VocabEase: A Text Simplification Pipeline

Ziyu Ge, Rajan Singh, Raj Surya, Cuong Ha
Not ChatGPT
{ge005157, sing0780, rajen064, ha000065}@umn.edu

Abstract

Reading levels can be a lot different from person to person and depend on many factors like the language of the text, a person's cognitive abilities, or simply how much they know about the topic. Simplifying and summarizing text means rephrasing it to make it easier for a specific audience to understand. In this project, we are building a pipeline with two main components: the Simplification Module and the Rephrasing Module. The Simplification Module is responsible for finding the complex words in a given text, selects better substitutions, and performs lexical substitution to create a simpler version of the text. For this module, we are trying two different approaches. One approach involves using NLP libraries available in Python, while the other relies on a BERT model to simplify the text with word replacement method. Once the text is simplified, it is passed to the Rephrasing Module, where the fine-tuned T5-Small model cleans it up and makes it more cohesive, fluent, and easier to read. After that, we compare how well both approaches worked, figure out what each does best, and see how our results stack up against existing models like T5 and BART.

1 Introduction

Educational and technical materials are valuable sources of information, but they often rely on complex language and jargon to explain concepts. This can be a huge challenge for learners, especially those with limited vocabulary or non-native speakers, as they will find it difficult to understand the information they are looking for.

Over the past decade, text simplification has gained attention as a way to address this issue, and many researches on this topic have been conducted to make textual information more accessible to a wider range of audience. This is the process of simplifying grammatically complex text while preserving its semantic meaning (Swain et al., 2019), and has been extensively used by researchers all over the world for many purposes. With the significant advancements of Natural Language Processing (NLP), text simplification has become an essential preprocessing step for tasks like parsing (Chandrasekar and Srinivas, 1997; Siddharthan, 2006), machine translation (Oliveira et al., 2010), and even summarization (Chandrasekar et al., 1996). In general, the following three steps are commonly found in most text simplification pipelines: Build a vocabulary for complex words and their potential substitutions; Identify complex words in a given document; And replace the complex words in the document with the most suitable synonyms (Swain et al., 2019).

One common approach to this area involves rule-based methods that use predefined linguistic rules to restructure sentences and substitute complex words with simpler alternatives, as described in (Bott et al., 2012). These methods usually perform well in keeping the grammars correct, but they are less flexible and thus are difficult for them to handle diverse texts. In another paper, (Xu et al., 2016) introduced a method that optimizes statistical machine translation models by training them to simplify sentences using aligned datasets. Even though this approach is promising, it relies heavily on parallel corpora which makes it less robust. Recent advancements in neural network-based methods have managed to overcome some of these challenges. One example is the paper by (Zhang and Lapata, 2017), where the authors introduced a deep reinforcement learning approach to directly learn simplification patterns from data. This approach appeared to be more successful as it can adapt to different text styles.

2 Proposed Methods

The proposed pipeline comprises two major modules: *Simplification* and *Rephrasing*. However, our

focus would be more on the first one.

2.1 Simplification Module

In a high level, this module will perform the following key steps:

1. Takes in a piece of complex text along with a description of the user's understanding level.
2. Differentiates the actual text and the description from the input.
3. Identifies the complex words from the text.
4. Searches for potential substitutions of these keywords.
5. Ranks and filters the substitutions based on their complexity and the user's understanding level.
6. Outputs a draft text with complex words replaced by the most suitable substitutions.

2.2 Rephrasing Module

As the name suggests, this module is responsible for rephrasing the simplified text generated by the *Simplification Module*, just so the final text is more cohesive and fluent. We choose T5-Small model (Raffel et al., 2020) for this task, and there are a few reasons. Firstly, T5-Small is much smaller and faster than larger models in the T5 family like T5-Base or T5-Large. Not only that, like other models in the T5 family, T5-Small is designed specifically for different text-to-text tasks, including text rephrasing. This means the model should be able perform the task very effectively with minimal or even no additional training, unlike models like GPT that often require a lot more fine-tuning.

3 Experiments

3.1 Simplification Module

Here, instead of sticking with only one approach, we decide to try two different ones and see which might be better.

3.1.1 First Approach

The sentence simplification process begins with a text substitution module. The first step involves classifying each word in the sentence as either a "complex" word or a "simple" word based on its complexity score. To compute the complexity score, three key metrics are used: word frequency, word length, and the number of syllables in the word. These metrics are assessed as follows:

1. **Word Frequency**: A database of words from sources like Google Books is used, along with their frequency counts. Words that appear more frequently in language use are considered simpler.
2. **Word Length**: Words that are longer in length are generally more complex, as they tend to be less common and harder to understand.
3. **Number of Syllables**: Words with more syllables are typically more complex, as they are harder to pronounce and understand.

Each of these metrics contributes to the overall complexity score of the word. The higher the word frequency, the lower the complexity; the longer the word, the higher the complexity; and the more syllables a word has, the higher its complexity.

Once the complexity score is calculated, words that exceed a specified threshold are classified as complex words. The threshold for this complexity score can be modified, allowing for the generation of sentences with varying levels of complexity. By adjusting this threshold, it is possible to generate sentences that are either more simplified or retain more of their original complexity, depending on the desired outcome.

These complex words are then reduced to their base or root form using **lemmatization**, which helps to standardize variations of the word (e.g., "running" becomes "run"). After lemmatization, a suitable substitute word is selected for each complex word. This substitute word is chosen from Meta's FastText WiFi News dataset, which provides synonyms along with relevance scores. The relevance score indicates how closely the synonym matches the context of the original word. A lower-complexity substitute word is chosen based on the highest relevance score, ensuring that the replacement maintains the intended meaning while simplifying the vocabulary.

This substitution process is applied iteratively to all words in the sentence, resulting in a sentence with simpler vocabulary. The substituted sentence is then passed through a **grammatical error correction module**. This module ensures that the substituted words are grammatically correct and fit naturally into the sentence. It helps correct any issues related to word forms, ensuring that the sentence remains syntactically sound.

Next, the corrected sentence is sent to a **T5-based pre-trained rephrasing module**, which

is designed to simplify sentence structure. The T5 model (Text-to-Text Transfer Transformer) is a state-of-the-art model for natural language processing that has been trained on a variety of text generation tasks. The rephrasing module simplifies the sentence structure by reducing syntactic complexity, making the text more readable and easier to understand. The model may adjust sentence length, rearrange word order, or simplify clauses to improve clarity.

The final output is a sentence with both simpler vocabulary and a more readable structure, making it easier to understand without losing the original meaning. The simplification is further evaluated through readability metrics, such as the **Flesch-Kincaid Grade Level (FKGL)** and **Flesch Reading Ease (FR)** scores, which measure the text's difficulty level and readability, respectively. These metrics help in assessing the effectiveness of the simplification process.

The threshold for the complexity score used in the text substitution module can be adjusted to generate sentences with varying levels of complexity. By modifying this threshold, sentences with lower or higher complexity can be generated, depending on the target readability level. This flexibility enables the generation of text that meets specific readability requirements or targets a particular audience.

3.1.2 Second Approach

For this approach, we use a Random Forest Classifier to determine whether a word in the text is complex or not. We choose this classifier because its ensemble nature makes it extremely robust to overfitting and helps us reduce some unnecessary work later down the road. After many trials, we notice that configuring the classifier with 100 estimators allows it to perform well while keeping the computation least expensive. If we use fewer estimators, the classifier becomes less stable and fails to generalize well across samples. On the other hand, if we increase the number, the classifier not only has little to no improvement in accuracy, but it also takes longer to run.

The classifier is trained on the CWI Shared Task 2018 Dataset (Yimam et al., 2018). This dataset contains around 27,000 English samples where each of them consists of a sentence, a target word from the sentence, and a binary label indicating whether the word is complex (1) or not (0). The dataset also provides some additional features, but we drop them as they are not relevant to our approach, and instead we engineer our own features including word length, syllable count, word frequency, and Part-of-Speech tag.

Once the complex words have been identified, we choose to use BERT (Devlin et al., 2019) for simplifying complex text instead of creating a method from scratch. Specifically, we take advantage of its *fill-mask* functionality to predict a set of possible substitutions for each complex word in the text. BERT is extremely suitable for this task as it is designed to understand the relationships between words in a sentence, and therefore it is more capable at finding substitutions that closely match the original meaning within the given context compared to other existing models.

3.2 Evaluation Metrics

Evaluating the readability of a text is subjective and varies depending on each person. For example, replacing complex words might be beneficial for non-native speakers, but simplifying and rephrasing the text as a whole could be more appropriate for children with limited literacy skills (Engelmann et al., 2023). In this project, since our main focus is on simplicity rather than coherence or fluency, we evaluate the quality of our simplified text using *Flesch Reading Ease* (FRE) and *Flesch-Kincaid Grade Level* (FKGL).

In general, FRE metric gives a text a score that ranges between 0 and 100, and a higher score means the text is simpler and more accessible. The formula for computing this metric is as follow:

$$\text{FRE} = 206.835 - 1.015\left(\frac{N_W}{N_S}\right) - 84.6\left(\frac{N_{SL}}{N_W}\right)$$

where, in a given piece of text, N_W is the total number of words, N_S is the total number of sentences, and N_{SL} is the total number of syllables. On the other hand, FKGL indicates the minimum education level needed for a person to understand the text. The formula for computing this metric is as follow:

$$\text{FKGL} = 0.39\left(\frac{N_W}{N_S}\right) + 11.8\left(\frac{N_{SL}}{N_W}\right)$$

Even though each serves a fairly different purpose, they both are very commonly used readability metrics for assessing how easy a piece of text is to read and understand.

4 Results

To evaluate the performance, we decide to use ChatGPT to generate 500 complex pieces of text and pass them to the proposed pipeline. Below are some examples:

> The increasing reliance on machine learning models for decision-making in critical areas such as healthcare, criminal justice, and finance requires a thorough evaluation of the potential biases embedded within these systems, as well as the ethical responsibility of developers and organizations to ensure fairness, transparency, and accountability in algorithmic outcomes.
>
> The proliferation of autonomous systems in both military and civilian applications raises profound ethical and legal dilemmas, as decision-making processes traditionally reserved for humans are delegated to algorithms, calling into question the accountability, transparency, and potential for unintended consequences inherent in these technologies.

4.1 First Approach

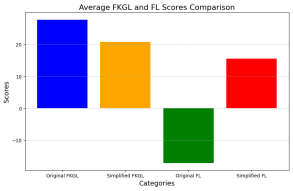

Figure 1: Performance of Method 1. FKGL: Flesch-Kincaid Grade Level; FL: Flesch Reading Ease.

To generate the output, the dataset was split into five equal parts, each containing 100 sentences. These splits were processed in parallel to expedite computation, and the outputs were subsequently combined for analysis. As shown in Figure 1, the pipeline demonstrates significant improvement in readability. The average FKGL score decreased from the original 27.67 to 20.76, indicating that the simplified text is more accessible. Similarly, the average FL score improved from -17.07 to 15.51, reflecting enhanced ease of reading. These results underscore the effectiveness of the pipeline in achieving the desired simplification.

However, despite these improvements, certain drawbacks were observed. One notable limitation is that the substituted words occasionally fail to capture the nuanced meaning of the original words. This issue can lead to a loss of contextual integrity in the simplified sentence. Additionally, the grammatical correction module does not always rectify these issues, resulting in sentences where grammatical structure is not entirely preserved.

Figure 2: FKGL and FL scores for different complexity thresholds.

An important advantage of the proposed method is its flexibility in adjusting the complexity threshold. As illustrated in Figure 2, by lowering the complexity threshold from 1.25 to 0.25, the pipeline produces sentences of reduced complexity. For a lower threshold of 0.25, the pipeline achieved an average FKGL score of 22.81, compared to 23.38 with the higher threshold of 1.25. Similarly, the average reading ease (FL) score increased to 3.73 from 0.98 when the lower threshold was used. Although the differences are not substantial, these results demonstrate the method's adaptability and ability to generate text tailored to varying complexity requirements.

Nonetheless, there are trade-offs associated with decreasing the complexity threshold. The average time taken to process a single sentence increased significantly, from 243 seconds at a threshold of 1.25 to 470 seconds at a threshold of 0.25. This increase is primarily due to the growing number of word substitutions required as the threshold is lowered. Each substitution involves searching a large database of words and their synonyms, leading to increased computational overhead. While the pipeline performs effectively in terms of simplifying text, this slow processing speed is a critical limitation that needs to be addressed for practical applications.

Future improvements could focus on optimizing the synonym search process, perhaps by caching frequently used synonyms or employing more efficient search algorithms. Additionally, fine-tuning the weighting of criteria (e.g., word frequency, length, syllables) during complexity computation could further enhance both the accuracy and performance of the pipeline.

4.2 Second Approach

Below show the Level 1 (Beginner) simplified results for the examples produced by this approach of our pipeline.

> The increasing use on machine learning models for decision making - in difficult areas such as health , public justice , and finance requires a complete study of the potential issues included within these systems , as well as the human role of people and companies to provide quality , control , and control in such outcomes .
>
> The development of such systems in both military and human areas has great personal and legal dilemmas , as decision making processes often held for humans are limited to algorithms , calling into question the control , transparency , and potential for future events present in these systems .

It is noticeable that the simplified texts are slightly shorter than the original ones, and the use of words is also different. However, we notice that the quality of the substitutions is not consistent. For example, some substitutions are fairly accurate and retain the original and contextual meaning very well, such as replacing "criminal justice" with "public justice" or "thorough evaluation" with "complete study". Some of the substitutions are less precise but still acceptable like "proliferation" being replaced with "development" or "civilian applications" being replaced with "human areas". However, there are also cases where the substitutions have little to no connection with the original context, such as replacing "ethical responsibility of developers" with "human role of people".

Figure 3 compares the average scores of the original texts, the texts simplified by our pipeline, and those simplified by models such as T5-Small, T5-Base, BART-Base, BART-Large, and Pegasus-XSum. From the figure, we can see that our pipeline achieves some level of simplification. This is reflected in a slight decrease in the average FKGL (from 27.67 to 23.41) and a decent improvement in the average FRE (from -17.01 to 10.41). One of the main reasons for the insignificant simplification achieved with this approach is the heavy reliance on the vocabulary. The vocabulary used in this approach is too small and does not include all the possible words present in the texts. This limits our approach's ability to accurately evaluate the complexity of a word and lead to situations where certain complex words are overlooked or misclassified as simple. As a result, this approach fails to replace these words, which then impacts the overall effectiveness of the simplification process. We believe that if we can somehow expand and refine the vocabulary further, this limitation can be improved properly.

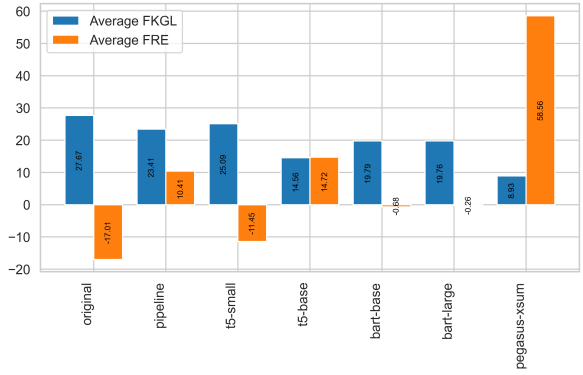

Figure 3: Readability Scores Comparison Between Second Approach and Existing Models

Among the existing models, T5-Base appears to demonstrate a performance trend similar to our pipeline as it also reduce the average FKGL to only 14.56 and increase the average FRE to 14.72. On the other hand, we notice that T5-Small and two BART models, even though achieve reasonable average FKGLs (25.09, 19.79, and 19.76, respectively), fail to show significant readability improvements (average FREs are still negative). This means these models are struggling to simplify the text effectively. We also notice that Pegasus-XSum stands out with the best performance among all approaches, where its average FKGL significantly drops to 8.93, and its average goes way up to 58.56.

5 Discussion

In this section, we go over the questions that are brought up in the project description.

Replicability: How easily are your results able to be reproduced by others?

→ When it comes to this matter, it is hard to guarantee that our results can be replicated exactly as the results from the model or models can be different each time they are run. However, the overall patterns in the results should generally be consistent, and the pipeline's behavior should be predictable, so that we can expect similar outcomes (to a certain degree) can be achieved under the same conditions.

How does using ChatGPT or other models compare to this pipeline?
→ When compared to ChatGPT, there are key differences that separate our pipeline. ChatGPT is based on a Transformer architecture with over 175 billion parameters, whereas the T5-Small model that we utilize employs 60 million parameters. ChatGPT uses a decoder-only transformer architecture, leveraging self-attention to identify the most relevant part of the context and generate the next most likely word using a decoder. On the other hand, the T5-Small model uses an encoder-decoder transformer architecture, where the encoder processes input context and the decoder generates output based on the input. T5 is particularly effective for text simplification tasks as it treats tasks as text-to-text transformations.

ChatGPT is highly effective at using its understanding of language to rephrase complex sentences, breaking down technical jargon into simpler words while maintaining a structure similar to the original input. It is also highly proficient at maintaining fluency and grammatical accuracy, making it an excellent choice for general-purpose simplification tasks.

The T5-Small model, however, has distinct advantages in cases requiring fine-tuning for domain-specific simplifications targeted at non-expert audiences. Its architecture allows T5-Small to build a deeper understanding of the input text and generate simplifications that preserve essential information while simplifying the language. The lightweight nature of T5-Small, with significantly fewer parameters, makes it more computationally efficient and easier to deploy in scenarios with limited computational resources.

While ChatGPT excels in versatility and fluency, T5-Small's ability to specialize for specific tasks through fine-tuning gives it an edge in situations where context-specific simplifications are required.

Datasets: Did your dataset or annotation affect other people's choice of research or development projects to undertake?
→ As for the dataset, we use an existing dataset without making any modifications or additions. Our focus is on using the dataset as-is to train and evaluate our pipeline rather than making changes to the data. Therefore, we do not believe this question is applicable to our work.

Ethics: Does your work have potential harm or risk to our society? What kinds? If so, how can you address them?
→ We do not believe our work can have any potential harm or risks to society. In contrast, it has the potential to be a useful tool (if it was better) for language learners, academic beginners, or anyone looking to simplify complex texts for better understanding.

Discussion: What limitations does your model have? How can you extend your work for future research?
→ Unfortunately, both of our approaches still have a lot of room for improvement. One of the key areas we aim to improve is the simplification capability. As of now, both approaches rely solely on lexical substitutions, and this limits the pipeline to considering only the literal meanings of the words being replaced without fully accounting for the context of the text. Even with BERT's help in the second direction, this approach still falls short because we do not have a good way to provide the model with adequate contextual information. This limitation becomes even more obvious when dealing with composite words or phrases where substituting part of a composite term could significantly change or completely lose the intended meaning.

6 Conclusion

In this project, we develop a text simplification pipeline that focuses on replacing complex words with simpler substitutions and rephrasing the simplified text to create a final version that is easier to understand. We attempt to explore two different approaches in our proposed pipeline: One of them uses existing NLP libraries in Python for generating simpler substitutions, while the other uses pre-trained models for the same task. Both approaches demonstrate some success in reducing text complexity and improving readability, still neither can reach a level where the text is simple enough for a young child, such as a five-year-old, to easily understand. When comparing our methods to existing models, we find that even though our approaches are not performing poorly, they do not entirely meet our expectations. To improve our pipeline, we could expand the vocabulary used to better identify complex words and improve our *Rephrasing Module*. This refinement could include using explanations for words that lack suitable substitutions, rather than relying solely on lexical replacement method.

References

Stefan Bott, Luz Rello, Biljana Drndarević, and Horacio Saggion. 2012. Can Spanish be Simpler? LexSiS: Lexical Simplification for Spanish. In *Proceedings of COLING 2012*, pages 357–374.

Raman Chandrasekar, Christine Doran, and Srinivas Bangalore. 1996. Motivations and Methods for Text Simplification. In *COLING 1996 Volume 2: The 16th International Conference on Computational Linguistics*.

Raman Chandrasekar and Bangalore Srinivas. 1997. Automatic Induction of Rules for Text Simplification. *Knowledge-Based Systems*, 10(3):183–190.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.

Björn Engelmann, Fabian Haak, Christin Katharina Kreutz, Narjes Nikzad Khasmakhi, and Philipp Schaer. 2023. Text Simplification of Scientific Texts for Non-Expert Readers. *arXiv preprint arXiv:2307.03569*.

Francisco Oliveira, Fai Wong, and Iok-Sai Hong. 2010. Systematic Processing of Long Sentences in Rule Based Portuguese-Chinese Machine Translation. In *Computational Linguistics and Intelligent Text Processing: 11th International Conference, CICLing 2010, Iaşi, Romania, March 21-27, 2010. Proceedings 11*, pages 417–426. Springer.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Advaith Siddharthan. 2006. Syntactic Simplification and Text Cohesion. *Research on Language and Computation*, 4:77–109.

Debabrata Swain, Mrunmayee Tambe, Preeti Ballal, Vishal Dolase, Kajol Agrawal, and Yogesh Rajmane. 2019. Lexical Text Simplification Using WordNet. In *Advances in Computing and Data Sciences: Third International Conference, ICACDS 2019, Ghaziabad, India, April 12–13, 2019, Revised Selected Papers, Part II 3*, pages 114–122. Springer.

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. 2016. Optimizing Statistical Machine Translation for Text Simplification. *Transactions of the Association for Computational Linguistics*, 4:401–415.

Seid Muhie Yimam, Chris Biemann, Shervin Malmasi, Gustavo H Paetzold, Lucia Specia, Sanja Štajner, Anaïs Tack, and Marcos Zampieri. 2018. A Report on the Complex Word Identification Shared Task 2018. *arXiv preprint arXiv:1804.09132*.

Xingxing Zhang and Mirella Lapata. 2017. Sentence Simplification with Deep Reinforcement Learning. *arXiv preprint arXiv:1703.10931*.