simplification techniques investigated (Shardlow, 2014). Its main purpose is to increase the comprehension of written

texts through simplification of their linguistic structure. It replaces uncommon words with more usual words. It also changes the sentence syntactic structure to an easier form to avoid ambiguity.

SIMPLIFICA

SIMPLIFICA (Scarton et al., 2010) is another tool for producing simplified texts in Portuguese. It helps authors write simple texts for poor literate readers. The author simply writes a text and gets back the simplified version of it. SIMPLIFICA uses lexical and syntactic simplification features to assist the readability of the text targeting Brazilian Portuguese. The tool preforms simplification on sentence level.

Simplext

The Simplext project (Saggion et al., 2011) develops tools that produce a simplified text for Spanish language. It has a particular focus on producing applications of text simplification for dyslexic readers (Shardlow, 2014).

Lexi

Lexi (Bingel et al., 2018) uses personalized adaptive lexical simplification to simplify Danish text. It is an open-source tool that could be extended to other languages. For other languages, it requires training a language model and word embeddings using monolingual corpus.

LexSiS

Lexical Simplification system for Spanish (LexSiS), was found by Bott et al. (2012). It uses the simplicity measures such as word length, and word frequency in Spanish language. LexSiS improves the performance of lexical simplification by using context vectors. This system uses the Spanish Open Thesaurus lexical database, which lists over 21,000 target words and provides a list of substitution sets for each word (Al-Thanyyan and Azmi, 2021). LexSiS replaces every word with the best candidate found in the lexical database.

Other works

There are other simplification researches done in languages other than English. In (Ferrés et al., 2017), it combines lexicon-based generation and decision trees based using Portuguese, Galician, Spanish, and Catalan languages. The simplification process goes through five phases: document analysis, complex word detection, word sense disambiguation, synonyms ranking, and language realization. Another work in simplification, (Pimienta Castillo, 2021) which applies lexical simplification to simplify English, Spanish, and German languages.

3.4 Deep Learning used in Text Simplification

As mentioned, Deep Learning is the-state-of-the-art method that solves many problems in NLP, such as text classification, speech recognition, caption generation, machine translation, document summarization, and even question answering. DL uses neural networks as

the central component to process and analyze written text and then produce the output results. It processes large amounts of data automatically without the need of manual human work. Special training data is needed for text mining techniques to be able to discover patterns in a given text like customer complaints, physician notes, news reports, etc. Using these patterns and data would result in producing useful output.

There are many NLP tools available that use Deep Learning to solve their problems. Some of these problems, like text summarization and text simplification, are known as text-to-text generators. In summarization, there are many researches like (Song et al., 2019) and (Abdi et al., 2021) and tools like NAMAS(Rush et al., 2015) and GPT-3(Dale, 2021) found to summarize a given text using DL techniques. On the other hand, there are only a few tools found for text simplification using DL techniques. Some of these tools are DRESS and EditNTS.

DRESS

There are many tools that can be used to solve NLP problems using Deep Learning, but there are very few tools that can simplify a given document and increase its readability using Deep Learning. DRESS (Deep REinforcement Sentence Simplification) (Zhang and Lapata, 2017) is one of the few NLP tools that provides a reinforcement learning-based text simplification model. DRESS treats sentences individually for simplification. It simplifies a given text sentence by sentence rather than the whole document or paragraph at once. In addition, splitting long sentences is not supported in DRESS. Also, for a given text, DRESS allows only one level of simplification instead of several simplified levels of a given

input text. The code is mostly written in Lua and few parts in Python, and it is publicly available on Github.

EditNTS

Another state-of-the-art sentence simplification model that uses deep learning methods is EditNTS (Dong et al., 2019) . The model learns explicit edit operations (ADD, DELETE, and KEEP) via a neural programmer-interpreter approach. It is trained to predict a series of edit operations on each word of the original complex sentence. Then, using this series of operations, it generates the simplified sentence. EditNTS favors generating short sentences with big semantic deviation (Lin and Wan, 2021). The tool is written purely in Python and available in GitHub for public.

In this chapter, we presented the libraries used in machine learning and deep learning techniques. Then we discussed the work accomplished in text classification based on the readability level of the text. After that, tools developed for text simplification in different languages were described. Finally, some text simplification models that uses deep learning techniques were explained. Each of these models (classification and simplification) need a dataset to be properly trained. The next chapter will discuss the different datasets available for text classification and simplification.

Chapter 4

Datasets

This chapter reviews the datasets that could be used for text classification and simplification. It includes datasets with aligned sentences and others with aligned documents. Some have 1-to-1 sentence alignments, and others include 1-to-N sentences alignments to allow for sentence splitting.

4.1 Wikipedia Corpus

4.1.1 Parallel Complex-Simple Dataset (PWKP)

PWKP is a large-scale parallel dataset for sentence simplification that was collected by (Zhu et al., 2010). The dataset was collected by aligning sentences from *English Wikipedia*[1] and *Simple English Wikipedia*[2]. The targeted audience of *Simple English Wikipedia* includes

[1] http://en.wikipedia.org

[2] http://simple.wikipedia.org

children, adults with low literacy, and people learning the English language. Therefore, the authors are requested to use common words and short sentences to write the articles (Zhu et al., 2010). The PWKP dataset pairs each sentence from English Wikipedia with one *(1-to-1 alignment)* or more *(1-to-N alignment)* sentences from Simple English Wikipedia, where *(1-to-N alignment)* would occur in case of sentence splitting. The PWKP dataset has over 108K sentence pairs derived from 65,133 Wikipedia articles. The dataset is publicly available online [3].

4.1.2 Coster-Kauchak Dataset

The Coster-Kauchak dataset is another text simplification dataset that uses sentence alignment. It was generated by Coster and Kauchak (2011). The dataset contains 137,362 aligned lines of sentences extracted by pairing the Simple English Wikipedia with the English Wikipedia.

This dataset consists of two files: *wiki.normal* and *wiki.simple*. Each file includes 137,362 lines of sentences. The n^{th} line in *wiki.normal* corresponds to the n^{th} line in *wiki.simple*. Each line may contain one or more sentences. So the relationship between the sentences found in each line from both files could be *1-to-1* (simple alignment), *1-to-N* (sentence splitting), or *N-to-1* (summarisation). The dataset is made publicly available online [4] by David Kauchak.

[3] https://fileserver.ukp.informatik.tu-darmstadt.de

[4] https://cs.pomona.edu/ dkauchak/simplification/data.v1/data.v1.tar.gz

4.1.3 WikiSmall Dataset

Later, in 2013, another dataset was introduced by Kauchak (2013) and called WikiSmall Dataset. The dataset is an updated version of Coster-Kauchak dataset, discussed in Section 4.1.2. This updated dataset, WikiSmall, uses sentence alignment with updated Wikipedia data and improved text processing. The Wikipedia data used in WikiSmall dataset was downloaded in May 2011.

WikiSmall dataset contains 167,000 aligned sentence pairs. The dataset contains two files: *normal.aligned* and *simple.aligned*. The files have the same number of lines and the n^{th} line in one file corresponds to the n^{th} line in the other file. The dataset contains metadata about each sentence in the corpus. It keeps track of where each sentence came from, including the article title and paragraph number. The sentences in the dataset are separated by tabs (Kauchak, 2013). In WikiSmall dataset you could find several types of alignments between the two files. *N-to-1* alignment, means summarising two or more sentences into one sentence. Also, *1-to-N* alignment found in the dataset, which represents splitting a sentence into two or more sentences. WikiSmall dataset is also publicly available [5].

4.1.4 WikiLarg Dataset

WikiLarg was also prepared by Kauchak (2013) using Wikipedia data downloaded in May 2011. Similar to WikiSmall, the dataset contains two files: *normal.txt* and *simple.txt*. The format of the files used in WikiLarg is the same one used in WikiSmall dataset, explained

[5] https://cs.pomona.edu/~dkauchak/simplification/data.v2/sentence-aligned.v2.tar.gz

in Section 4.1.3. The data was collected using around 60,000 English Wikipedia articles and Simple English Wikipedia articles paired with each other based on their titles. The dataset contains all the Simple English Wikipedia articles that have a corresponding article in the English Wikipedia during May 2011. WikiLarg dataset contains over 3,856k aligned lines. Because of its large size of data, compared with WikiSmall dataset, it is often used for training purpose. WikiLarg dataset is publicly available online[6] as well.

4.2 Datasets that used Mechanical Turk

4.2.1 Turk Corpus

The Turk dataset was collected by Xu et al. (2016) through Amazon Mechanical Turk. For every complex sentence, it has multiple (8) simplification references. The dataset contains 2,350 sentences split into 2,000 instances for tuning and 350 for testing. This dataset is usually used for tuning and testing. Other larger datasets could be used for training, like WikiLarg.

4.2.2 ASSET Corpus

A recent dataset for assessing sentence simplification in English aligned with Turk Corpus (ASSET) was released by Alva-Manchego et al. (2020). It contains the same set of original complex sentences found in Turk Corpus. ASSET has 1-to-1 and 1-to-N alignments, with 10 simplification references per original complex sentence collected through

[6] https://cs.pomona.edu/~dkauchak/simplification/data.v2/document-aligned.v2.tar.gz

Amazon Mechanical Turk. Same as Turk Corpus, ASSET contains 2,350 sentences split into 2,000 instances for tuning and 350 for testing. However, their multiple references were produced by executing several rewriting transformation operations for simplification, like lexical paraphrasing, compression, and sentence splitting. Alva-Manchego et al. (2020) showed that human judges found ASSET Corpus type of simplifications simpler than those from Turk Corpus.

4.3 Newsela Corpus

Newsela [7] is a corpus of thousands of news articles professionally simplified to several versions with different reading complexity levels. The corpus initially contains 10,786 documents in English and Spanish languages. The documents have different readability levels vary from 0 to 5 targeting students of grades between 2 and 12. The corpus has 2,154 original documents labeled with *level 0* which means they are not simplified and are in their most difficult reading level. Each document, under *level 0* category, has a certain article topic and title. Those documents are simplified into four, or sometimes five, versions with the same article topic but different titles and readability levels. *Level 1* label represents to the first level of the simplified document, and *Level 4* label means the most readable version of the document. The higher the readability number, the simpler the document text.

Each simplified version is written by expert editors. Newsela provided a metadata for all the documents in the corpus. The metadata provides information for every document

[7] https://newsela.com/data/

including its language, article topic, article title, readability level, and targeted students grade. These information helps users in classifying or using the documents. This corpus, Newsela, is very well structured because it is reviewed and corrected manually by human experts. A dataset and scripts created by Xu et al. (2015) was constructed from an earlier version of this corpus. The corpus is not publicly available but could be requested from their website.

In this chapter, we presented all the available datasets that could find for text simplification and classification based on readability level. The next chapter will explore the methods and models that we will use for classifying and simplifying texts. It will also show the evaluation methods that we will apply to measure the performance.

Chapter 5

Methodology

This chapter explains the methodology we used for our text classification component and our simplification component. It also discusses the available readability metrics and the evaluations measures that we applied to assess the performance of the models.

5.1 Classifier Model

There are many models that could be used to perform text classification and simplification. Some of these models are simple machine learning models and some are powerful deep learning models. In our classification task, explained in Chapter 6, we used five categories to label the documents in our dataset with their readability level. To choose which model works best with this task, we will relly on experimentation and on several studies, including (Nayak et al., 2013).

5.1.1 SVM

The study in (Nayak et al., 2013) mentioned that using SVM classifier performs well with multiple labels and a small number of features. SVM uses less memory since it uses a subset of training points in the decision phase (Madi and Baba-Ali, 2019). However, it consumes too much time when training on large datasets and preforms poorly with overlapping classes. It is also sensitive to the type of kernel used.

SVM has two important parameters: **Gamma** and **C**. *Gamma* is a parameter of the RBF kernel and it symbolizes the '*spread*' of the kernel and therefore the decision region. When *gamma* value is too low, the '*curve*' of the decision boundary is very low and thus the decision region is very broad. But when *gamma* is too high, the '*curve*' of the decision boundary is high, which means only the near by points are considered.

C is a parameter of SVC and it represents the penalty for misclassifying a data. When C is small, the classifier accepts misclassified data points . But when C is too large, the classifier heavily penalize misclassified data. Therefore it bends over backwards to avoid any misclassified data points, resulting in overfitting. The Figure 5.1 shows how C and Gamma different values affect SVM model performance. In our work, we used grid search to set C and Gamma values to 100 and 0.001 respectively.

5.1.2 Random Forest

The Random Forest (RF) classifier is also a good classification model that could be used for datasets with multiple labels. It provides a consistent performance across different datasets. Random Forest uses *decision tree* as a base algorithm. It identifies the important

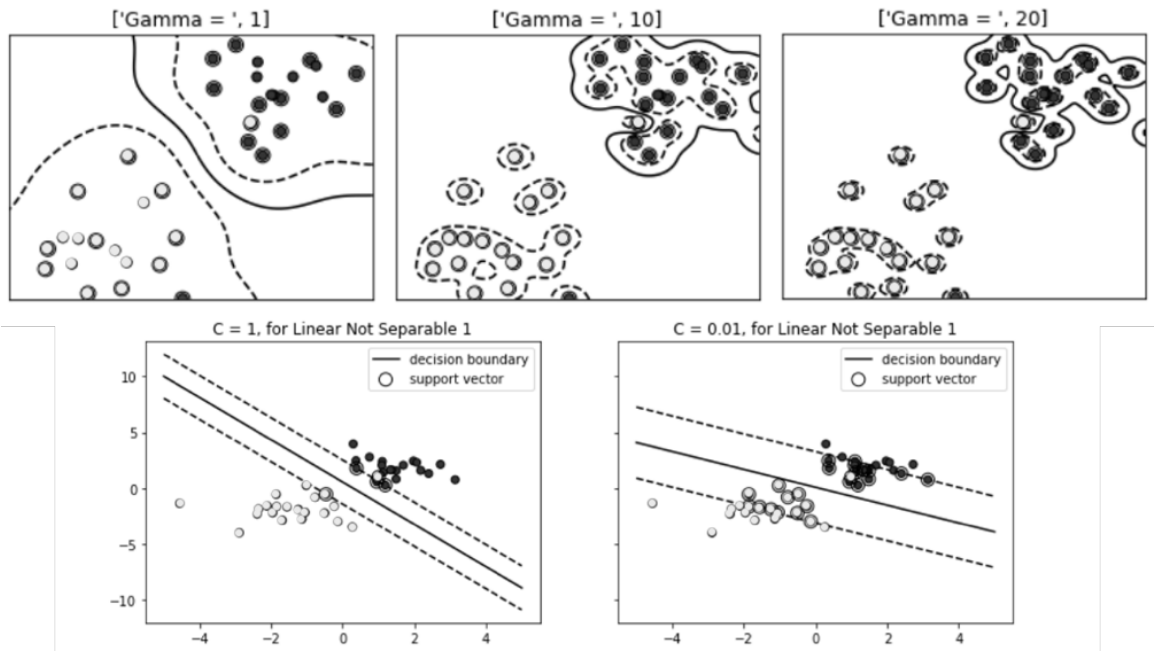

Figure 5.1: C and Gamma affecting SVM performance from Chen (2019)

features and gives them a rank in the process. So, all data with similar feature values will be assigned to the same class. That is why (Nayak et al., 2013) recommended using the Random Forest classifier for data with large number of labels. To avoid overfitting, RF classifier could pruning the tree by tuning the value of the maximum depth allowed in the classifier.

5.1.3 CNN

Convolutional Neural Network (CNN) is also a good classifier as it uses deep learning concept to preform. It has proven to work well in text classification tasks where the selected features go beyond the words in the text given, as shown in (Amin and Nadeem, 2018). Experimenting with other deep learning classifiers is left as future work.

The CNN model consists several layers that could be divided into two types: a *convolu-*

tion layers, that splits the input data features for analysis; and a *fully connected layer*, that uses the output of the convolution layer to predict the best class for the input document. It could also contain *pooling layers* that helps scale down the amount of information the convolution layer generated for each feature and maintains the most essential information. It is not unusual to have convolution and pooling layers repeated several times in a CNN classifier.

5.2 Readability Metrics

There are many readability metrics that could be used to measure the readability level of a text, Flesch-Kincaid, Gunning Fog, Dale Chall, Automated Readability Index (ARI), SMOG, and more. We will be focusing on Flesch-Kincaid and Gunning Fog as they are the most popular metrics.

Flesch-Kincaid is still used by the Microsoft Office Packet (Microsoft-Office, 2021) for readability testing. In the 1990, the US Department of Defense started using the Flesch-Kincaid as a standard test for readability. Add to that, it is still used mostly in the education field. For Gunning Fog, it is particularly popular in the management sector. Therefore, it is frequently used to evaluate the readability of annual reports.

Flesch-Kincaid

Flesch-Kincaid (Kincaid et al., 1975) is the famous readability metric. It was initially found for US Navy to measure the difficulty level of technical manuals used in training. The Flesch-Kincaid formula, shown in Equation 5.1, predicts a reading grade level of

written materials. The formula depends on the average number of words in the sentences, and the average number of syllables of the words. It gives back a score that varies from 0.0 to 100.0 where smaller scores means very difficult text and the higher the score, the easier the text to read and understand.

$$0.39 \left(\frac{totalWords}{totalSentences} \right) + 11.8 \left(\frac{totalSyllables}{totalWords} \right) - 15.59 \tag{5.1}$$

Gunning Fog Index

Another famous metric is *Gunning Fog Index* (Gunning, 1969) that calculates the FOG index of a given text. This calculated index determines whether a text can be read easily by its intended audience or not. Texts intended for wide audience need to have a Gunning Fog Index less than 12 (Milheim, 2018). The higher the index number, the lower the readability level "*Foggy text*", and vice versa.

Gunning Fog Index uses the formula shown in Equation 5.2. It is similar to Flesch-Kincaid formula. It considers the average sentences lengths in term of words, and syllables to predict the readability level of a text. It uses the number of syllables to count a hard-word factor. According to Gunning, a *hard-word factor* is the portion of complex words among all other words. He defined *complex words* as all the words of three syllables or more (Zhou et al., 2017), excluding proper names or verb forms that become three syllables by adding *-ed* or *-es* (Gunning, 1969).

$$0.4 \left[\left(\frac{words}{sentences} \right) + 100 \left(\frac{complexWords}{words} \right) \right] \tag{5.2}$$

5.3 Simplification Model

5.3.1 Seq2Seq Architecture

Sequence-to-sequence models are popularly used in solving most of text-to-text generation problems, including text simplification. A Seq2Seq model consists at least two RNNs, an Encoder, and a Decoder as Sojasingarayar (2020) explained. The Figure 2.6 shows a basic Seq2Seq architecture used for simplification.

RNN

The main advantage of using RNN is modeling sequences (sentences) of any number of symbols (words) so that predicting each symbol dependents on the previous symbols (words). However, RNNs suffer from short-term memory problem due to vanishing gradients where gradients tend to slowly disappear during back propagation. If a gradient value becomes extremely small, the learning will stop as expressed in the Equation 5.3, where W_i represents the wight for the current hidden state, W_{i-1} is the weight for the previous hidden state, LR is the Learning rate, and G is the gradient.

$$W_i = W_{i-1} - LR * G \tag{5.3}$$

To overcome the impact of short-term memory, an improvement was made to RNN version and introduced a gate structure versions: *Long Short-Term Memory* (LSTM) and *Gated Recurrent Units* (GRU). GRU has two gates *reset* and *update*; while LSTM has three gates *input, output,* and *forget*. Because of the architecture and the number of

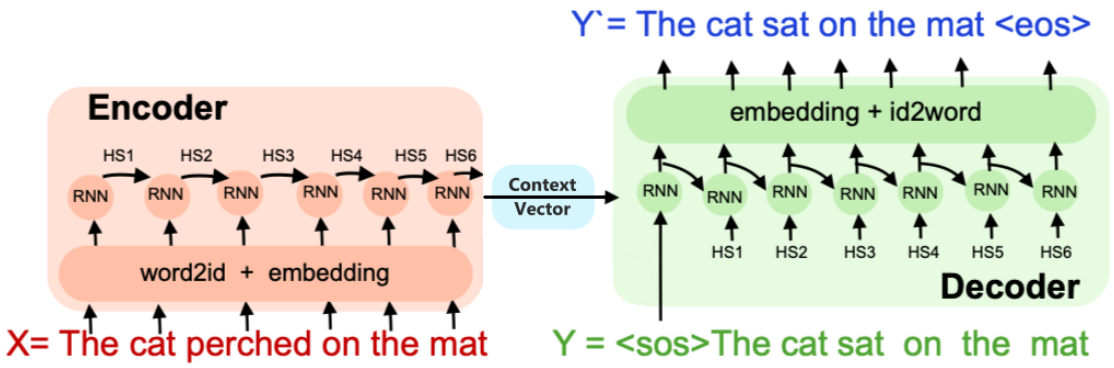

Figure 5.2: Encoder and Decoder in the Seq2Seq model

the gates, GRU has much simpler computation and implementation than LSTM. Besides, GRU consumes less memory units than LSTM, therefore, it trains faster. According to Yang et al. (2020), GRU is 29.29% faster than LSTM in training speed. Also according to the same study, GRU performance surpass LSTM using long text and small dataset. However, using other types of datasets, LSTM performance loss is much better than GRU. Using Seq2Seq model, the RNN hidden states used could be LSTM or GRU, depending on the task it is used for.

Encoder and Decoder

The Encoder takes a sequence (sentence) as an input and processes one token (word) at each time. Each RNN hidden state in the Encoder, influences the next hidden state . The Encoder consists of an Embedding layer, RNN layer (GRU was chosen for the dataset used), and a Linear layer. The objective of using an Encoder is to convert all the information in the input sequence into a vector of fixed length (**Context Vector**). After processing all the tokens and produce the Context Vector, the Encoder passes the vector onto the Decoder.

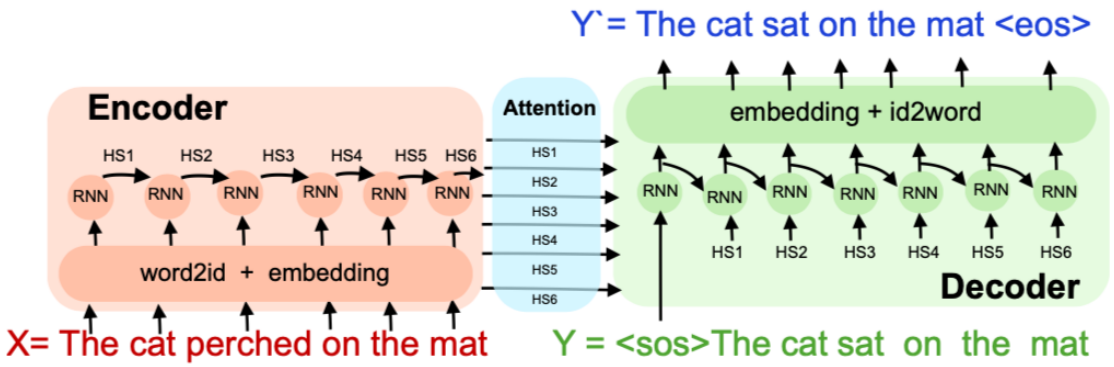

Figure 5.3: Encoder and Decoder in Seq2Seq model with Attention Layer

The Decoder reads the Encoder output (Context Vector) and tries to predict the target sequence token by token using the previous hidden state output. Similar to the Encoder, the Decoder consists of an Embedding layer, RNN layer (we used GRU units), and a Linear layer as shown in the Figure 5.2. The Embedding layers make a vocabulary table for the output and pass it into RNN layer (GRU layer in this case) to calculate the predicted output state. There are two functions used with embedding, **word2id** which is used in the Encoder unit to convert input sentences into vectors before sending them to RNN layer. The other function is **id2word** that we used in the Decoder unit to switch the produced vectors to words before outputting the predicted sentence.

Attention Layer

The predicted sequence produced by the Decoder is heavily influenced by the Context Vector formulated by the output of last RNN hidden state in the Encoder unit. This is a challenge that the model faces especially when dealing with long sentences. In processing long sequences, there is a high chance that the initial context has been lost by the end of the sequence. To solve this matter, a technique called Attention was introduced which allows

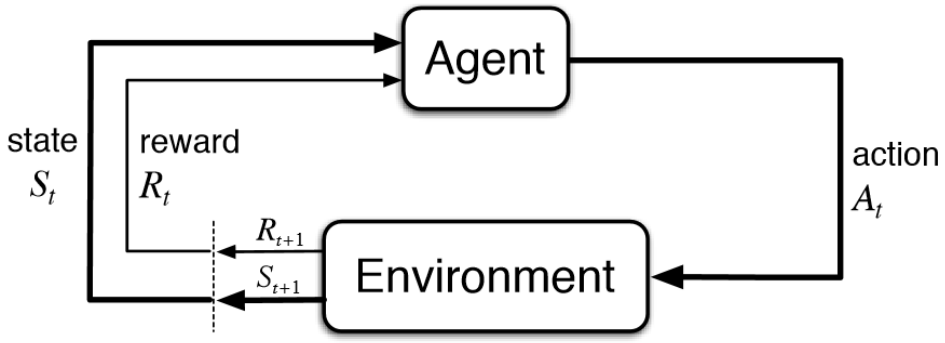

Figure 5.4: Basic Reinforcement Learning model

the model to focus on different parts of the input sequence at every stage of the output sequence (Luong et al., 2015; Galassi et al., 2020). This technique allows the Context Vector to be preserved from the beginning of the input sequence (original complex sentence) to the end as shown in Figure 5.3.

5.3.2 Reinforcement Learning

Reinforcement Learning (RL) is one of the most active research topics in the field of Artificial Intelligence and its popularity is growing. RL is a type of machine learning technique that enables an agent to learn in an interactive environment by trial and error using rewards earned from its own actions and experiences.

RL Characteristics

RL has two main components, the agent that learns and the environment that the agent deals with using an updated policy and a reward function (Sutton and Barto, 2018). RL does not provide a supervised training; however, it allows for a real number or reward

signal as a way to guide the training. During the training phase, sequential decision making (actions to take) is made in rounds (steps) by the agent, and the feedback from the environment is not instantaneous. The feedback is always delayed to the end of each step to be calculated as a reward for the agent. The agent's set of actions determine the subsequent reward it receives. The RL parameters values could affect the learning speed.

RL Approaches

There are several approaches used for RL: Model-Based, Value-based, and Policy-based approach. **Model-based** RL models the environment during training. Each environment needs a different model representation. It creates a model of the environment's behaviors without using previous knowledge. However, incorporating previous knowledge would speed up the learning pace (Arulkumaran et al., 2017).

For **Value-based** RL maximizes a value function V(s). The function provides the maximum expected future reward that the agent can get at each state. The value of each state is the accumulated rewards that the agent get. The agent chooses the state with the biggest value. One of the popular value-based methods is tabular *Q learning* and a discrete state representation (Zang et al., 2020), where the agent learns the value based on the action obtained. In text simplification, this approach is not convenient due to the large number of actions (number of words or vocabulary size).

The last approach, **Policy-based** RL, optimizes a given policy to obtain better outcomes. The agent learns the policy function that maps each state to the best action where $action = policy(state)$ The policy describes the agent's behavior in choosing actions and

gain maximum reward in the future. Finding the state could be *Deterministic* or *Stochastic*. The policy at a given state will always return the same action using a **Deterministic** state. While a **Stochastic** state gives a distribution probability over the actions (Ding et al., 2020). In our simplification model, we will use the Policy-based approach. The policy will be updated after each sentence simplification step, which produces a set of actions.

RL Performance

There are several components needed for a RL system. The agent that learns, the environment which the agent deals with, the policy that the agent follows to choose actions, and the reward signal that the agent observes upon taking actions. The agent chooses actions based on the policy found in the given environment. Then the agent gets a reward value depending on how good is the observation gained through the action chosen. The better actions chosen, the better observation results produced, leading to a higher reward value given.

To start the RL loop, we could use Policy-based approach or Value-based approach. Using Value-based approach, we build a Q-table with n columns and m rows, where n is the number of actions and m is the number of states. Then we initialised the values of the table cells with zero. The key to run the reinforcement learning algorithm is to assign the right values to the Q-table using the function shown in Equation 5.4 to update all the cells in the table. Every time we observe an outcome sentence and a reward value, we need to update the Q-table with $Q^{new}(s_t, a_t)$ value using a special reinforcement learning formula.

This formula is expressed in details through Equation 5.4. The RL algorithm aims to maximize the reward signal given to the agent at every step during training stage. The agent chooses the actions that influence the environment to produce a better and higher rewards.

$$\boldsymbol{Q^{new}(s_t, a_t) = Q(s_t, a_t) + \alpha.[R(s_t, a_t) + \gamma.maxQ(s_{t+1}, a) - Q(s_t, a_t)]} \quad (5.4)$$

where:

$Q^{new}(s_t, a_t)$: New Q value for the state s_t and action a_t

$Q(s_t, a_t)$: Current Q value for the state s_t and action a_t

α : Learning rate

$R(s_t, a_t)$: Reward for taking an action a_t at state s_t

γ : Discount Rate

$maxQ(s_{t+1}, a)$: Maximum expected reward for state s_{t+1} and all possible actions a

For text simplification, we use the Policy based approach. The approach uses a tuned policy P_t, an updated status S_t, and a reward R_t to improve the output of the model where $R_t = P_t(S_t)$. The status S_t starts with zero as an initial value for the loop. Then it updates this value according to the input X and the previous action y'_{t-1} where $Policy(y'_t|y'_{t-1}, X)$. Every chosen action $y_t \in V$ represents a word, where V is the vocabulary dictionary. We use a pretrained Seq2Seq model with Attention Layer model (S2SA) as our main component in the RL loop to enhance the performance of the predicted simplified sentences. We also use our readability level classifier to decide the reward deserved by the agent. The actions in this model would the words to choose from the vocabulary dictionary we built.

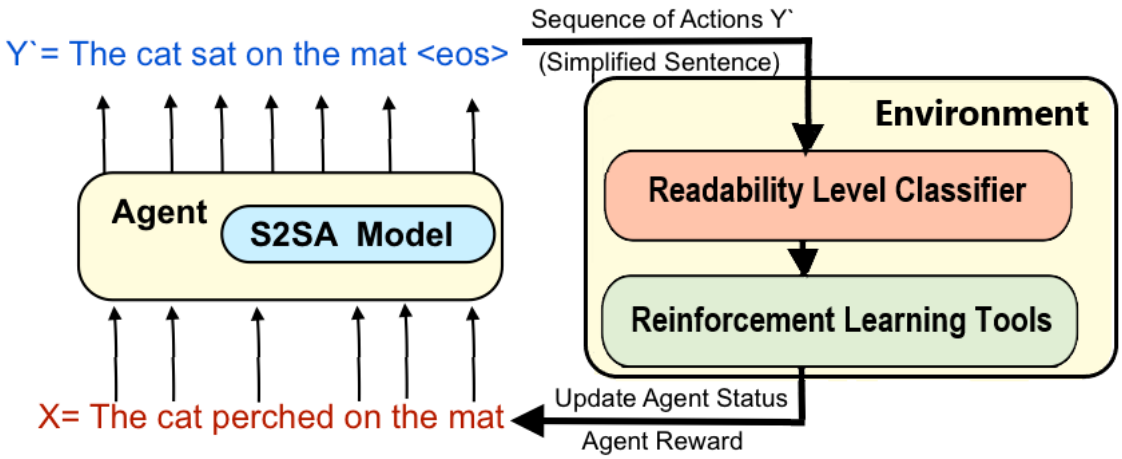

Figure 5.5: Simple illustration for S2SA model with Reinforcement Learning for simplification

The agent continues to take actions (choose words) until it produces the $< eos >$ token. Then the list of chosen actions (words forming the sentence) will be evaluated for calculating the reward and updating the status. Figure 5.5 shows a simple illustration of our RL model using S2SA and the readability classifier, with an example.

On the other hand, using Policy-based approach, we setup a policy P and update its value after every step using the current status of the environment S. The action A is calculated afterwards based on the policy P and the environment S values $A_t = P_t(S_t)$. After performing the action, the reward R is calculated based on a predefined reward function. After performing several RL loops, the P value in the policy function will be tuned to produce a higher reward.

5.4 Evaluation Method

The best way to determine the quality level of a simplified text is through human evaluation. A simplified text should not be evaluated based only on criteria such as shorter sentences and shorter words (Lasecki et al., 2015). The evaluation should take into consideration the fluency in grammar, meaning preservation, and the readability of the sentences. But there is a need for automatic measures. There are several metrics that were proposed for evaluating text simplification models, like SARI, BLEU, and SAMSA. SAMSA (Sulem et al., 2018a) is a metric developed to measure the simplicity of the sentence's structure like sentence splitting. However, it has not been used popularly in papers like SARI and BLEU.

5.4.1 BLEU

BLEU (Papineni et al., 2002) is a metric proposed initially for Machine Translation. It is designed to evaluate bilingual translation systems but also popularly used on monolingual tasks like text simplification or summarisation. This popularity is justified in (Štajner et al., 2014) where it showed that BLEU correlates with human judgments of grammatical and meaning preservation.

However, this metric is not adequate for all text generation problems (Zhu et al., 2010). It is not well suited for assessing and evaluating simplicity from neither a lexical (Xu et al., 2016) nor a structural (Sulem et al., 2018b) point of view. BLUE is not recommended by Sulem et al. (2018b) to be used for text simplification. They argue that BLEU scores do not reflect grammar or overall meaning very well. However, it is still widely used in evaluating

the simplification models in the papers we compare with, therefore we will include it in our results.

5.4.2 SARI

While BLEU struggles to reflect meaning and grammar correctly, SARI is a measure strongly correlated to the simplicity level. SARI (Xu et al., 2016) is a lexical simplification metric that measures the efficiency of the words added, deleted, or kept by a simplification model. The metric compares the output text from multiple simplification models with the original input sentences.

SARI focuses on lexical text simplification. Even though the performance of SARI is slightly slower than BLEU (Xu et al., 2016), it is the main metric used for evaluating text simplification models. Although SARI is able to measure and assess the quality of lexical simplification, it fails to correlate with human judgments when structural simplification is performed by the evaluated systems. SARI could get the measure of text paraphrases. However, text simplification sometimes involves text transformation, which is beyond paraphrasing.

5.4.3 SAMSA

SAMSA (Sulem et al., 2018a) is a simplification metric that measures structural simplicity that includes sentence splitting. It uses semantic parsing to assess the simplification quality by decomposing the input based on its semantic structure and comparing it to the output. Due to the vast space of valid simplifications for a given sentence, SAMSA provides a

reference-less automatic assessment to avoid the problems that *reference-based* methods face.

The SAMSA metric is based on the idea that an optimal split of the input is where each possible structure is assigned to its own sentence in the simplified output. Then it measures to what extent this assertion holds for the input-output pair under consideration (Niklaus et al., 2019). As a result, SAMSA score would be maximized when each split sentence represents exactly one semantic unit in the input. SAMSA is highly correlated with structural simplicity and grammar. It measures the preservation of the sentence level of semantics as well as the structural simplicity. This implies that the output sentences contained in the corpus are grammatically correct and present a simpler syntax than the input (Niklaus et al., 2019).

However, since our work deals with simplification on sentence level that does not include sentence splitting, we cannot use SAMSA as a metric to assess our work results later in this thesis. SAMSA would be more appropriate to evaluate paragraphs and documents that includes sentence splitting. Moreover, SAMSA is rarely used to evaluate the current simplification models. SARI and BLEU are more common methods to evaluate the simplification models, even though they are not initially created for that purpose.

After explaining the methodology of the classification and simplification models in this chapter, we are ready to present the work of our readability system. In the next chapter, Text Classification and Data Augmentation using Readability Levels, we will show the text classification setup, followed by the result and analysis. Then we will present the sentence classification used for augmenting a larger dataset through Wikipedia Corpus

and Mechanical Turk.

Chapter 6

Text Classification and Data Augmentation using Readability Levels

In this chapter, we present our text classification experiments. We outline the main features of our text classifiers and analyze the results. The classes are the readability of the text, and we classify at documents level and sentence level. Later in this chapter, data augmentation is applied to enrich the simplification dataset.

6.1 Overview

We apply Text Classification (TC) on Newsela dataset using classical machine learning methods (**SVM** and **Random Forest**) and deep learning method (**CNN**). Since Newsela

labels documents with one of five categories, the TC models classify documents into these five categories, based on their readability level learned from Newsela dataset (from which we used a part as training data, and left aside a part as test data, as explained below). First, we extract different features from the dataset then use several models to classify the documents and compare their results. The classification uses Python and tools provided in NLTK[1], NumPy[2], Scikit-learn[3], SpaCy[4], and Keras[5] libraries to build the models. Later, we modify the model features to handle classification on sentence level for simplification. Using this modified version of TC model, we produce and augment (larger) text simplification dataset using the Wikipedia Corpus and the Mechanical Turk Corpus presented in Chapter 4.

6.2 Text Classification Setup

6.2.1 Data Preparation and Preprocessing

Data Preparation

The dataset is provided by Newsela [6]. It has 10,789 documents, 9,624 documents written in English and 1,165 documents written in Spanish. Since the possible features extracted from both languages are not exactly the same, we decided to work only on the English documents.

[1] https://www.nltk.org
[2] https://numpy.org
[3] https://scikit-learn.org
[4] https://spacy.io
[5] https://keras.io
[6] https://newsela.com/data/

Readability	Number of English documents	Targeted Students Grade
Level 0	1,967 documents	Covers mostly grade **10** to **12**
Level 1	1,910 documents	Covers mostly grade **8** to **9**
Level 2	1,910 documents	Covers mostly grade **6** to **7**
Level 3	1,910 documents	Covers mostly grade **4** to **5**
Level 4	1,882 documents	Covers mostly grade **2** to **3**
Level 5	42 documents	Covers grade **2**

Table 6.1: Unbalanced distribution of documents in Newsela Corpus

The English documents are initially categorized into six categories based on their readability level. The categorization were assigned manually by human experts as stated in Newsela website. Table 6.1 shows the distribution of the simplified English documents which is unbalanced due to the *level 5* category. This unbalanced data could negatively affects the results. Since *Level 5* has only 42 documents, and targets the same grade as *Level 4*, we decided to combine *Level 5* documents with *Level 4* documents under *Level 4* category. This will result in having a more balanced distribution of the data with 1,924 documents under *Level 4* category without mislabeling any document.

Data Preprocessing

To extract features from each document and perform classification, the data must be preprocessed. We substituted multiple newlines with a single newline in each document. Furthermore, multiple spaces and tabs were replaced with a single space character. We also removed non-ASCII codes, text between the tags "<" and ">" , special characters (like $ and #), single characters, and numbers. Then we converted all letters to lowercase letters. Finally, we extracted the features from the cleaned data.

There are many types of features that interest us in this experiment: basic features, syntactic features, and common readability metrics. *Basic features* include words and textual information of the text, like simple counts and frequencies. On the other hand, *syntactic features*, require more effort. We need to use linguistic tools and external resources in order to compute these features. Also, popular *readability metrics* could be used as features. We start with extracting **basic features**. We count the number of paragraphs, average number of sentences at each paragraph, average number of words at each sentence, number of words, average length of words, and average number of syllables for *content words*, or non-stop words.

For **syntactic features**, we use NLTK library to find syntactic characteristics like sentences and words in the document. *Syntactic categories* could include passive voice sentences, adverbial phrases or clauses. Also, it includes nouns, adjectives, verbs, adverbs, pronouns, conjunctions, and even negations in sentences. These different syntactic categories are extracted using syntax tree (parse tree) for each sentence. We then calculate the percentage of each type of syntactic categories in the text. For every type, one or more values is calculated and used as a feature. For example, we find the percentage of the passive voice sentences among all sentences. On the other hand, two values are calculated, like computing the average number of nouns in a sentence and the percentage of nouns among all words found in the document.

Finally, we add popular **readability metrics** as another type of features, in particular the *Gunning Fog* Index and the *Flesch-Kincaid* Grade Level. Using the Python package `textstat`, we calculate the values of these readability metrics, for both Gunning Fog Index

and Flesch-Kincaid Grade. We will see how effective the performance of our model is with and without using the readability metrics as part of our features. Now, we have a dataset of 9,629 documents ready to be classified using about 30 features from three different types.

The dataset is classified several times. Each time, we use different set of features to perform the classification. The features of interest are divided into three groups. These groups are *Flesch-Kincaid Metric* (**FK**), *Gunning Fog Metric* (**GF**), and the rest of the *Extracted Features* (**EF**) we previously extracted using both basic and syntactic features. First, we will be using a dataset that holds only one group of features for classification, e.g., FK or EF. Then, the dataset will cover two or more groups of features, e.g. EF with FK, and so on. This will show us which group of features gives use the best classification results.

We choose to split our data into 80% for training (10% of this training set is used for validation) and 20% for testing. We will report the results by 10-fold cross validation on the training data and the results on the test data as well. Also, the cross validation will help us choose the proper parameters of our models. This will allow us to ensure that the models are not overfitted on the training data, so that they also work well on the test data.

6.2.2 Classification Implementation

For classification, we start with classical machine learning classifiers as an initial work. Then we compare their results with the results of a deep learning model. To classify the documents into five categories (0 to 4), we choose the Random Forest and the SVM classifiers as our basic machine learning models, as mentioned before, because both classifiers

Accuracy	FK	GF	EF	EF+FK	EF+GF	EF+FK+GF
Training						
SVM	61.34%	65.30%	70.17%	72.66%	**73.39%**	73.06%
Random Forest	61.44%	64.87%	72.98%	74.46%	75.56%	**75.76%**
Test						
SVM	62.88%	66.77%	65.57%	67.96%	68.22%	**68.33%**
Random Forest	61.06%	64.69%	74.30%	75.18%	75.75%	**75.80%**
FK: Flesch-Kincaid, **GF**: Gunning Fog Metric, and **EF**: Extracted Features						

Table 6.2: Classification accuracy using the Random Forest and SVM classifiers

RMSE	FK	GF	EF	EF+FK	EF+GF	EF+FK+GF
Training						
SVM	0.68	0.63	0.50	0.44	**0.43**	**0.43**
Random Forest	0.68	0.61	0.51	0.48	**0.46**	**0.46**
Test						
SVM	0.67	0.62	0.63	0.57	**0.56**	**0.56**
Random Forest	0.66	0.64	0.55	0.54	0.52	**0.51**
FK: Flesch-Kincaid, **GF**: Gunning Fog Metric, and **EF**: Extracted Features						

Table 6.3: RMSE values using the Random Forest and SVM classifiers.

are suitable for classifying a set of data with multiple classes, as stated in (Nayak et al., 2013). This study also showed that using the SVM classifier with a 'rbf' kernel performs well with multiple labels and a small number of features. Therefore, for the SVM classifier, we set the kernel value to 'rbf'. Then we ran a grid search to tune the values of the two main parameters, gamma and c. After applying the grid search, we found that the best results are obtained when the gamma and c values are set to 0.001 and 100, respectively.

In the Random Forest classifier, we set the maximum depth to 8 to allow pruning and avoid overfitting. For both classifiers, SVM and Random Forest, we applied *10-fold cross*

validation on the training data, then found the mean value of the accuracies. The results in Table 6.2 show the mean accuracy values after applying 10-fold cross validation. Since in our case, it matters if the classifiers makes errors by confusing classes that are close to each other or far apart, we calculated the *Root Mean Squared Error* (RMSE) as an additional evaluation measure. RMSE is usualy employed for numeric outputs (continues), but we still want to check if the confusing is between distant classes or close by classes. That is why we used RMSE and the scale of this measurement is from 0 to 1. Table 6.3 shows the RMSE mean values after applying 10-fold cross validation. The table also shows the results on the test data.

For the deep learning part, we built a simple CNN model since it was proven to work well in text classification tasks as shown in (Amin and Nadeem, 2018). The model consists three convolution layers, two pooling layers, one flat layer, and one dense layer at the end. The model has a total of 206,981 parameters. For the convolution layers, we used kernel size between 3 and 5 and set the activation to `'relu'`. In the last layer, the dense layer, we used `'softmax'` for the activation and specified the number of output categories to five (0 to 4). To compile the model, we chose `'adam'` as the value for the optimizer and used `'sparse_categorical_crossentropy'` for the loss function. To fit the model with training data, we used fit function with batch size equals to 10 and epochs equals to 30. We used 10% of the training data as validation data[7], and trained our model on the 90% data that remained.

[7]Choosing 20% for validation data resulted in an overfitted model with two points decrease in the accuracy value and two points increase in the RMSE value.

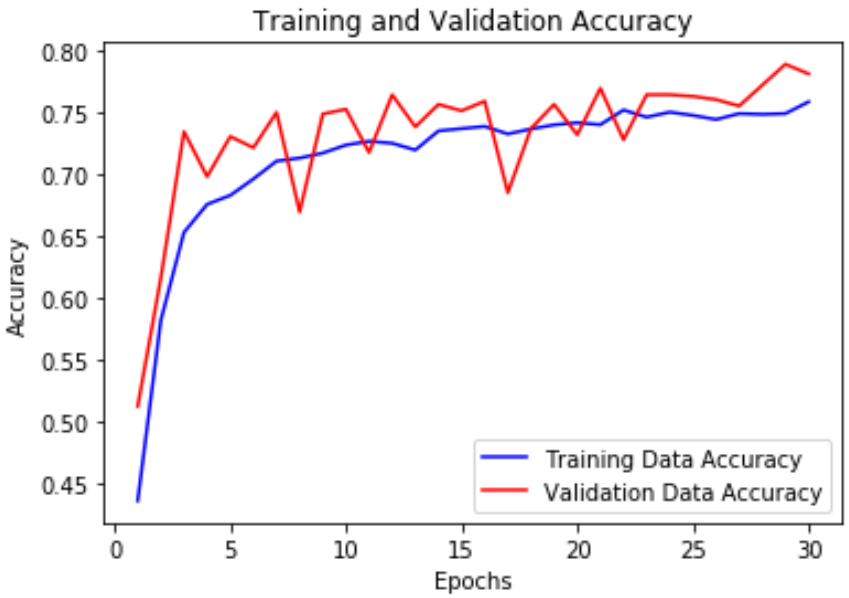

Figure 6.1: The accuracy values for the CNN classifier

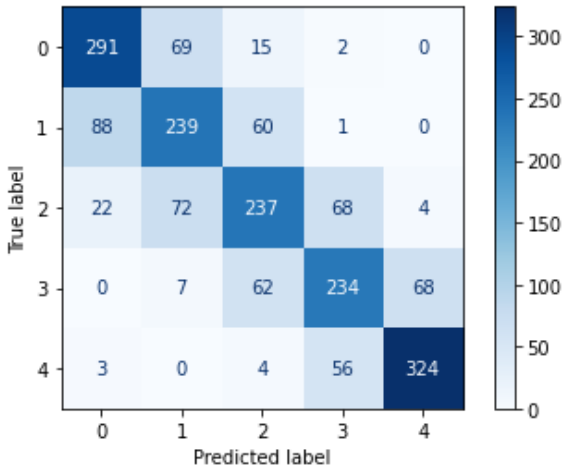

Figure 6.2: The Confusion Matrix for the test data using the SVM classifier

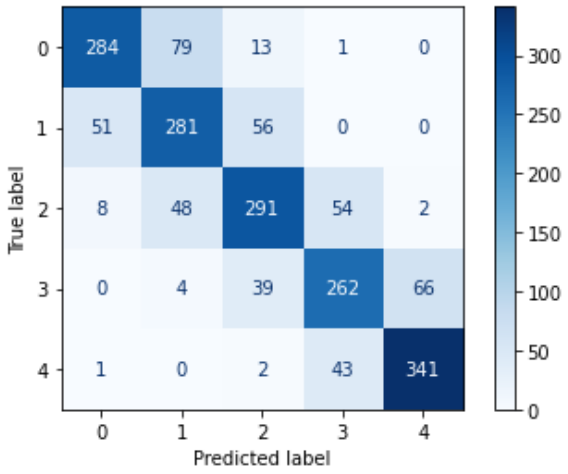

Figure 6.3: The Confusion Matrix for the test data using the Random Forest classifier

6.3 Results and Discussion

6.3.1 Classical Machine Learning Models Performance

With five categories and working with all groups of features EF, FK, and GF to classify the data, Random Forest gives the best results with accuracy above 75%. SVM's accuracies are above 68%, for cross validation on the training data and and for the test data. The confusion matrix for SVM and Random Forest are shown in Figure 6.2 and Figure 6.3. Also, using all groups of features EF, FK, and GF gave the lowest RMSE value. Table 6.2 and Table 6.3 summarize the accuracy and the RMSE results of classifying the data with different groups of features using the two classifiers, SVM and Random Forest. Please note that the baseline, for a dummy classifier (random choice), is 20% (not shown in the tables).

Training the Random Forest classifier and applying it on test data was time efficient. The overfitting issue was solved by pruning the trees, and choosing a shorter depth for the

model. On the other hand, the SVM model used less memory when performing classification, but more time for training and then for classifying the test data. However, since our dataset is not very large, 9,629 documents, the running time was reasonable.

To evaluate the two models, we found the accuracy mean value of our results after applying 10-fold cross validation. The results provided in Table 6.2 show that Random Forest classifier gives a better accuracy values than the SVM classifier on both the training and the test data. This is clear in the last column using all features with the test data, Random Forest classifier gives 75% accuracy compared with the SVM classifier that gives only 68%. Looking at the classifiers results from a different evaluation angle, we considered *RMSE* for both machine learning models. We found the RMSE mean value of our results after applying 10-fold cross validation for training and test data. As shown in the last column in Table 6.3, when training the models using all groups of features, the SVM classifier gives a better (lower) RMSE value (0.43) than Random Forest classifier (0.46). However, with test data, Random Forest has the better RMSE value (0.51) compared with a higher value for SVM (0.56).

The difference in the classification results is clearly visible when using certain types of features. When using only one group of features, KF, GF, or EF, we see that FK metric gives the lowest accuracy results and the highest RMSE values using either SVM or Random Forest. The GF metric results are slightly better than FK, but they do not reach the results obtained from using EF. Using the EF, which includes using all text readability features except readability metrics, almost outperforms the famous readability metrics. We tried combining EF, since it gives the best overall performance alone, with

Classifier Model	RMSE	Accuracy
Training Data		
CNN	0.50	**77.67%**
SVM	**0.43**	73.06%
Random Forest	0.46	75.76%
Test Data		
CNN	**0.51**	**76.90%**
SVM	0.57	68.33%
Random Forest	**0.51**	75.80%

Table 6.4: Comparing the three classifiers based on RMSE and Accuracy

other readability metrics expecting to have an improvement in the results. Combining EF with GF gives an improvement in the results that is slightly better than combining EF with FK. Add to that, combining FK with other features sometimes does not add any improvements to the results.

6.3.2 CNN Performance

On the other hand, using CNN to classify the documents gives competing results with accuracy above 75% and RMSE around 0.50, as shown in Table 6.4. We first trained the classifier over the training data, while keeping 10% as validation data. The classifier ran for 30 epochs. At every epoch, we trained on 90% of the training data and used the 10% left for validation. The accuracies on both the validation data and the training data are shown in Figure 6.1. We stopped the training after 30 epochs because the performance after that started to decrease. After training the CNN classifier, we apply it on our test data. We show the *confusion matrix* for the test data in Figure 6.4.

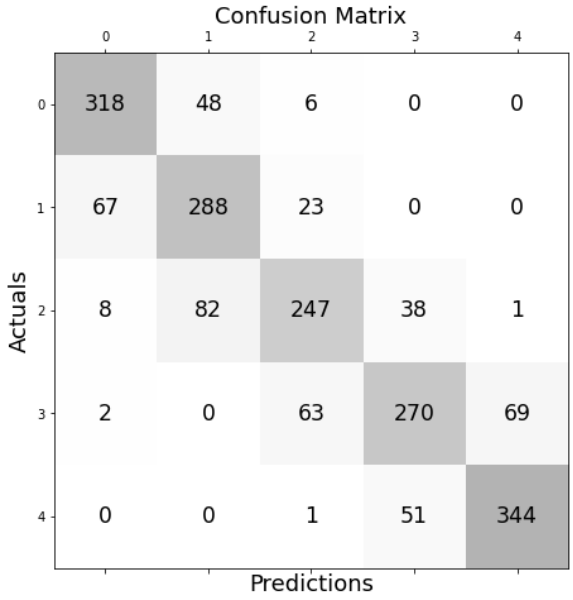

Figure 6.4: The Confusion Matrix for the test data using the CNN classifier

Comparing all three classifiers, as shown in Table 6.4, we found that the CNN classifier outperforms the SVM classifier in terms of accuracy. Also, CNN gives better accuracy values compared with Random Forest, on both the training data (using cross-validation) and the test data. However, on the training data, SVM takes the lead with RMSE equals 0.43. But on the test data, the CNN and Random Forest classifiers give a better RMSE values than the SVM classifier. The results are shown in Table 6.4.

6.3.3 Error Analysis of the Results

To explain the low accuracy results of the classifiers, we checked the confusion matrix for the test data when using CNN classifier, shown in Figure 6.4. We see a small percentage of documents mistakenly assigned to adjacent classes. Then we checked the original distribution of the documents among the readability classes, shown in Table 6.5. We see a clear

Grade	12	11	10	9	8	7	6	5	4	3	2
Level 0	1853	2	16	25	14	-	1	-	-	-	-
Level 1	-	-	4	725	980	158	39	4	-	-	-
Level 2	-	-	-	-	43	1050	726	86	5	-	-
Level 3	-	-	-	-	-	2	292	1224	386	6	-
Level 4	-	-	-	-	-	-	-	28	1178	494	224

Table 6.5: Distribution of grades documents among the readability levels

overlapping between adjacent grades documents at every readability level. This caused the low accuracy results and high RMSE values in our classifiers.

In the dataset we are using, the Newsela Corpus, each readability level folder has a number of documents that belong to certain grades. Documents of a particular grade can be assigned to more than one readability level. However, most of the documents, are assigned to a certain level, as shown in Table 6.5. The distribution of grades documents and the confusion matrix obtained have almost the same percentages and follow the same curve. The assignment of the grades documents were done manual by professionals, as mentioned in the Newsela corpus description. The manual documents assignment did not prevent overlapping between classes, causing a distribution similarity between the confusion matrix in Figure 6.4 and Table 6.5.

Due to the overlap between adjacent grades, there are few documents assigned that are not enough to decide their readability levels. The total number of documents is 9,629 as shown in Table 6.5. By choosing the groups with enough documents for training, the total number of documents is 8,372 documents distributed as in Table 6.6. We apply the

Grade	12	11	10	9	8	7	6	5	4	3	2
Level 0	1853	2	16	25	14	-	1	-	-	-	-
Level 1	-	-	4	725	980	158	39	4	-	-	-
Level 2	-	-	-	-	43	1050	726	86	5	-	-
Level 3	-	-	-	-	-	2	292	1224	386	6	-
Level 4	-	-	-	-	-	-	-	28	1178	494	224

Table 6.6: Distribution of Filtered grades documents among the readability levels

Classifier Model	RMSE	Accuracy
Training (xval)		
CNN	0.46	**83.34%**
SVM	0.35	79.18%
Random Forest	0.39	81.81%
Test		
CNN	0.48	**82.01%**
SVM	0.51	75.52%
Random Forest	0.48	81.59%

Table 6.7: Comparing classifiers results using Filtered grades documents

classifiers on this new filtered dataset and we found improved results, as shown in Table 6.7. However, we always get better results using CNN.

6.4 Sentence Classification and Data Augmentation

Using simplification and data augmentation on document level was not sufficient in our case. That is because of two reasons. There are limited datasets available that provides simplified document, Newsela is one of them. Besides, using a document length as an input

Classifier Model	RMSE	Accuracy
Training (xval)		
CNN	0.49	**85.69%**
SVM	0.53	81.02%
Random Forest	0.48	85.64%
Test		
CNN	0.49	**85.52%**
SVM	0.53	80.68%
Random Forest	0.48	85.48%

Table 6.8: Comparing classifiers results using aligned sentences

sequence costs a lot of memory during training. Therefore, we focus on classification and simplification at sentence level. We modified our classifiers to work with *complete sentences* and extract the features needed for classifying the text. We removed the paragraph features, i.e., the length of a paragraph in terms of number of sentences and number of words, the average number of passive voice sentences in a paragraph, etc. We also used sentence alignment for Newsela dataset found in (Jiang et al., 2020) which applies a neural CRF model for sentence alignment. We obtained 464,555 aligned sentences in different readability levels. We excluded any alignment that has less than thee words, since the minimum number of words needed to form a proper *complete sentence* is set to three words in our experiments. The alignment provides complex sentence id, complex sentence text, simple sentence id, and simple sentence text. The id of those sentences consist the level of the document it was taken from, the document name, the paragraph number, and the sentence number. Using these aligned sentences, we trained sentence classifiers to find the best classification model. We split the aligned sentences into 80% for training and 20% for test.

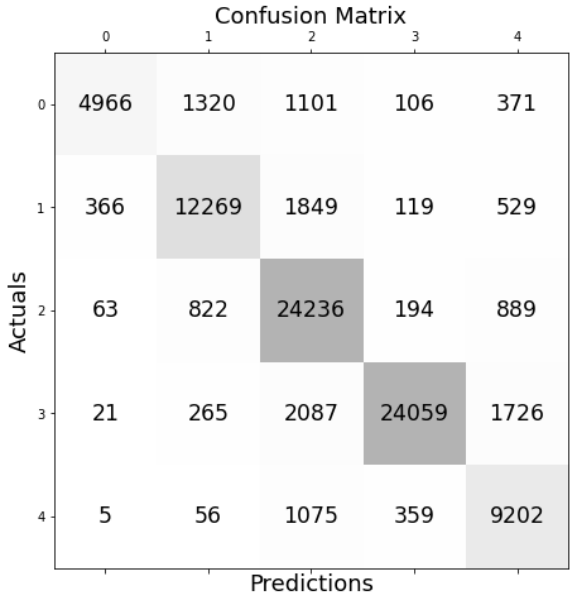

Figure 6.5: Confusion matrix for test data using CNN classifier and aligned sentences

The Accuracy and RMSE scores for the sentence classification models are shown in Table 6.8, with the CNN model having the highest accuracy result, as expected from our document classification experiment. Figure 6.5 shows the Confusion Matrix for test data using CNN classifier over aligned sentences. The alignment technique used for aligning Newsela sentences in (Jiang et al., 2020) includes sentence splitting, merging, and paraphrasing with deletion, which resulted in more meaningful sentences.

Using the CNN sentence classifier discussed above, we were able to automatically classify more simplified sentences based of their readability level. We used the datasets found in Wikipedia Corpus and Mechanical Turk, mentioned in Chapter4, to find the sentences matching readability level. We excluded all sentences with non-English words. We also removed sentences that are too short (less than 3 words length) to form complete sentences. Also, complex sentences simplified to more than one sentence (split sentences) were ex-

cluded since we are not dealing with split sentences. This allowed us to produce a larger dataset with five readability levels that can be used in text simplification. We augmented our initial entence simplification dataset, extracted from (Newsela) with 238,019 pairs of sentences divided into 4 categories (to level 1, to level 2, to level 3, and to level 4). The augmented simplification dataset has 702,574 pairs of sentences that can be used in future text simplification researches.

6.5 Conclusion

The purpose of our classifiers is to classify documents based on five readability levels. Using the Newsela Corpus as our dataset, we adopted its five readability levels, level 0 (difficult text) to level 4 (simple text). Then we extracted several features from the dataset: basic feature, syntactic features, and readability metrics. We built different classification models, SVM and Random Forest as basic machine learning models, and CNN as the deep learning model. Each model was applied over the dataset several times using different types of features each time. After comparing the three models performance, we see that CNN has better Accuracy and RMSE scores.

We aligned the sentences in Newsela Corpus, in order to be able to perform classification on sentence level. To handle sentence classification, we modified our classifiers' features. Then we applied these modified versions (sentence classification models) over the aligned sentences from Newsela. Like for the document classification performance, the CNN model also got the best scores for sentence classification. Using this modified version of the sentence classification models, we were able to classify more documents from the Wikipedia

Corpus and the Mechanical Turk Corpus, at sentence level. By doing this, we built an augmented dataset of aligned sentences, and we will provide more training data classified based on readability level for our text simplification model. Part of this work was published in (Alkaldi and Inkpen, 2021).

The next chapter, Text Simplification, will explain our text simplification system with and without reinforcement learning. It will present, compare, and analyse several sentence simplification results using aligned sentences from the Newsela dataset. Then, it will apply the simplification on the augmented dataset produced by our classification model and discuss the results.

Chapter 7

Text Simplification

This chapter presents our text simplification system with and without the use of reinforcement learning technique. We designed several sentence simplification models using the aligned sentences from the Newsela dataset and then adding the augmented dataset produced by our sentence classification model from the previous chapter. At the end of the chapter, the results are presented and analysed.

7.1 Overview

We apply Text Simplification (TS) over aligned sentences from Newsela dataset using deep learning methods. The aligned dataset was described in Section 6.4 along with its classification results. We built a simplification model using a seq2seq model with an attention layer (S2SA). Then we enhanced our model by applying Reinforcement Learning with rewards based on the readability level of the predicted sentences (S2SARL). The readability

levels of those sentences are calculated using our classification model presented in Chapter 6. Then using the aligned sentences, we apply the simplification models S2SA, S2SARL, and EditNTS (from related work), to compare their results. After that, we apply the simplification models over the dataset augmented as described in Section 6.4.

7.2 Text Simplification Framework

7.2.1 Data Preparation and Preprocessing

The dataset used for simplification consists in the aligned sentences from Newsela corpus. We used sentences from the Newsela dataset found in (Jiang et al., 2020) which uses a neural CRF model for aligning the sentennces. The paper provides a set of records, each record contains the simple sentence's label, the simple sentence's text, the complex sentence's label, and the complex sentence's text. Each sentence label provides the name of the article, its level, the paragraph number, and the sentence number where it was taken from. The aligned sentences are categorized into five groups based on the document where output sentence is taken from. The readability label of the document folder determines the category of the sentence. The aligned pairs of sentences are labeled with the readability level of the target sentence. We excluded pairs that had non-English words or consisted in less than three words in a sentence (not a proper *complete sentences*) and obtained 464,555 pairs of Newsela Aligned Sentences (hereafter, the **NAS** dataset).

We also used the classified sentences found in Section 6.4. Using the trained sentences classifier against Wikipedia Corpus and Mechanical Turk Corpus, we produced 238,019

pairs of automatically Classified Simplified Sentences (hereafter **CSS**). We used CSS to augment the NAS dataset and obtained 702,574 pairs of sentences as our Augmented Simplification Dataset (hereafter **ASD**), in order to be able to provide more training data for our models. All three datasets (NAS, CSS, and ASD) are divided into four categories (level 1 to level 4) based on the readability level of the target sentences (simple sentences).

For every category, we split the datasets into 90% for training (10% of it for validation) and 10% for testing.

7.2.2 Setup Simplification Models

Setup Seq2seq Model with Attention (S2SA)

To begin the simplification task, we start with a sequence-to-sequence model (seq2seq) that takes a sequence of items (words) as input and gives back another sequence of items as output. The basic Seq2Seq model consists in two RNNs, an Encoder, and a Decoder, as proposed in (Sojasingarayar, 2020). Our Seq2Seq model uses GRU as RNN units, since GRU has much simpler architecture and implementation than LSTM. GRU needs less memory units than LSTM; therefore, it trains faster. Besides, according to Yang et al. (2020), when using long text and a small dataset, GRU performance's surpassed that of LSTM. Therefore, using GRU is more appropriate for our work.

The model consists in two main components, an encoder and a decoder. Both components have an embedding layer with 256 dimensions, GRU units with dropout equals to 10%, and a linear layer as the last layer to pass the output through. The hidden states for the encoder and the decoder are 256 states. The last hidden state in the encoder is

actually the *Context Vector* we pass to the decoder. The *Context Vector* in this model acts as a bottleneck. It makes it challenging to deal with long sentences without forgetting the source input. Therefore, to enhance our seq2seq model performance, we added an *attention layer* (Luong et al., 2015) to the decoder unit. With the attention layer, our seq2seq model can deal with all sentences with any length without forgetting the source input. The attention layer contains two linear layers with 256 hidden dimensions. This allows the decoder to know where to focus to predict better outputs.

Setup Seq2seq with Reinforcement Learning (S2SARL)

To further boost our simplification model results, we used a reinforcement learning (RL) loop with our pretrained S2SA model because (**S2SARL**) is the state-of the-art technology in deep learning for text simplification. Figure 5.4 shows a simple illustration of our RL model with an example. RL is a machine learning technique that enables an agent to learn in an interactive environment by trial and error using rewards earned from its own actions (Sutton and Barto, 2018). The main components of a RL system are the environment and the agent. We start the model by creating the vocabulary dictionary table using the words found in the dataset with frequency of two or more. Then for the agent, we set up our S2SA model introduced earlier in Section 7.2.2 to produce set of actions (words) using the dictionary table created. We initialized the *reward*, *status*, *total loss*, and the vocabulary dictionary table to zeros. Then we built a *step* function that uses the environment tools to perform a simplification for a given sentence (sequence of input words). After performing every *step*, the agent updates the *reward*, *status*, *loss*, and the vocabulary dictionary values

with new values based on the predicted simplified sentence (sequence of actions).

To prepare the environment, we set up the $Target$ Level number (1 to 4) and prepared tools to help the agent during training like: observe current status, get all possible outputs for an action (predicted word), and give appropriate rewards based on a set of chosen actions (predicted simplified sentence). The environment has several tools that helps during training. The environment current state could be observed based on the output it produces using *get-observe* which we built in the environment. Also, we wrote *get-actions* as a function to give all the outputs possible for a given action. Then the appropriate reward is given by *reward* based on the chosen actions. The reward value R_t is determined by the readability level of the predicted sentence ($PrdS$). For every ($PrdS_t$), we use the sentence readability level classifier ($Rclf$) described in Section 6.4 to classify the $PrdS_t$ sentence into its readability level. Then we calculate the reward R_t as follows:

$$R_t = \begin{cases} -0.5 & \text{if } Rclf(PrdS_t) < Target \\ +2.0 & \text{if } Rclf(PrdS_t) == Target \\ +1.0 & \text{if } Rclf(PrdS_t) > Target \end{cases}$$

The reward function shows if the predicted sentence readability level is less than the *Targeted Level* that we are trying to reach, the environment gives -0.5 as a penalty. This encourages the agent to predict simpler sentence in the next *step*. If the predicted sentence readability level matches the *Targeted Level*, then the environment gives back $+2.0$ as a reward to encourage the agent to keep this level of simplicity. However, if the predicted sentence was too simple, i.e., the readability level is more than the *Targeted Level* as a reward. Penalizing the agent with negative rewards for exceeding the *Targeted Level* did

Dataset	Model	SARI	BLEU
To Level 1	EditNTS	26.48	65.23
5,129 pairs	S2SA	**31.76**	65.61
	S2SARL	31.57	**70.22**
To Level 2	EditNTS	20.62	46.81
9,780 pairs	S2SA	27.18	53.95
	S2SARL	**31.56**	**60.53**
To Level 3	EditNTS	20.26	33.28
13,922 pairs	S2SA	30.83	45.24
	S2SARL	**32.27**	**53.85**
To Level 4	EditNTS	23.21	23.97
17,626 pairs	S2SA	31.69	42.60
	S2SARL	**32.42**	**50.97**

Table 7.1: Test scores for simplification models trained on NAS using NAS test data

not improve the output. Yet giving a smaller reward like +1.0, improved the results. We also used *is-done* as a way to determine if there are any sentences left for simplification.

Figure 5.5 shows the structure of our S2SARL model, with a simple simplification example. The RL loop aims to maximize the reward given to the agent at every step during training stage. Therefore, the agent chooses the actions that influence the environment to produce higher rewards. Our RL loop is different from the one in the DRESS system, and it is designed specifically for our task of simplifying a sentence to a specified readability level. The model returns SARI and BLEU scores for the simplified sentence at the end of each step, for comparison purpose.

Dataset	Model	SARI	BLEU
To Level 1	EditNTS	21.92	49.45
1,350 pairs	S2SA	**28.12**	51.23
	S2SARL	26.34	**67.67**
To Level 2	EditNTS	21.89	49.30
10,652 pairs	S2SA	30.97	65.79
	S2SARL	**32.57**	**70.92**
To Level 3	EditNTS	17.12	35.13
9,380 pairs	S2SA	31.56	59.26
	S2SARL	**32.50**	**64.50**
To Level 4	EditNTS	21.60	27.35
2,422 pairs	S2SA	**29.79**	65.78
	S2SARL	29.36	**69.70**

Table 7.2: Test scores for simplification models trained on CSS applied on the CSS test data

7.2.3 Evaluation Method

To evaluate our work, we used EditNTS (Dong et al., 2019) as a notable simplification model to compare our work with. EditNTS uses neural programmer-interpreter to produce a series of edit operations (delete, keep, and add) to operate on the original sentence. The evaluation considers 12 trained versions of each model: EditNTS, S2SA, and S2SARL. Each model is trained against the datasets NAS, CSS, and ASD including the categories from Level 1 to Level 4 for each dataset.

Then after training each model, we report the results using SARI and BLEU scores since they are popularly used in measuring the quality of text simplification models. SARI measures the simplicity of a sentence by focusing on the words added, deleted and kept

Dataset	Model	SARI	BLEU
To Level 1	EditNTS	25.32	61.25
6,478 pairs	S2SA	**32.07**	69.18
	S2SARL	30.75	**70.23**
To Level 2	EditNTS	20.99	46.94
20,432 pairs	S2SA	28.43	60.31
	S2SARL	**32.30**	**65.22**
To Level 3	EditNTS	19.89	33.77
23,301 pairs	S2SA	30.67	50.24
	S2SARL	**32.47**	**56.43**
To Level 4	EditNTS	23.06	24.86
20,048 pairs	S2SA	31.62	44.08
	S2SARL	**32.62**	**51.10**

Table 7.3: Test scores for simplification models trained on ASD applied on the ASD test data

(Alva-Manchego et al., 2019). While BLEU score in the other hand is more related to the meaning preservation as shown in (Xu et al., 2016). After that, we apply the 36 resulted trained models against one common test data. We choose the test part of the ASD dataset, Level 1 to Level 4, since they are not automatically classified and rather assigned by professional editors as mentioned in Section 4.3. We then compare the reported scores.

7.3 Experiments and Results

7.3.1 Training and Testing the Models

We trained our simplification models S2SA and S2SARL along with EditNTS against every readability category, labeled from level 1 to level 4, from the training parts of NAS and CSS datasets. We also trained them against ASD categories, which includes both NAS and CSS datasets as an augmented simplification dataset.

To avoid memory problems due to the vocabulary dictionary size used for embedding each dataset sentences, we used a batch size of 128 for training the models to level 1 and 4, and batch size of 64 for training the models to level 2 and 3. The number of epochs was set to 20 for all the models training over all four categories. We recorded SARI and BLEU scores for all the experiments to measure the simplification models' performance on the set aside test sets.

7.3.2 Results

After training and validating the models, we apply them on the test data that was split from each dataset category. The results on the test data are shown in Table 7.1,7.2, and 7.3. The tables show that S2SARL model always gives the best BLEU score compared with S2SA and EditNTS for all readability levels. However, when the dataset is small, like shown for level 1 and level 4 in Table 7.2, the S2SA model obtains better SARI scores. The model S2SARL gives better SARI results only when trained on a bigger dataset, and that is why we augmented the simplification dataset (to produce the ASD

Sentence	Text
Source	volterra is a town in the tuscany region of italy .
Target	volterra is a town in italy .
Predicted	volterra is a town in tuscany , < *eos* >
Output Readability	Level 3
Source	he was appointed cbe in 1969 .
Target	he was given the honour of cbe in 1969 .
Predicted	he was given the honor of cbe in 1969 . < *eos* >
Output Readability	Level 3
Source	the seat of the district is the town of cossonay .
Target	the capital is the town of cossonay .
Predicted	the capital is the town of < *unk* > .< *eos* >
Output Readability	Level 4
Source	punctuation , capitalization , and spacing are usually ignored , although some -lrb- such as " rats live on no evil star " -rrb- include the spacing .
Target	rats live on no evil star .
Predicted	rats live on no evil star . < *eos* >
Output Readability	Level 3

Table 7.4: Simplification using S2SARL with Newsela and readability level 3

set). EditNTS prefers to generate short sentences with big semantic deviation. It usually deletes important information of the original sentence and generates shorter sentences, as discussed in (Lin and Wan, 2021). This explains the EditNTS scores in the table. However, tuning the weights for the operations (add, delete, and keep) in EditNTS code by giving delete operation the least weight score and assigning a higher weight values to add and keep operations, might improve the results of the EditNTS model.

The results of S2SARL are influenced by the readability level we calculate for every predicted sentence using the classification model discussed in Section 6.4. At every learning step, the agent tries to increase the reward calculated based on the readability level of the predicted sentence. The accuracy of the sentence classification model is 85.52%, as shown in Table 6.8, Since we are dealing with Newsela aligned sentences, we know that the readability levels are labeled manually by professionals, which does not prevent overlapping between adjacent readability levels, as explained in Section 6.3.3 and presented in Table 6.5. Therefore, when aligning the sentences, the overlapping remained between the sentences resulting in lower classification accuracy score.

Table 7.4 shows some of the simplified sentences predicted with S2SARL using Newsela level 3 sentences. The table shows the predicted sentence along with its readability level. The predicted sentence level could be equal to the target level, which is 3 in this example set, or higher like the third sentence, which is level 4, but never less than the target level. That is because the rewards function in the reinforcement loop gives **-0.5** for any predicted sentence level less than the target level, **+2** if the predicted level equals to the target level, and **+1** if the predicted sentence level is higher than the target level. Some examples

produced by S2SARL model are shown in Table 7.4. Predicted sentences could be similar to target sentences but not exactly the same. They could have the same words but with different spelling, e.g., honour and honor, as shown in Table 7.4. Also, some words are not present in the dataset during training, therefore they are not present in the dictionary table (source vocabularies $\times$ target vocabularies). In this situation, the model predicts the symbol $< unk >$ to refer to the unknown vocabulary. The predicted sentence could be longer or shorter than the source sentence since the goal does not include the sentence length. The model aims to produce sentences with readability matching the required level.

Also, S2SARL could predict words in different spelling, i.e. *honour* and *honor* in Table 7.4, the second sentence. Or it could produce the symbol $< unk >$ like the third sentence in Table 7.4. These issues are related to the dictionary created during training phase. The word could appear the dictionary in a different spelling, or could not be found at all.

Also, to compare the performance of the two models S2SARL and S2SA, Table 7.5 shows the prediction of the two models using the augmented dataset with readability level of 4. The table shows how S2SA sometimes produce sentences with lower readability level than we anticipate, which is level 4 in these sentences.

Some examples produced by S2SARL model are shown in Table 7.4. Predicted sentences could be similar to target sentences but not exactly the same. They could have the same words but with different spelling, e.g., honour and honor, as shown in Table 7.4. Also, some words are not present in the dataset during training, therefore they are not present in the dictionary table (source vocabularies $\times$ target vocabularies). In this situation, the model predicts the symbol $< unk >$ to refer to the unknown vocabulary. The predicted sentence

Sentence	Text	Level
Source	thank you for your contributions .	
Target	thank you for your changes .	
S2SA	thank you for your changes . < *eos* >	4
S2SARL	thank you for your changes . < *eos* >	4
Source	the capital of the state is aracaju (pop 664,908) .	
Target	the state 's capital is aracaju .	
S2SA	the capital of the state is . . < *eos* >	3
S2SARL	the capital of the state is aracaju . < *eos* >	4
Source	the birthstone for july would be a red ruby .	
Target	its birthstone is the ruby .	
S2SA	july 's birthstone is the ruby . < *eos* >	3
S2SARL	its birthstone is the ruby . < *eos* >	4
Source	boynton beach was originally incorporated in 1920 as the town of boynton .	
Target	boynton beach was founded in 1920 .	
S2SA	boynton was part of the town of boynton . < *eos* >	4
S2SARL	boynton beach was founded in 1920 . < *eos* >	4

Table 7.5: Simplification using S2SA and S2SARL with level 4 augmented data

could be longer or shorter than the source sentence since the goal is not the sentence length. The model aims to produce sentences with readability matching the required level. Also, to compare the performance of the two models S2SARL and S2SA, Table 7.5 shows the prediction of the two models using the augmented dataset with readability level of 4. The table shows how S2SA sometimes produce sentences with lower readability level than we anticipate, which is level 4 in these sentences.

7.4 Comparison and Analysis

The text simplification models applied in this work (EditNTS, S2SA, and S2SARL) are trained on 12 different datasets: NAS (Level 1 to Level 4), CSS (Level 1 to Level 4), and ASD (Level 1 to Level 4). The experiments produced 36 trained models: 12 EditNTS, 12 S2SA, and 12 S2SARL models as shown in the Tables 7.3, 7.2, and 7.3. To compare the performance of all those models, we test them on the same test data that should not include any automatically classified sentences as targets, i.e, CSS and ASD. Therefore, we tested all the models on the NAS test data (Level-1 to Level-4) since all its target sentences are classified and labeled by expert editors as explained in Section 4.3.

The test results are compared as shown in Table 7.6. Looking at the table, S2SARL model outperforms the other two simplification models across all readability levels. That is due to the involvement of the output sentence readability level during the training phase of the model (in the RL loop).

As shown in Table 7.6, S2SARL models give the best BLEU scores across all four readability levels when trained with ASD since it is the largest simplification dataset (in term of the number of training sentence pairs) compared with NAS and CSS. However, for SARI scores, S2SARL models report the best scores throughout all four readability levels when trained against the CSS dataset. Although ASD is larger than CSS since it contains the CSS and the NAS datasets, training S2SARL model over ASD did not increase the SARI scores. This could be due to the alignment technique used for aligning Newsela sentences (NAS) in (Jiang et al., 2020). The alignment includes sentence splitting, merging, and paraphrasing with deletion which resulted in more meaningful sentences, while the

sentences found in CSS do not include sentence splitting or merging.

To summarise the analysis, S2SARL gives better BLEU scores when trained with ASD (which includes CSS and NAS with sentence splitting, merging, and paraphrasing). That is because BLEU score focuses on grammar and meaning (Van den Bercken et al., 2019). On the other hand, SARI score pays more attention to the lexical aspects of the sentences (Alva-Manchego et al., 2019). Therefore, S2SARL returns good SARI scores when trained against CSS only, where the lexical part is not changed as much compared with the NAS dataset.

7.5 Conclusion

The goal of the simplification is to produce simple sentences at a certain readability level using deep learning models. We used aligned sentences from the Newsela dataset and the augmented dataset created in Section 6.4. Then we created simplification models: Seq2Seq with Attention layer (S2SA) and Seq2Seq with Attention layer and reinforcement learning (S2SARL). The S2SARL model employs the readability level in the simplification process to produce simplified sentence to the desired level. To compare the performance of the created models, we applied S2SA, S2SARL, and EditNTS on the same Newsela aligned sentences. The results of SARI and BLEU scores were compared and analysed. S2SARL reward function depends on the readability level of the sentences that uses the classification model found in Section 6.4 to have about 85% accuracy.

Then we applied S2SA and S2SARL models on the augmented dataset, discussed in

Section 6.4, and showed the improved results of S2SARL. The improvement is due to augmented dataset using the calculated readability level as a factor to produce the dataset.

The next chapter, Conclusion and Future work, will conclude our work in enhancing text readability using deep learning techniques and discuss possible future work that could improve the performance further.

Trained	Model	SARI	BLEU
on NAS Level1	EditNTS	26.48	65.23
	S2SA	31.76	65.61
	S2SARL	31.57	70.22
on CSS Level1	EditNTS	26.41	65.37
	S2SA	34.07	33.35
	S2SARL	**34.08**	36.25
on ASD Level1	EditNTS	26.81	65.70
	S2SA	31.36	73.11
	S2SARL	31.26	**76.47**
Tested on NAS **Level 1** (5,129 pairs)			
on NAS Level2	EditNTS	20.62	46.81
	S2SA	27.18	53.95
	S2SARL	31.56	60.53
on CSS Level2	EditNTS	15.66	46.15
	S2SA	31.36	35.07
	S2SARL	**32.51**	43.67
on ASD Level2	EditNTS	20.63	46.82
	S2SA	25.23	61.78
	S2SARL	31.73	**68.69**
Test on NAS **Level 2** (9,780 pairs)			

Trained	Model	SARI	BLEU
on NAS Level3	EditNTS	20.26	33.28
	S2SA	30.83	45.24
	S2SARL	32.27	53.85
on CSS Level3	EditNTS	15.72	32.36
	S2SA	33.23	23.30
	S2SARL	**33.24**	23.27
on ASD Level3	EditNTS	20.52	33.77
	S2SA	32.21	52.96
	S2SARL	32.23	**61.88**
Test on NAS **Level 3** (13,922 pairs)			
on NAS Level4	EditNTS	23.21	23.97
	S2SA	31.69	42.60
	S2SARL	32.42	50.97
on CSS Level4	EditNTS	12.71	24.22
	S2SA	33.23	12.32
	S2SARL	**33.24**	12.41
on ASD Level4	EditNTS	23.32	24.86
	S2SA	32.31	61.22
	S2SARL	32.32	**61.38**
Test on NAS **Level 4** (17,626 pairs)			

Table 7.6: Test 36 simplification models on ASD test data across all four readability levels

Chapter 8

Conclusion and Future Work

8.1 Conclusion

To conclude our work in this thesis, we provided a complete system that helps enhance text readability. The system applies state-of-the-art technology in deep learning to classify and simplify a text taking into consideration the reader's level of reading.

8.1.1 Summary of Contributions

The system uses syntactic, semantic, and readability metrics features to classify sentences and documents into five classes to match their readability level, labeled from *0* (*most difficult text*) to *4* (*most simple text*) using deep learning techniques. Using this classifier, we were able to generate a corpus of classified pairs of complex and simple sentences based on simple sentences readability level. This could be employed as a larger corpus for text simplification with sentences categorized into five readability levels. The system

uses the generated dataset to simplify sentences using state-of-the-art deep learning techniques. To improve the simplification performance, we applied reinforcement learning loop that uses the readability level of the generated text as part of the simplification process. Incorporating the readability level in the reinforcement learning reward function led to an improvement in the predicted sentences and their readability level was often the required one. Choosing the level of simplification is not an option in other simplification systems. Our simplification system is the only one we are aware of that allows users to specify the readability level wanted for the output text.

8.1.2 Potential Users

This work can be useful to people with comprehension disabilities that affect their reading and language-based processing skills, such as *dyslexia*. These users could use the system to improve their reading fluency, decoding, reading comprehension, and to be able to recall what they read. Users could provide the system with their preferable text that they can easily read to determine their readability level as a target level. Then accordingly, simplify any text they want into that target readability level. Also, children in schools could benefit. The system could simplify a given text to match the child's reading level. It could also help educators to determine the reading level for a given text and decide whether it is suitable for a certain grade or not. Other targeted users can be adults learning English. These users may have low literacy or might be non-native English speakers (Siddharthan, 2002). The system could assist them in reading English documents easily and fluently. It could provide texts with different readability levels to match the users' reading skills.

8.2 Future Work

In future work, it would be beneficial to add human evaluation as a way to measure the predicted sentences. Also, aligned paragraphs found in Newsela corpus could be used to train the simplification model to perform simplification on paragraph level or even on document level. It could also include sentence splitting when appropriate. To further improve the simplification performance, the reward function in the reinforcement loop could include more criteria. These criteria could be SARI score to measure *simplicity*, or cosine between predicted and target sentences vectors to measure similarity between the two, in addition to the readability level classifier. Adding more measurements could provide more control over the simplified sentences' quality. More deep learning techniques could be tested in future work for both tasks (text classification and generating simplified text), since new or improved deep learning methods are frequently published.

In another line of future work, a similar system could be designed for other languages, like Arabic. There are some works on classifying Arabic sentences (Khallaf and Sharoff, 2021) but they use binary classifiers (simple vs complex). We could modify our system to classify Arabic sentences using deep learning techniques into five different classes. We could also extend it to classify paragraphs as well. Then, we can use the classifier to produce a dataset for training Arabic simplification models. There is some work towards Arabic simplification by Khallaf and Sharoff (2022) which uses the seq2seq techniques with Arabic-BERT and FastText. As future work, we could train our model using reinforcement learning on their Arabic corpus and compare our performance with those of (Khallaf and Sharoff, 2022).

Bibliography

Asad Abdi, Shafaatunnur Hasan, Siti Mariyam Shamsuddin, Norisma Idris, and Jalil Piran. A hybrid deep learning architecture for opinion-oriented multi-document summarization based on multi-feature fusion. *Knowledge-Based Systems*, 213:106658, 2021.

G. Adriaens. Simplified english grammar and style correction in an. In *MT Framework: The LRE SECC Project. In: ASLIB Proceedings*, pages 73–82, 1994.

Charu C. Aggarwal and ChengXiang Zhai. A survey of text classification algorithms. In *Mining Text Data*, 2012.

Suha S Al-Thanyyan and Aqil M Azmi. Automated text simplification: A survey. *ACM Computing Surveys (CSUR)*, 54(2):1–36, 2021.

Wejdan Alkaldi and Diana Inkpen. Classifying documents to multiple readability levels. *AAAI 2021 Spring Symposium on Artificial Intelligence for K-12 Education*, 3 2021.

Sandra Aluisio and Caroline Gasperin. Porsimples: Simplification of portuguese texts fostering digital inclusion and accessibility. *Proceedings of the NAACL HLT 2010 Young Investigators Workshop on Computational Approaches to Languages of the Americas*, 06 2010.

Sandra Aluisio, Lucia Specia, Caroline Gasperin, and Carolina Scarton. Readability assessment for text simplification. In *Proceedings of the NAACL HLT 2010 Fifth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 1–9. Association for Computational Linguistics, 2010.

Fernando Alva-Manchego, Louis Martin, Carolina Scarton, and Lucia Specia. EASSE: Easier automatic sentence simplification evaluation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations*, pages 49–54, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-3009. URL https://aclanthology.org/D19-3009.

Fernando Alva-Manchego, Louis Martin, Antoine Bordes, Carolina Scarton, Benoît Sagot, and Lucia Specia. Asset: A dataset for tuning and evaluation of sentence simplification models with multiple rewriting transformations. *arXiv preprint arXiv:2005.00481*, 2020.

Muhammad Zain Amin and Noman Nadeem. Convolutional neural network: Text classification model for open domain question answering system, 2018.

Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. A brief survey of deep reinforcement learning. *arXiv preprint arXiv:1708.05866*, 2017.

Linda Baker, Mariam Jean Dreher, and John T Guthrie. *Engaging young readers: Promoting achievement and motivation*. Guilford Press, 2000.

Himani Bansal, Gulshan Shrivastava, Gia Nhu Nguyen, and Loredana-Mihaela Stanciu.

Social Network Analytics for Contemporary Business Organizations. IGI Global, 03 2018. ISBN 9781522550976. doi: 10.4018/978-1-5225-5097-6.

Yoshua Bengio et al. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009. URL https://ieeexplore.ieee.org/document/8187120.

Sadik Bessou and Ghozlane Chenni. Efficient measuring of readability to improve documents accessibility for arabic language learners. *arXiv preprint arXiv:2109.08648*, 2021.

Joachim Bingel, Gustavo Paetzold, and Anders Søgaard. Lexi: A tool for adaptive, personalized text simplification. In *the 27th International Conference on Computational Linguistics*, pages 245–258, 2018.

G. Bo and H. Xianwu. SVM multi-class classification. *Data Acquis Process*, 3(17), 2006. URL http://en.cnki.com.cn/Article_en/CJFDTotal-SJCJ200603017.htm.

Stefan Bott, Luz Rello, Biljana Drndarević, and Horacio Saggion. Can spanish be simpler? lexsis: Lexical simplification for spanish. In *Proceedings of COLING 2012*, pages 357–374, 2012.

R. Chandrasekar and B. Srinivas. Automatic induction of rules for text simplification. *Knowledge-Based Systems*, 10(3):183 – 190, 1997. ISSN 0950-7051. doi: https://doi.org/10.1016/S0950-7051(97)00029-4. URL http://www.sciencedirect.com/science/article/pii/S0950705197000294.

Kumar Chellapilla, Sidd Puri, and Patrice Simard. High performance convolutional neural

networks for document processing. In *Tenth international workshop on frontiers in handwriting recognition*. Suvisoft, 2006.

Lujing Chen. *Support Vector Machine — Simply Explained: The simplistic illustration of basic concepts in Support Vector Machine*, 2019. URL https://towardsdatascience.com/support-vector-machine-simply-explained-fee28eba5496. accessed: Jun 13, 2020.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014. URL http://arxiv.org/abs/1406.1078.

Christina Clark and Kate Rumbold. Reading for pleasure: A research overview. *National Literacy Trust*, 01 2006.

William Coster and David Kauchak. Simple English Wikipedia: A new text simplification task. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 665–669, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P11-2117.

David Roxbee Cox and E Joyce Snell. *Analysis of binary data*, volume 32. CRC press, 1989. ISBN 9781315137391. doi: https://doi.org/10.1201/9781315137391.

Robert Dale. Gpt-3: What's it good for? *Natural Language Engineering*, 27(1):113–118, 2021.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

S. Devlin and J.I. Tait. *The use of a Psycholinguistic database in the Simplification of Text for Aphasic Readers*, pages 161–173. CSLI Publications, 1998. ISBN 1-57586- 092-9.

Zihan Ding, Yanhua Huang, Hang Yuan, and Hao Dong. Introduction to reinforcement learning. In *Deep reinforcement learning*, pages 47–123. Springer, 2020.

Yue Dong, Zichao Li, Mehdi Rezagholizadeh, and Jackie Chi Kit Cheung. Editnts: An neural programmer-interpreter model for sentence simplification through explicit editing. *arXiv preprint arXiv:1906.08104*, 2019.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: a library for large linear classification. *Journal of Machine Learning Research*, 9: 1871–1874, 08 2008. doi: 10.1145/1390681.1442794.

Daniel Ferrés, Horacio Saggion, and Xavier Gómez Guinovart. An adaptable lexical simplification architecture for major Ibero-Romance languages. In *Proceedings of the First Workshop on Building Linguistically Generalizable NLP Systems*, pages 40–47, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-5406. URL https://aclanthology.org/W17-5406.

R. Fu, Z. Zhang, and L. Li. Using lstm and gru neural network methods for traffic flow prediction. In *2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, pages 324–328, Nov 2016. doi: 10.1109/YAC.2016.7804912.

Andrea Galassi, Marco Lippi, and Paolo Torroni. Attention in natural language processing. *IEEE Transactions on Neural Networks and Learning Systems*, 32(10):4291–4308, 2020.

Christian Giovanelli, Xin Liu, Seppo Sierla, Valeriy Vyatkin, and Ryutaro Ichise. Towards an aggregator that exploits big data to bid on frequency containment reserve market. In *IECON 2017-43rd Annual Conference of the IEEE Industrial Electronics Society*, pages 7514–7519. IEEE, 10 2017. doi: 10.1109/IECON.2017.8217316.

Yoav Goldberg and Omer Levy. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *CoRR*, abs/1402.3722, 2014. URL http://arxiv.org/abs/1402.3722.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

Robert Gunning. The fog index after twenty years. *Journal of Business Communication*, 6(2):3–13, 1969. doi: 10.1177/002194366900600202. URL https://doi.org/10.1177/002194366900600202.

Eui Hong Sam Han and George Karypis. Centroid-based document classification: Analysis and experimental results. In Djamel A. Zighed, Jan Komorowski, and Jan Zytkow, editors, *Principles of Data Mining and Knowledge Discovery - 4th European Conference, PKDD 2000, Proceedings*, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), pages 424–431. Springer- Verlag, 1 2000. ISBN 9783540410669.

Frank E. Harrell. *Regression Modeling Strategies With Applications to Linear Models,*

Logistic Regression, and Survival Analysis. Springer, 2001. ISBN 978-1-4757-3462-1. doi: https://doi.org/10.1007/978-1-4757-3462-1.

James E. Hoard, Richard Wojcik, and Katherina Holzhauser. *An Automated Grammar and Style Checker for Writers of Simplified English*, pages 278–296. Springer Netherlands, Dordrecht, 1992. doi: 10.1007/978-94-011-2854-4_19. URL https://doi.org/10.1007/978-94-011-2854-4_19.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9 (8):1735–1780, November 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735.

Ke Huang. *Unconstrained smartphone sensing and empirical study for sleep monitoring and self-management*. PhD thesis, University of Massachusetts Lowell, 2015.

Kentaro Inui, Atsushi Fujita, Tetsuro Takahashi, Ryu Iida, and Tomoya Iwakura. Text simplification for reading assistance: A project note. In *Proceedings of the Second International Workshop on Paraphrasing - Volume 16*, PARAPHRASE '03, pages 9–16, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics. doi: 10.3115/1118984.1118986. URL https://doi.org/10.3115/1118984.1118986.

Chao Jiang, Mounica Maddela, Wuwei Lan, Yang Zhong, and Wei Xu. Neural crf model for sentence alignment in text simplification. *arXiv preprint arXiv:2005.02324*, 2020.

Shengyi Jiang, Guansong Pang, Meiling Wu, and Limin Kuang. An improved k-nearest-neighbor algorithm for text categorization. *Expert Systems with Applications*, 39(1):1503

– 1509, 2012. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2011.08.040. URL http://www.sciencedirect.com/science/article/pii/S0957417411011511.

Jin Huang and C. X. Ling. Using auc and accuracy in evaluating learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 17(3):299–310, March 2005. ISSN 2326-3865. doi: 10.1109/TKDE.2005.50.

David Kauchak. Improving text simplification language modeling using unsimplified text data. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1537–1546, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P13-1151.

Nouran Khallaf and Serge Sharoff. Automatic difficulty classification of arabic sentences. *arXiv preprint arXiv:2103.04386*, 2021.

Nouran Khallaf and Serge Sharoff. Towards arabic sentence simplification via classification and generative approaches. *arXiv preprint arXiv:2204.09292*, 2022.

J. Peter Kincaid, Robert P. Jr. Fishburne, Richard L. Rogers, and Brad S. Chissom. Derivation of new readability formulas (automated readability index, Fog Count and Flesch Reading Ease formula) for Navy Enlisted Personnel. Technical report, Institute for Simulation and Training, University of Central Florida, Millington, Tennessee, February 1975. URL https://stars.library.ucf.edu/istlibrary/56.

Kamran Kowsari, Kiana Jafari Meimandi, Mojtaba Heidarysafa, Sanjana Mendu, Laura E.

Barnes, and Donald E. Brown. Text classification algorithms: A survey. *CoRR*, abs/1904.08067, 2019. URL http://arxiv.org/abs/1904.08067.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. Recurrent convolutional neural networks for text classification. In *AAAI*, 2015.

Ray R. Larson. Introduction to information retrieval. *Journal of the American Society for Information Science and Technology*, 61(4):852–853, 2010. doi: 10.1002/asi.21234. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/asi.21234.

Walter S. Lasecki, Luz Rello, and Jeffrey P. Bigham. Measuring text simplification with the crowd. In *Proceedings of the 12th Web for All Conference*, W4A '15, pages 4:1–4:9, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3342-9. doi: 10.1145/2745555.2746658. URL http://www.cs.cmu.edu/~jbigham/pubs/pdfs/2015/measuringsimplicity.pdf.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998. ISSN 1558-2256. doi: 10.1109/5.726791.

Hang Li. Deep learning for natural language processing: advantages and challenges. *National Science Review*, 2017.

Tatiana Likhomanenko, Gabriel Synnaeve, and Ronan Collobert. Who needs words? lexicon-free speech recognition. *ArXiv*, abs/1904.04479, 2019.

Xiaojun Wan Zhe Lin and Xiaojun Wan. Neural sentence simplification with semantic

dependency information. In *AAAI Workshop on Deep Learning on Graphs: Methods and Applications*, 2021.

Adam Lopez. Statistical machine translation. *ACM Comput. Surv.*, 40(3):8:1–8:49, August 2008. ISSN 0360-0300. doi: 10.1145/1380584.1380586. URL http://doi.acm.org/10.1145/1380584.1380586.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

Sarah Madi and Ahmed Baba-Ali. *Classification Techniques for Wall-Following Robot Navigation: A Comparative Study*, pages 98–107. Springer, 01 2019. ISBN 978-3-319-99009-5. doi: 10.1007/978-3-319-99010-1_9.

Larry M. Manevitz and Malik Yousef. One-class svms for document classification. *J. Mach. Learn. Res.*, 2:139–154, March 2002. ISSN 1532-4435.

Joseph Marvin Imperial and Ethel Ong. Under the microscope: Interpreting readability assessment models for filipino. *arXiv e-prints*, pages arXiv–2110, 2021.

Word365 Microsoft-Office. Get your document's readability and level statistics, 2021. https://support.microsoft.com/en-us/office.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages

3111–3119. Curran Associates, Inc., 2013. URL http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf.

Karen L. Milheim. *Cultivating Diverse Online Classrooms Through Effective Instructional Design*. Information Science Reference, Walden University, USA, 2018. ISBN 9781522531203. doi: 10.4018/978-1-5225-3120-3.

Swayma Mittal. Deep learning techniques for text classification. *Data Driven Investor*, August 2018.

Sushobhan Nayak, Raghav Ramesh, and Suril R. Shah. A study of multilabel text classification and the effect of label hierarchy. Technical report, CS224N Project Report, Stanford University, 2013.

Newsela Inc. Newsela Dataset. http://https://newsela.com/data/, 2019. Accessed: 2020-05-01.

Christina Niklaus, André Freitas, and Siegfried Handschuh. MinWikiSplit: A sentence splitting corpus with minimal propositions. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 118–123, Tokyo, Japan, October–November 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-8615. URL https://www.aclweb.org/anthology/W19-8615.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, page 311–318, USA,

2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL https://doi.org/10.3115/1073083.1073135.

Egon S. Pearson. BAYES' THEOREM, EXAMTNED IN THE LIGHT OF EXPERDIENTAL SAMPLING. *Biometrika*, 17(3-4):388–442, 12 1925. ISSN 0006-3444. doi: 10.1093/biomet/17.3-4.388. URL https://doi.org/10.1093/biomet/17.3-4.388.

Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1162. URL https://www.aclweb.org/anthology/D14-1162.

Jorge S Pimienta Castillo. Multilingual lexical simplification. *Universitat Pompeu Fabra*, 2021.

Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, 63(10):1872–1897, 2020.

J.R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986. ISSN 1573-0565. doi: 10.1007/BF00116251. URL https://doi.org/10.1007/BF00116251.

J.R. Quinlan. Simplifying decision trees. *International Journal of Man-Machine Studies*, 27(3):221 – 234, 1987. ISSN 0020-7373. doi: https://doi.org/10.1016/S0020-7373(87)80053-6. URL http://www.sciencedirect.com/science/article/pii/S0020737387800536.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9, 2019.

Alexander M. Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1044. URL https://www.aclweb.org/anthology/D15-1044.

Horacio Saggion, Elena Gómez-Martínez, Esteban Etayo, Alberto Anula, and Lorena Bourg. Text simplification in simplext. making text more accessible. *Procesamiento del lenguaje natural*, 47(47):341–342, 09 2011.

Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523, jan 1988. doi: 10.1016/0306-4573(88)90021-0. URL https://doi.org/10.1016%2F0306-4573%2888%2990021-0.

Gandhi Priyank Sanjay and Dr. Viral Nagori. Comparing existing methods for predicting the detection of possibilities of blood cancer by analyzing health data. *International Journal for Innovative Research in Science & Technology*, 4(9):10–14, 2018. URL http://www.ijirst.org/articles/.

Carolina Scarton, Matheus de Oliveira, Arnaldo Candido, Caroline Gasperin, and Sandra M. Aluísio. Simplifica: a tool for authoring simplified texts in brazilian portuguese guided by readability assessments. In *NAACL*, 2010.

Carolina Scarton, Gustavo Paetzold, and Lucia Specia. Text simplification from professionally produced corpora. *the Eleventh International Conference on Language Resources and Evaluation*, 2018.

J. Schmidhuber. Deep Learning. *Scholarpedia*, 10(11):32832, 2015. doi: 10.4249/scholarpedia.32832. revision #184887.

Christie Schneider. The biggest data challenges that you might not even know you have. *IBM*, May 2016. URL https://www.ibm.com/blogs/watson/2016/05/biggest-data-challenges-might-not-even-know/#.

Matthew Shardlow. A survey of automated text simplification. *International Journal of Advanced Computer Science and Applications(IJACSA), Special Issue on Natural Language Processing 2014*, 4(1), 2014. doi: 10.14569/SpecialIssue.2014.040109. URL http://dx.doi.org/10.14569/SpecialIssue.2014.040109.

Advaith Siddharthan. An architecture for a text simplification system. In *In LEC '02: Proceedings of the Language Engineering Conference (LEC'02*, pages 64–71, 2002.

Advaith Siddharthan. A survey of research on text simplification. *ITL - International Journal of Applied Linguistics*, 165:259–298, 01 2014. doi: 10.1075/itl.165.2.06sid.

Abonia Sojasingarayar. Seq2seq AI chatbot with attention mechanism. *CoRR*, abs/2006.02767, 2020. URL https://arxiv.org/abs/2006.02767.

Shengli Song, Haitao Huang, and Tongxiao Ruan. Abstractive text summarization using lstm-cnn based deep learning. *Multimedia Tools and Applications*, 78(1):857–875, 2019.

Anuroop Sriram, Heewoo Jun, Sanjeev Satheesh, and Adam Coates. Cold fusion: Training seq2seq models together with language models. *arXiv preprint arXiv:1708.06426*, 2017. doi: 10.48550/ARXIV.1708.06426. URL https://arxiv.org/abs/1708.06426.

Sanja Štajner, Ruslan Mitkov, and Horacio Saggion. One step closer to automatic evaluation of text simplification systems. In *Proceedings of the 3rd Workshop on Predicting and Improving Text Readability for Target Reader Populations (PITR)*, pages 1–10, Gothenburg, Sweden, April 2014. Association for Computational Linguistics. doi: 10.3115/v1/W14-1201. URL https://www.aclweb.org/anthology/W14-1201.

Sanja Štajner, Simone Paolo Ponzetto, and Heiner Stuckenschmidt. Automatic assessment of absolute sentence complexity. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence, IJCAI*, volume 17, pages 4096–4102, 2017.

Elior Sulem, Omri Abend, and Ari Rappoport. Semantic structural evaluation for text simplification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 685–696, New Orleans, Louisiana, June 2018a. Association for Computational Linguistics. doi: 10.18653/v1/N18-1063. URL https://www.aclweb.org/anthology/N18-1063.

Elior Sulem, Omri Abend, and Ari Rappoport. BLEU is not suitable for the evaluation of text simplification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 738–744, Brussels, Belgium, October-November

2018b. Association for Computational Linguistics. doi: 10.18653/v1/D18-1081. URL https://www.aclweb.org/anthology/D18-1081.

Ilya Sutskever, James Martens, and Geoffrey Hinton. Generating text with recurrent neural networks. In *Proceedings of the* 28^{th} *international conference on machine learning (ICML 11)*, pages 1017–1024, 01 2011.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Tetsuro Takahashi, Tomoya Iwakura, Ryu Iida, Atsushi Fujita, and Kentaro Inui. Kura: a transfer-based lexico-structural para-phrasing engine. In *Proceedings of the 6th Natural Language Processing Pacific Rim Symposium (NLPRS 2001) Workshop on Automatic Paraphrasing: Theories and Applications*, pages 37–46, 11 2001.

Tin Kam Ho. Random decision forests. In *Proceedings of 3rd International Conference on Document Analysis and Recognition*, volume 1, pages 278–282 vol.1, Aug 1995. doi: 10.1109/ICDAR.1995.598994.

Laurens Van den Bercken, Robert-Jan Sips, and Christoph Lofi. Evaluating neural text simplification in the medical domain. In *The World Wide Web Conference*, pages 3286–3292, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017. URL https://arxiv.org/abs/1706.03762.

Ting-Fan Wu, Chih-Jen Lin, and Ruby C. Weng. Probability estimates for multi-class classification by pairwise coupling. *J. Mach. Learn. Res.*, 5:975–1005, December 2004. ISSN 1532-4435.

Wei Xu, Chris Callison-Burch, and Courtney Napoles. Problems in current text simplification research: New data can help. *Transactions of the Association for Computational Linguistics*, 3:283–297, 2015. doi: 10.1162/tacl_a_00139. URL https://www.aclweb.org/anthology/Q15-1021.

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. Optimizing statistical machine translation for text simplification. *Transactions of the Association for Computational Linguistics*, 4:401–415, 2016. doi: 10.1162/tacl_a_00107. URL https://www.aclweb.org/anthology/Q16-1029.

Shudong Yang, Xueying Yu, and Ying Zhou. Lstm and gru neural network performance comparison study: Taking yelp review dataset as an example. In *2020 International Workshop on Electronic Communication and Artificial Intelligence (IWECAI)*, pages 98–101. IEEE, 2020. doi: 10.1109/IWECAI50956.2020.00027.

Kiley Yeakel and Stephanie Tzeng. Autograder: Classifying documents to grade school level. *stanford university*, 2019.

Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. Recent trends in deep learning based natural language processing. *CoRR*, abs/1708.02709, 2017. URL http://arxiv.org/abs/1708.02709.

Xinshi Zang, Huaxiu Yao, Guanjie Zheng, Nan Xu, Kai Xu, and Zhenhui Li. Metalight:

Value-based meta-reinforcement learning for traffic signal control. In *AAAI 2020 - 34th AAAI Conference on Artificial Intelligence*, volume 34 of *AAAI 2020 - 34th AAAI Conference on Artificial Intelligence*, pages 1153–1160. AAAI press, 2020.

Xingxing Zhang and Mirella Lapata. Sentence simplification with deep reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 584–594, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1062. URL https://www.aclweb.org/anthology/D17-1062.

S. Zhou, H. Jeong, and P. A. Green. How consistent are the best-known readability equations in estimating the readability of design standards? *IEEE Transactions on Professional Communication*, 60(1):97–111, March 2017. ISSN 1558-1500. doi: 10.1109/TPC.2016.2635720.

Zhemin Zhu, Delphine Bernhard, and Iryna Gurevych. A monolingual tree-based translation model for sentence simplification. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 1353–1361, Beijing, China, August 2010. Coling 2010 Organizing Committee. URL https://www.aclweb.org/anthology/C10-1152.