Constructing Tunable Sentence Simplification Models using Deep Learning

by

Sanqiang Zhao

B.S., Zhengzhou University, China, 2010

M.S., University of Pittsburgh, USA, 2012

Submitted to the Graduate Faculty of

the School of Computing and Information Sciences in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2021

UNIVERSITY OF PITTSBURGH

SCHOOL OF COMPUTING AND INFORMATION SCIENCES

This dissertation was presented

by

Sanqiang Zhao

It was defended on

April 23th 2021

and approved by

Constructing Tunable Sentence Simplification Models using Deep Learning

Sanqiang Zhao, PhD

University of Pittsburgh, 2021

Sentence simplification aims to reduce the complexity of a sentence while retaining its original meaning so that certain individuals can read and understand it. **Substitution**, **Dropping**, **Reordering**, and **Splitting** are widely accepted as four important operations [102]. Recent approaches [102, 88, 90, 53, 94, 98] view the simplification process as a monolingual text-to-text translation, where the translation model learns the operations automatically from examples of complex-simplified sentence pairs extracted from online resources. In the current literature, the two publicly available resources commonly used are Wikipedia and Newsela. However, both resources are limited in several ways, and only contribute to certain operations. As a result, a model trained on these resources favors those operations and lead to inadequate simplification. I argue a truly useful sentence simplification system should simplify a sentence with sufficient operations and even with parameters to enable different ways of achieving simplifications. Current sentence simplification models, particularly deep learning models, cannot do both. This dissertation aims to enhance the usefulness of sentence simplification by exploring two questions: (1) can a sentence be simplified with all suitable operations? and (2) can a sentence simplification model be equipped with parameters to enable different styles of sentence simplification?

To answer the above research questions, I identify three research objectives. The first two objectives focus on addressing the challenges of data shortages. To maximally reduce the sentence complexity, I aim to improve the sentence simplification system to contribute comprehensively to all four operations. First, because of the limitation of the resources I stated above, I introduce an approach to generate a training dataset that will supply or replace the existing training dataset. Via Back-Translation and heuristics, a training dataset with less noise that contributes comprehensively to all operations is generated. Second, due to rich linguistic and simplification resource existed but deep learning model cannot directly use, I explore several deep learning model architectures that enable my model to integrate these resources. My third research objective is to address the flexibility in the simplification style. This is because users in different settings may prefer various forms of simplified sentences, which are related to different simplification operations. Thus, I explore a deep learning model architecture that allows the insertion of style-related parameters for different styles of sentence simplifications.

Table of Contents

List of Tables

List of Figures

1.0 Introduction

1.1 Introduction

Sentence simplification aims to reduce the complexity of a sentence while retaining its original meaning. It can benefit individuals with low-literacy skills [83] including children, non-native speakers and individuals with language impairments such as dyslexia [65] and aphasia [5]. As a result, sentence simplification systems can reduce the reading difficulty and transform sentences so that they are suitable for certain individuals. Not only human readers but also natural language processing applications can benefit from sentence simplification. Such simplification is used as a pre-processor to facilitate parsing or translation tasks. [8]. In those tasks, complex sentences are considered stumbling blocks to such systems. Recently, sentence simplification has also been demonstrated to help with summarization [40], sentence fusion [22], semantic role labeling [81], question generation [29], paraphrase generation [101], and biomedical information extraction [35].

Reading difficulty stems from either lexical or syntactic complexity. Therefore, sentence simplification can be divided into two categories: lexical simplification and syntactic simplification. These two categories can be further implemented by a set of operations. **substitution**, **dropping**, **reordering**, and **splitting** (samples in Table 1) are widely accepted as four important operations [102, 98][1]. The splitting operation divides a long sentence into several shorter sentences to reduce the original sentence's complexity. The dropping operation further removes unimportant or redundant parts of a sentence to make it more concise. The reordering operation interchanges the order of parts or components in a sentence to make its structure and syntax simpler. The substitution operation replaces difficult phrases or words with simplified synonyms.

This dissertation aims to enhance the usefulness of sentence simplification, with particular attention to these four operations. More specifically, this dissertation explores two major questions: (1) can a sentence be simplified with all suitable operations? and (2) Can

[1]There are other definitions of operation, discussed in 2.1.1.

a sentence simplification model support different styles of sentence simplification in which each style can be composed of a set of operations?

1.2 Problem Statement

Earlier work [81, 72, 37, 32] focuses on individual aspects of the simplification problem. For example, several systems performed syntactic simplification using rules aimed at sentence splitting, while others turned to lexical simplification by substituting difficult words with more common synonyms or paraphrases. They are more rule-based and only focus on certain aspects of sentence simplification. Recent approaches [102, 88, 90, 53, 94, 98] are more data-driven. They view the simplification process as a monolingual text-to-text translation, where the translation model learns operations automatically from examples of complex-simplified sentence pairs extracted from online resources.

Although earlier researchers focused on rule-based approaches, current deep learning-based generation models gain better performance in multiple domains [80, 96, 68, 44], including sentence simplification [99, 98], most likely because deep learning models can more easily perceive patterns from large amounts of data. On the contrary, a rule-based approach always requires a large amount of manpower that cannot scale to a large dataset. Therefore, this dissertation focuses on deep learning modeling.

In the current literature, two publicly-available, commonly-used resources are Wikipedia (Table 1) and Newsela (Table 2). However, I found both resources to be limited in several ways, and only contribute to certain operations among substitution, dropping, reordering, and splitting. A model trained on these resources favors those few operations and leads to inadequate simplification.

Table 2 shows sentences in different grade levels (with a lower grade level implying lower reading difficulty) of the same sentence rewritten by professional editors. Obvious, reduced grade level results in more substitution ("nationwide" substitute to "in most states" and "decade" substitutes to "10 years"), dropping (lengths of sentences become shorter), reordering (the syntactic complexity is reduced) and splitting (the sentence in the 7th-grade

Table 1: Sample from Wikipedia.

Source	Sentences
Complex	The vegetation of the small and narrow islands, encompassed by the sea, is very luxuriant; including rainforests, sago, rice and the famous spices - nutmeg, cloves and mace, among others .
simplified (Splitting)	The vegetation of the small and narrow islands, with their wet climate, is very luxuriant. **It** includes rainforests, sago, rice, and the famous spices; including nutmeg, cloves, and mace.
Complex	The town was officially renamed " Allentown " on April 16, 1838, after years of popular usage.
simplified (Re-ordering)	**The name of the town** became " Allentown " on April 16 1833 because it was liked by people.
Complex	He was elected on may 17, 2005, defeating incumbent mayor James Hahn~~, and then re-elected for a second term in 2009~~.
simplified (Dropping)	He was elected on may 17, 2005, defeating the mayor in office, James Hahn .
Complex	Admission to Tsinghua is **extremely** competitive .
simplified (Substitution)	Admission to Tsinghua is **very** competitive

Table 2: Sample from Newsela.

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

is split). By observing this trend, I believe a truly useful sentence simplification system should simplify a sentence with sufficient operations.

After analyzing two publicly-available, commonly-used resources (i.e. Wikipedia and Newsela), I summarize the lessons learned and corresponding goals:

1. Both Wikipedia and Newsela are limited in several ways and contribute only to some operations. Infrequent cases are always treated as noise if they are merely trained using sentence pairs. As a result, a model trained on these resources favors those operations and leads to inadequate simplification. I argue that a truly useful sentence simplification system should simplify a sentence with sufficient operations. Although researchers in the current literature have introduced a variety of methods to facilitate simplicity, most methods have no clear definitions of simplicity and what changes are made to a given complex sentence. Therefore, the first goal of this dissertation aims to enhance the usefulness of sentence simplification by exploring the question: can a sentence be simplified with all suitable operations?
2. Another lesson learned from these two corpora is that excessively focusing on one operation may not be of interest to some users. For example, excessively focusing on the dropping operation may lose the essential meaning of the original complex sentence, which is harmful to meaning preservation. Similarly, excessively focusing on substitution also reduces meaning preservation. For example, "the winner of the Kate Greenaway medal" is not exactly identical to "the recipient of the Kate Greenaway medal". There are different operation contribution rates in both datasets, leading to various styles of simplified sentences. Users in different settings may prefer various forms of simplified sentences. In a more technical perspective, other resources besides the two corpora, such as linguistic and simplification resources, may only contribute to certain styles/operations; this makes integration harder because different resources may conflict with each other. A model that can be equipped with style-related parameters is critical to integrate with those resources. Style-related parameters inform the model of what information is important and relevant. Therefore, the second goal of this dissertation aims to enhance the usefulness of sentence simplification by exploring another question: can a sentence simplification model be equipped with style-related parameters to enable different styles

of sentence simplification?

In sum, this dissertation aims to enhance the usefulness of sentence simplification by exploring two questions: (1) can a sentence be simplified with all suitable operations? and (2) can a sentence simplification model be equipped with style-related parameters to enable different styles of sentence simplification?

To answer the first question, my objective is to improve each operation individually by incorporating various resources. Below is a discussion of the resources that I consider to be particularly useful.

1. To address such challenges as data shortages in Wikipedia and Newsela, I develop an approach to obtain a new training dataset that will supply or replace the existing training dataset. Back-translation (namely, using a translation system to translate the non-English side of the parallel text to get English-English paraphrase pairs) provides a significant number of parallel sentences. Because the pool of parallel sentences does not necessarily contribute to sentence simplification, I select sentences pairs that contribute to certain operations (such as using an existing word mapping databases to recognize whether a substitution is valid). However, back-translation does not perfectly solve the data shortages, because (1) the translation system prefers to preserve meaning (generate a sentence with very similar meaning and structure to original sentence) that leads to generating a few samples that help to drop operation and splitting operation. (2) The translation system may generate noise, such as an influent sentence.
2. To resolve the limitation of back-translation and also supply more resources, another set of resources may be useful are:
   a. To mitigate the limitations of back-translation, I discover existed corpora in other tasks can help the dropping and splitting operations. Finally, I find that corpora in sentence compression and sentence split tasks. The sentence compression corpus is used to train a model that can shorten a sentence, which helps the dropping operation. The sentence split corpus is used to train a model that can split a long sentence, which assists the splitting operation.
   b. To reduce influent sentence noise introduced by back-translation, I use an uncon-

ditional language model to validate the fluency of a sentence. The unconditional language model calculates the probability distribution of a text sequence. Unconditional language models always prefer commonly-used words and sentence structures, which always indicate lower reading difficulty.

c. PPDB is a database containing lexical and paraphrase rules, which are presented as a mapping from a complex word or phrase to a simplified one. Although the mapping is not in sentence form, the database provides rich information on how to conduct substitution.

To answer the second of the above questions, my objective is to allow the insertion of a style-related parameter to enable various styles of sentence simplification.

Users with different settings may prefer various forms of simplified sentences. For example, simplified sentences in Wikipedia and Newsela are two different styles. In order to fulfill different user preferences, I design a deep learning model architecture that allows the insertion of style-related parameters for different styles of sentence simplification.

Another reason for enabling various styles of sentence simplification is that the above resources mostly focus on certain operations, and different resources, which may conflict with each other. For example, the sentence splitting corpus provides information regarding how to perform sentence splitting operations but makes the model ignore lexical simplification. To maximally reduce the sentence complexity, a model that simplifies a sentence with sufficient operations is required. One option is cascading models, but this is sub-optimal for a few reasons. (1) The simplification process will go through multiple models, and one failure in the middle will result in overall failure. (2) The number of models grows as new resources are included. (3) It may force inaccurate simplification. A sentence split model attempts to split every sentence no matter whether it is suitable. Therefore, it is necessary to create a single model that performs all suitable operations. To achieve this aim, I design a style that contains all suitable operations and makes the model generate such a style.

1.3 Research Questions

Based on the above description of my research objectives, I will study the following main research questions in this dissertation. Each research question is then divided into several small sub-questions.

1.3.1 RQ1: How to Analyze the Existing Sentence Simplification Corpus

I quantify each operation and analyze the problems of the current sentence simplification corpus. The two publicly-available sentence simplification corpora commonly used in the current literature are limited in several ways and only contribute to certain operations. As a result, a model trained on these resources favors those operations and leads to inadequate simplification. The model does not fulfill the goal of useful sentence simplification. This issue prompts me to investigate the approaches to improving each operation individually.

1.3.2 RQ2: How to Improve Substitution Operation in Sentence Simplification

1.3.2.1 RQ2.1 How to Use Back-Translation to Generate a Training Dataset

I use back-translation to generate sentence pairs. Then I select pairs containing a replacement that matches PPDB's mapping rules. Such pairs contribute to substitution.

1.3.2.2 RQ2.2 How to Integrate with PPDB

To further improve the performance, I also integrate the PPDB into the sentence simplification model itself. The expectation is to make the model aware of substitutions explicitly, in addition to learning the substitutions implicitly from the data.

1.3.3 RQ3: How to Improve Reordering Operation in Sentence Simplification

1.3.3.1 RQ3.1 How to Use Back-Translation to Generate a Training Dataset

I use back-translation to generate sentence pairs, selecting pairs where the syntax complexity of the translated sentence is lower than the original sentence. Such pairs contribute to substitution.

1.3.3.2 RQ3.2 How to Use PPDB to Generate a Training Dataset

To provide more variant sentence pairs that contribute to the reordering operation, I also employ a syntactic mapping rule in PPDB to obtain more data.

1.3.4 RQ4: How to Improve Dropping and Splitting Operations in Sentence Simplification

1.3.4.1 RQ4.1 How to Use Back-Translation to Generate a Training Dataset

I use back-translation to generate sentence pairs and select pairs that contribute to the dropping and splitting operations. Pairs with shorter translated sentences contribute to the dropping operation. Pairs with a larger number of translated sentences contribute to the splitting operation.

1.3.4.2 RQ4.2 How to Fuse the Models Trained on Other Corpora

Based on my initial experiment and due to the limitation of back-translation, in which the generates sentence prefers to preserve meaning, the back-translation is less helpful to the dropping and splitting operations. This issue prompts me to investigate other alternative corpora. However, different corpora may conflict with each other; for example, a sentence split corpus aims to split a sentence but ignores the substitution. To resolve this issue, I introduce an approach to fuse the models that are trained on the different corpora.

1.3.5 RQ5: How to Allow the Insertion of Style-related Parameters to Enable Different Styles of Sentence Simplification

1.3.5.1 RQ5.1 How to Enable Different Styles of Sentence Simplification

Users in different settings may prefer various forms of simplified sentences, which are related to various simplification operations. Therefore, a deep learning model architecture that allows the insertion of style-related parameters for different styles of sentence simplification is critical.

1.3.5.2 RQ5.2 How to Integrate with the Unconditional Language Model

The unconditional language model calculates the probability distribution of a text sequence. Language models always prefer commonly-used words or sentence structures. Commonly-used words and sentence structures always indicate lower reading difficulty. Therefore, I believe that the unconditional language model indirectly contributes to all of the operations. This prompts me to study how the unconditional language model help the sentence simplification.

1.4 Scope Definition

In this dissertation, I make the following assumptions to define the scope of my study.

1. Although previous research focused on rule-based approaches, current deep learning based generation models achieve better performance in multiple domains [80, 96, 68, 44] including sentence simplification [99, 98]. On the contrary, a rule-based approach always requires a large amount of manpower that cannot scale to a large dataset. Therefore, I believe that focusing on a deep learning based approach is appropriate.
2. Siddharthan [72] states that the sentence simplification can benefit certain types of readers with external simplification rewriting operations. For example, conceptual simplification (where the content, as well as the form, are simplified) benefits child-aged readers.

This dissertation focuses on four major simplification rewriting operations: dropping, substitution, reordering, and splitting, all of which benefit a large number of people with reduced literacy.

1.5 Structure

This section will discuss the overall structure of this dissertation. Chapter 1 introduces the research problem statement and specifies the research questions. Chapter 2 introduces background information (including model and data preprocessing) and related research. Chapter 3 analyzes the existing sentence simplification corpus. And Chapter 4 focuses on constructing a new training corpus based on the analysis. Chapter 5 further evaluates model performance using my constructed corpus. Chapter 6 explains the solution of intensifying the model by allowing multiple styles of sentence simplification. Finally, Chapter 7 made a conclusion and discuss benefits and weaknesses of each components.

2.0 Background and Related Work

2.1 Sentence Simplification

2.1.1 Definition of Operations

Reading difficulty stems either from either lexical or syntactic complexity. Sentence simplification can, therefore, be classified into two types: lexical simplification and syntactic simplification. These two types of simplification can be further implemented by a set of simplification rewriting operations. **Splitting**, **dropping**, **reordering**, and **substitution** are widely accepted as important simplification rewriting operations [102, 98]. Although there are some other definitions of operations, they tend to be highly similar. For example, Xu et al. [94, 18] state that sentence simplification can be achieved by three major types of operations: splitting, deletion, and paraphrasing, with the paraphrasing operation including reordering, lexical substitutions, and syntactic transformations. Reordering and syntactic transformations are similar operations, classified as "reordering" in this dissertation. Woodsend et al.[88] state that sentence simplification includes deletion, substitution, insertion, and reordering. However, "insertion" refers to the insertion of functional words; therefore, it is classified as "reordering" in this dissertation.

2.1.2 Background of Sentence Simplification Models

My deep learning architecture is a conditioned language model. In a regular (unconditioned) language model, each token w_t is conditioned on the previous tokens. The probability of a sequence of words $\mathcal{P}(w_1, w_2, ...w_n)$ is calculated, as in Equation 1.

$$\begin{aligned}\mathcal{P}(w_1, w_2, ...w_n) =& \mathcal{P}(w_n|w_1, w_2, ...w_{n-1})\mathcal{P}(w_{n-1}|w_1, w_2, ...w_{n-2})....\\ & \mathcal{P}(w_2|w_1)\mathcal{P}(w_1)\end{aligned} \tag{1}$$

In a conditional language model, additional context c is added, as in Equation 15. Each token w_t in the sentence is conditioned on previous ones along with the context c.

$$\begin{aligned} \mathcal{P}(w_1, w_2, ...w_n|c) =&\mathcal{P}(w_n|w_1, w_2, ...w_{n-1}, c)\mathcal{P}(w_{n-1}|w_1, w_2, ...w_{n-2}, c)....\\ &\mathcal{P}(w_2|w_1, c)\mathcal{P}(w_1|c) \end{aligned} \tag{2}$$

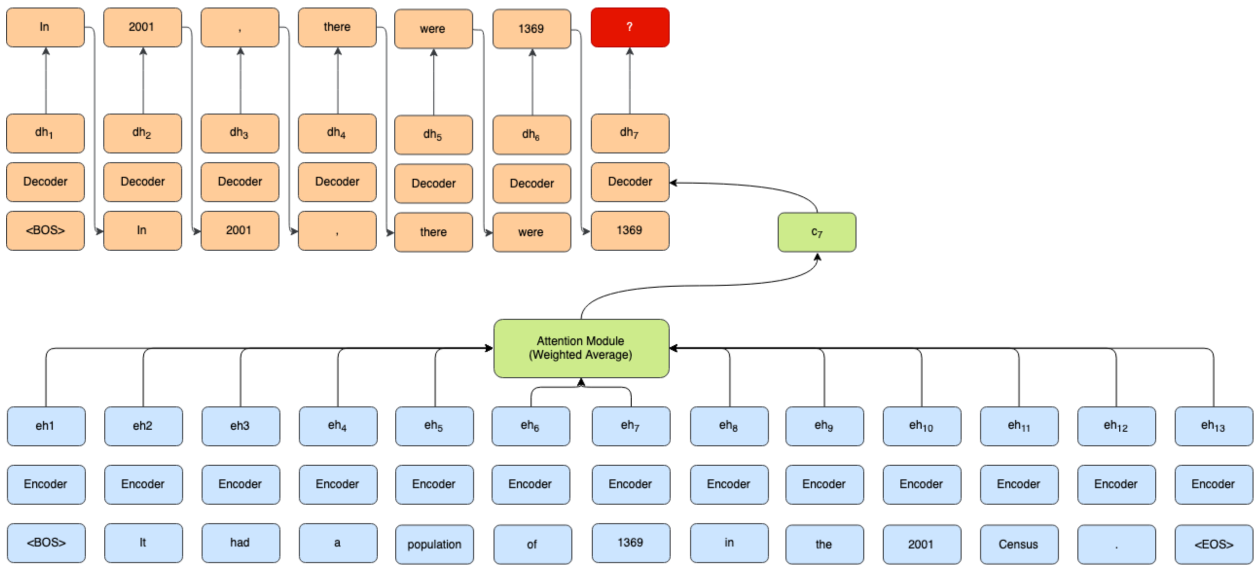


Figure 1: Overview of Deep Learning Architecture for Sentence Simplification

As seen in in Figure 1, there are three major modules. The "Encoder" encodes a complex sentence into a machine-readable representation. A "Decoder" encodes previously generated words in a simplified sentence into a machine-readable representation. An attention module connects representations encoded from the "Encoder" and "Decoder". The complex sentence "it had a population of 1369 in the 2001 Census ." is encoded into a list of hidden states $\{eh_1, eh_2, eh_3, ...eh_{13}\}$ via the module "Encoder". Similarly, the "Decoder" module encodes the previous generated words "in 2001 , there were 1369" into hidden states $\{dh_1, dh_2, dh_3, ...dh_7\}$. An attention module [47] combines the list of hidden states hidden states $\{eh_1, eh_2, eh_3, ...eh_{13}\}$ into context vector c_7 (the figure focuses on the generation of the seventh word) via Equation 3. When generating the next word in a simplified sentence, the "Decoder" considers two pieces of (1) context vector c, which encodes information in the complex sentence and (2) the hidden states of the previously generated words

$\{dh_1, dh_2, dh_3, ...dh_7\}$, which helps generate a fluent sentence.

$$
\begin{aligned}
c_7 &= \sum_i (\alpha_i * eh_i) \\
\alpha_i &= \frac{e_i}{\sum_j e_j} \\
e_j &= exp(eh_j * dh_7)
\end{aligned}
\tag{3}
$$

Previously, both the "Encoder" and "Decoder" can be implemented using a recurrent neural network model, such as RNN/LSTM [52]. Zhao et al. [99] demonstrated the Transformer[80] achieves better performance. Therefore, I focus on implementing the "Encoder" and "Decoder" using Transformer.

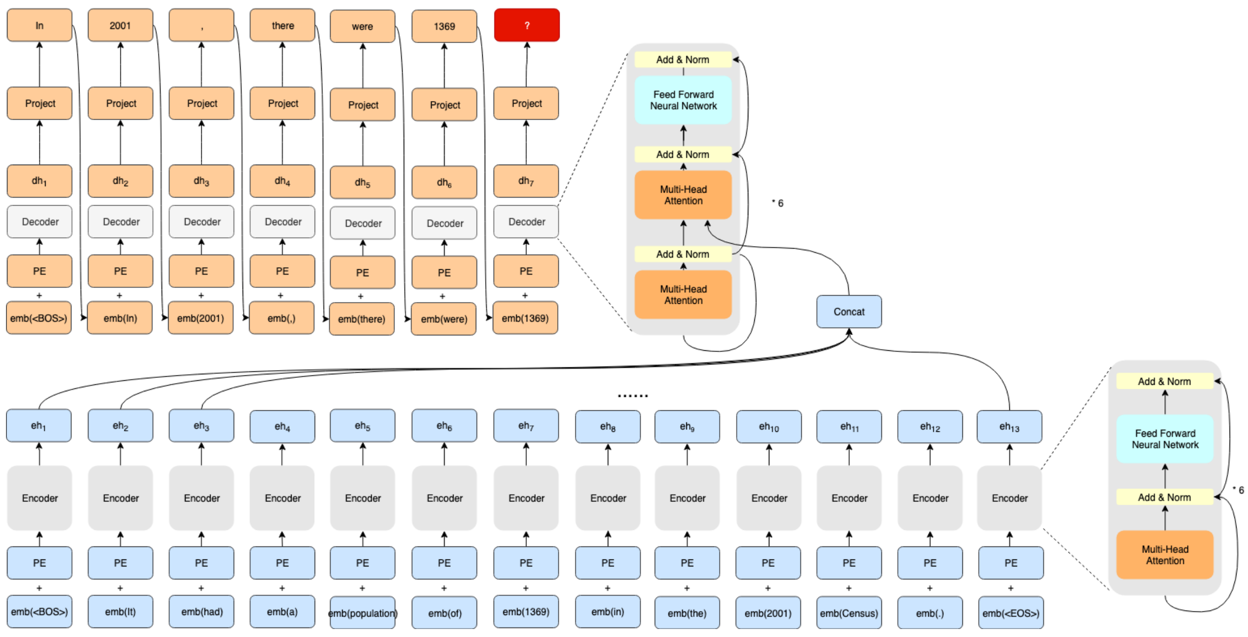


Figure 2: Overview of Deep Learning Architecture for Sentence Simplification

The details of the architecture are illustrated in Figure 2, which describes the same sentence simplification process as in Figure 1.

Each word is represented as a trainable word embedding via a function emb(). Positional encoding (PE) [80] is added to each word embedding for the purpose of injecting information about the relative or absolute position of the tokens in the sentence. The word embedding list is sent to the “Encoder”. The “Encoder” (bottom right corner of Figure 2) is a stack of L identical layers (I set $L = 6$). Each layer has two sub-layers: one layer is for multi-head self-attention and the other is a fully connected feed-forward neural network. The multi-head self-attention layer encodes the output from the last layer into hidden state $eh_{s,l}$ (step s and layer l) as shown in Equation 4, where $\alpha^{enc}_{s\prime,l}$ indicates the attention distribution over step $s\prime$ and layer l. Each hidden state summarizes the hidden states in the last layer through the multi-head attention function a()[80]. The hidden states of the last layer of the “Encoder” are represented as $\{eh_1, eh_2, eh_3, ...eh_{13}\}$.

The “Decoder” (top right corner of Figure 2) also consists of a stack of L identical layers (I set $L = 6$). Along with the same two sub-layers as those in the “Encoder”, the “Decoder” inserts another multi-head attention sub-layer aiming to attend to the encoder outputs. The bottom multi-head self-attention sub-layer plays the same role as the one in the “Encoder”, where the hidden state $dh_{s,l}$ is computed by Equation 5. The above multi-head attention sub-layer is used to find relevant information from “Encoder” outputs. Through the same mechanism, context vector $c_{s,l}$ (step s and layer l) is computed by Equation 6. The hidden states of the last layer of the “Decoder” are represented as $\{dh_1, dh_2, dh_3, ...dh_7\}$. Via Equation 7, each hidden state of the “Decoder” is projected into a vector with vocabulary size and such a vector is used to predict a word (using argmax). The predicted word is sent to the “Decoder” of the next time step.

$$eh_{s,l} = \sum_{s\prime} \alpha^{enc}_{s\prime,l} eh_{s\prime,l-1}, \qquad \alpha^{enc}_{s\prime,l} = a(eh_{s,l}, eh_{s\prime,l-1})^1 \tag{4}$$

$$dh_{s,l} = \sum_{s\prime} \alpha^{dec}_{s\prime,l} dh_{s\prime,l-1}, \qquad \alpha^{dec}_{s\prime,l} = a(dh_{s,l}, c_{s\prime,l-1})^2 \tag{5}$$

$$c_{s,l} = \sum_{s\prime} \alpha^{dec2}_{s\prime,l} eh_{s\prime}, \qquad \alpha^{dec2}_{s\prime,l} = a(dh_{s,l}, e_{s\prime}) \tag{6}$$

$$\begin{aligned} predicted\ word &= argmax(softmax(dh_7 * W_{proj} + b_{proj})) \\ softmax(e) &= \frac{exp(e)}{\sum_i exp(e_i)} \end{aligned} \tag{7}$$

2.1.3 Related Work of Sentence Simplification Models

Earlier work [81, 72, 37, 32] concentrate on individual aspects of the simplification problem. For example, there are several systems that only perform syntactic simplification using rules geared towards sentence splitting. Others perform lexical simplification by substituting difficult words with more common synonyms or paraphrases.

Recent approaches [102, 88, 90, 53, 94, 98] view the simplification process as a monolingual text-to-text translation, where the translation model automatically learns operations from examples of complex-simplified sentence pairs extracted from online resources.

For statistical machine translation models, Zhu et al.[102] propose a tree-based sentence simplification model, drawing inspiration from statistical machine translation. Woodsend et al.[88] use quasi-synchronous grammar and integer programming to score simplification rules. Wubben er al.[90] propose a two-stage model, PBMT-R. A standard phrase-based machine (PBMT) model was trained on complex-simplified sentence pairs, and K-best generations from PBMT were re-ranked based on measurements of dissimilarity from a complex sentence. A hybrid proposed by Narayan et al.[53] is also a two-stage model combining deep semantics and machine translation framework. Zhang et al.[98] argue that the RNN/LSTM model generates sentences without sufficient capability for simplification. They propose DRESS and DRESS-LS, which employ reinforcement learning to reward simpler outputs. SBMT-SARI [94] and DMASS/DESS [99] achieve the 2nd state-of-the-art performance by employing an external knowledge base to promote simplification. Pointer+Ent+Par [27] argued that a good simplified sentence is supposed to be also be logically entailed by its original complex sentence. Therefore, they introduced pointer network for performing copy mechanism, at the same time, they enhanced the entailment and paraphrasing capabilities via multi-task learning by combining auxiliary tasks of entailment and paraphrase generation. NTS+SARI [55] focused on the decoding algorithm, and fine-tuned beam search targeting at high SARI. NSELSTM-S [82] utilized an augmented memory to automatically encode more information in their model. ACCESS [51] proposed a controllable text simplification model that allows explicit ways for users to manipulate and update simplified outputs. Their motivation is similar to mine, but they only focus on the superficial attributes of control, such as sentence length. In addition, the limited attributes in the current publicly available data sets reduces the freedom of their control capabilities.

2.1.4 Related Work of Dataset Generation

The idea of generating a training dataset via back-translation is inspired by automatically generating or discovering paraphrase corpora or parallel texts and they use such methods to learn sentence embeddings.

Many methods have been developed to generate or find paraphrases. Barzilay et al. [3] extracted paraphrases from multiple translations of the same source material. Lin et al. [45] discovered paraphrases by applying a Dependency parser. Dolan et al. [15, 16] discovered articles from multiple news sources. William et al. [12] aligned sentences from standard and Simple English Wikipedia. Xu et al. [93] followed a similar alignment approach but targeted Newsela. Xu et al. [95, 92] discovered paraphrases from crowdsourcing. Suzuki et al.[74] obtained paraphrases from applying diverse machine translation systems to translate a single source sentence. Bannard et al.[2] and Ganitkevitch et al.[24] used methods in statistical machine translation to discover lexical and phrasal paraphrases in parallel text. However, most previous methods generate or discover paraphrases at the phrase level and cannot be extended to large data sets.

Chao et al. [33] improved the quality and quantity of complex-simplified sentence pairs in the training dataset by introducing a novel neural conditional random field (CRF) alignment model. This model not only uses the sequential nature of sentences in parallel documents but also uses the big pre-trained language model to capture semantic similarity. They believe that existing alignment algorithms use surface-level similarity measures, such as the Jaccard coefficient or the cosine distance of the TF-IDF vector, the paraphrase and the context of surrounding sentences cannot be captured. Instead, their CRF alignment model, which is capable to capture deep-level similarity feature, can extract higher quality aligned sentence pairs from parallel documents. However, I found that due to the limitation of available text in the parallel document, the contribution to those rewriting operation is still limited and I am seeking for generative approach to construct higher quality dataset.

My approach is most similar to [86, 85]. They used a neural machine translation system to translate the non-English side of parallel text, resulting in English-English paraphrase pairs. Thus, this method is more easily extended to large data sets due to a large amount of machine translation data. Schwenk et al.[67] and Sennrich et al.[69] improved the target side quality of machine translation by introducing monolingual data obtained by back-translation. Hu et al. [31] followed the same approach to obtain English-English paraphrases pairs. Additionally, to enhance the diversity and adequacy of paraphrases pairs, they used heuristic methods to select the most useful pairs. The heuristic they used is a group of constraints, such as PPDB constraints and, morphological variants. I follow a similar approach to [86, 85] given a complex English sentence and by using another language as a pivot, I employ a neural machine translation system to translate the sentence to the pivot language and translate back. I then use several heuristics to select sentence pairs that contribute to certain simplification rewriting operations. Below, I will discuss the translation system used in this dissertation.

2.1.5 Google's Neural Machine Translation System

Google's Neural Machine Translation system (GNMT) [89, 80] is an end-to-end translation system employing a deep learning architecture similar to that of Figure 2. Compared to the previous phrase-based translation system, it shows the GNMT can achieve roughly a 60% reduction in translation errors on several popular language pairs evaluated by humans. In addition, more studies demonstrate that that neural machine translation generally produces fluent sentences [28, 78, 79, 77]. Therefore, I use GNMT as a base machine translation system to generate data.

2.2 Linguistic and Simplification Resources

2.2.1 PPDB and Simple PPDB

2.2.1.1 Background of PPDB and Simple PPDB

The PPDB (i.e. Paraphrase Database) [24, 59] is an automatically constructed collection of paraphrases. It is built by discovering lexical and phrasal paraphrases from a large amount of parallel text, obtained from statistical machine translation (Table 3).

In addition to the lexical and phrasal paraphrases, PPDB also provides syntactic paraphrases (Table 4), which is helpful for syntactic simplification and reordering operation.

Table 3: Samples of lexical and phrasal paraphrases rules from PPDB

Source	Target
situated	existed
situated	sited
situated	relocated
situated	accommodated
situated	instituted

Table 4: Samples of syntactic rules from PPDB

Source	Target
the manner in which {NN, 1}	the way {NN, 1}
{NNP, 1} s population	the people of {NNP, 1}

Due to the manner in which PPDB is constructed, the paraphrase mapping is not necessarily a simplification mapping. Pavlick et al. [58] trained a supervised model to associate simplification scores with each mapping pair. By truncating the pair with low simplification scores, a subset of PPDB referred to as Simple PPDB, is released. Simple PPDB refers to a paraphrase knowledge base for simplification. It contains 4.5 million paraphrase rules, each of which provides a mapping from a complex word or phrase to simplified ones. More recently, Maddela et al. [49] released a new version of Simple PPDB, termed Simple PPDB++ (Table 5 and 6). Simple PPDB++ is trained on a corpus of human-rated word complexity lexicon with the help of a neural readability ranking model. Simple PPDB++ includes 10 million simplifying paraphrase rules, including lexical (Table 5) and phrasal paraphrases (Table 6). Each rule in the Simple PPDB++ contains a mapping rule along with its simplification score and paraphrase score, in which the simplification score represents the relative complexity of a target word or phrase and the paraphrase score indicates the meaning preservation of the mapping. A low simplification score implies that the target words in the rules are not easily understood, while a low paraphrase score implies that the meaning is not preserved. In this dissertation,I combined simple PPDB and PPDB to improve the recall rate.

Table 5: Sample rules from Lexical Simple PPDB++

Source	Target	Simplification Score	Paraphrase Score
situated	located	1.100474	3.42205
situated	implanted	1.010248	3.29858
situated	stationed	1.001779	3.33203

Table 6: Sample rules from Phrasal Simple PPDB++

Source	Target	Simplification Score	Paraphrase Score
situated in	located in	1.063261	3.91841
situated in	set up at	1.107658	3.69714
situated in	based in	0.958551	3.78694

2.2.1.2 Related Work of Integrating PPDB or Simple PPDB

The format of the substitution mapping rule only reveals which word or phrase is a simplified version of a given complex word or phrase. Because the applying rule is context dependent, direct replacement is not an optimal solution. This issue prompts researchers to investigate how to integrate Simple PPDB into their models.

In the sentence simplification domain, Xu et al. [94] used PPDB paraphrase rules as one of their features in statistical machine translation models. My previous approach [99] used Simple PPDB rules in the neural sentence simplification model. I allocated an external memory for Simple PPDB rules and employed an attention module to select suitable rules to generate a simplified sentence. One drawback of this method is that it is limited to the rules that appear in the training dataset. However, Simple PPDB contains more rules. This prompts me to investigate a better approach of integrating with Simple PPDB.

2.2.2 Unconditional Language Models

2.2.2.1 Background of Unconditional Language Models

Unlike the conditional language models discussed in Section 2.1.2, unconditional language models provide a probability distribution over a sequence of text (Equation 1). The unconditional language model, covering many language aspects, plays an important role in boosting the fluency of generation models, which prompts researchers to integrate language models into sentence simplification models (conditional language models).

Research on the unconditional language model has a long history and only the most popular and recent unconditional language models are included in this dissertation. Unconditional language models are typically left-to-right, which is the same way a sentence is generated. However, if each word in a sentence can only see the context to its left, it clearly loses a great deal of information.

For left-to-right unconditional language models, TransformerXL [13] revises the "Decoder" in the Transformer architecture to enable learning dependency beyond a fixed length. GPT [61] and GPT2 [62] employe "Decoder" in the Transformer architecture with much deeper layers and multi-task objectives. GPT-CAS [14] propose Coordinate Architecture Search (CAS) to search an effective deep learning architecture of the unconditional language model. In their results, the Transformer network, with the addition of the LSTM sublayer, is shown to be an effective unconditional language model architecture.

For bidirectional unconditional language models, ELMo [60] uses bidirectional LSTM and multiple objectives for language modeling. BERT [14], followed GPT [61] and employed deep bidirectional Transformer architecture that results in better performance. In this dissertation, for bidirectional unconditional language models, I focus on BERT because they have released the pre-trained models (it being particularly expensive to train from scratch) and also achieve good performance.

2.2.2.2 Related Work of Integrating Unconditional Language Models

Gulcehre et al.[26] propose shallow fusion. This integrates the unconditional language model by changing the decoding objective (Equation 8), in which $\mathcal{P}_{ulm}(y|x)$ represents the probability generated from the unconditional language model and $\mathcal{P}_{clm}(y|x)$ represents the probability generated from the conditional language model. Shallow fusion also uses a hyperparameter λ to coordinate the probabilities from the two language models.

$$\hat{y} = argmax\, log\, \mathcal{P}_{clm}(y|x) + \lambda log\, \mathcal{P}_{ulm}(y|x) \tag{8}$$

Unlike the above work, which only integrates an unconditional language model in the testing time, Sriram et al.[73] propose cold fusion, which trains the conditional language model first. They then include the pre-trained unconditional language model as a fixed part of the network. They argue that this approach allows the conditional language model to use its model capacity to condition the source sentence because the language aspect is already covered by the pre-trained unconditional language model. Furthermore, their model also includes a gating network which learns to regulate the contributions of the unconditional language model at each time step. Fan et al. [17] modified cold fusion by merging the two language models, resulting in better performance. Devlin et al.[14] treated the unconditional language model as a tool for a second-stage refinement process after the conditional language model predicts the sequence. The benefit of such a two-stage model is that it mitigates the left-to-right constraint because, in the second stage, the entire generated sequence is visible by the refinement process. Zhang et al. [98] employed reinforcement learning (policy gradient) [87] and used perplexity from the unconditional language model as a reward to promote simpler output. Unlike the above methods, the unconditional language model is not combined with the conditional language model in each time step but instead rewards the entire sentence.

2.2.3 Other Corpora

2.2.3.1 Sentence Split

Sentence splitting is the task of breaking down a long sentence into shorter ones that together convey the same meaning.

The WebSplit corpus [54] was constructed by matching sentences in the WebNLG corpus [25]. However, as argued by Botha et al. [4], the WebSplit is sub-optimal because the sentences are (1) often unnatural and (2) not diverse. They proposed to extract sentence split pairs by mining Wikipedia's edit history and released a sentence split dataset called WikiSplit. It contains one million naturally-occurring sentence rewrites, providing 60 times more distinct split examples and a 90 times larger vocabulary than the WebSplit corpus.

2.2.3.2 Sentence Compression

Knight et al. [41] mined a small parallel compression corpus by automatically aligning abstracts to sentences in articles. Filippova et al. [21, 20] extracted a deletion-based compression corpus by aligning news titles to the first sentences in the articles. Clarke et al. [10, 11] released two manually-created compression corpora for deletion-based compression. However, the size of these two manually-created corpora is too small. Toutanova et al.[76] provide a large manually-created corpus constructed by Amazon Mechanical Turk workers.

2.3 Enabling Different Styles of Sentence Simplification

Enabling different styles of sentence simplification requires a controller to manage model generation according to a certain style. The objective of controlling generation is to solve the problem of unequal distribution of the training data and users' objectives. For example, based on training corpus distribution, a model may be likely to generate a long sentence even though users prefer a shorter sentence. The controlling model allows control over the style of a generated sentence and would force the model to generate a short sentence.

Kikuchi et al.[38] controlled the length of a generated summary in a sentence summarization task by feeding a label that indicates an intended output length in addition to the source input. Sennrich et al.[71] introduced a method termed side constraints, which provides a style-related parameter in the source sentence to control of honorifics in the neural machine translation task. The side prefix is also used for multiple domains, such as controlling meta-textual information [84], user personality [43], topic[9] and domain[42]. Johnson et al.[34] employed the side constraints framework for multilingual translation. Ficler et al. [19] followed the side constraints approach and found they could control multiple linguistic style aspects. [75] compared prefix constraints and side constraints, arguing that prefix constraints are more flexible. Niu et al. [56] used a similar approach as the prefix constraints to control politeness in a dialogue generation system. Celikyilmaz et al. [6] employed prefix constraints to control multiple style aspects for neural text summarization.

Reinforcement learning is another approach to controlling generation. In this method, the model rewards a user-preferred style. Niu et al.[56] used reinforcement learning to control the level of politeness. Zhang et al. [98] employed three rewards to represent simpler outputs, such as rewards for simplicity, relevance, and fluency.

2.4 Evaluation

I report the results of an experiment with two metrics that are widely used in the literature of sentence simplification: SARI (System output Against References and against the

Input sentence) [94] and FKGL (Flesch–Kincaid Grade Level) [39].

FKGL computes sentence length (number of words) and word length (number of syllables) as a way of measuring a sentence's simplicity. A lower value of FKGL indicates a simpler sentence.

However, FKGL measures the simplicity of a sentence without considering ground truth simplification references, and it insufficiently correlates with human judgment [94], so I use an additional metric, SARI. SARI computes the arithmetic mean of N-grams (N includes 1,2,3 and 4) F1-score of three rewrite operations: addition, deletion, and keeping. Specifically, it rewards addition operations where a word in the generated simplified sentence does not appear in the original sentence but is mentioned in the reference sentences. It also rewards kept or deleted words in both the simplified sentence and the reference sentences. [94] demonstrates that multi-reference SARI correlates most strongly to human judgment in sentence simplification tasks.

3.0 Analyzing the Existing Corpora

In this section, I introduce the methods used to measure each simplification rewriting operation. Based on that proposed measurement, I analyze the two traditional, publicly-available commonly-used resources, Wikipedia and Newsela. I also analyze the two recently published resources, Wiki-Auto and Newsela-Auto. After the analysis, existing training corpora contains poorly aligned sentence pairs. These poorly aligned sentence pairs result from either poor algorithmic extraction from parallel articles or articles that are not very parallel. In addition, existing training corpora are also limited in several ways and contribute to only some operations. Infrequent cases are always treated as noise if they are merely trained using sentence pairs. The analysis results prompt me to develop research questions in the next chapters.

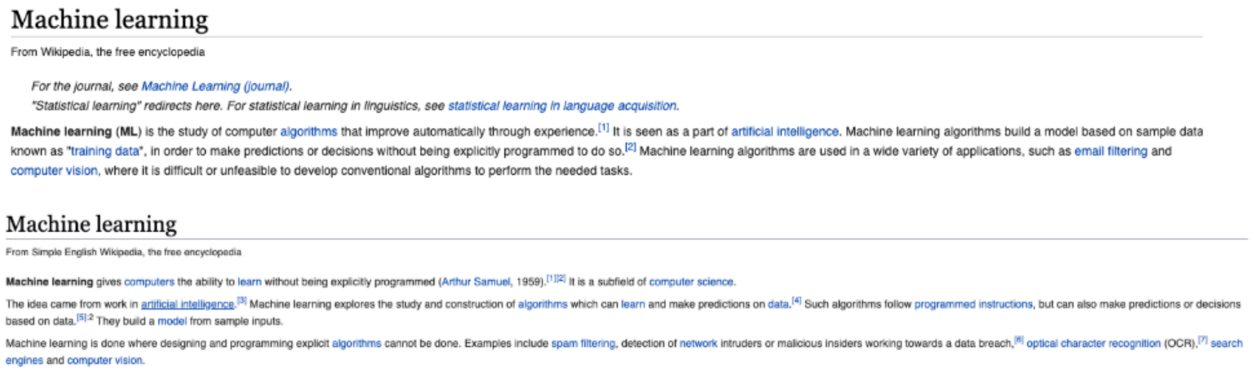
3.1 Measurement of Operation

Currently, most measurements are done by humans, and these measurements are concentrated on small development/test data sets. However, it is difficult to ask humans to measure the entire training data set. Also because of the measurements are straightforward, which makes automatically measurement possible. In this section, I introduce the automatic methods for measuring each simplified rewrite operation.

3.1.1 Relevance Measurement

Most existing datasets are collected from paired complex-simplified articles. Figure 3 and 4 show screenshots of parallel articles from Wikipedia and Newsela respectively. For the screenshot of paralleled articles for Wikipedia, I put the first paragraph of the "Machine Learning" article from Wikipedia (top) and Simple Wikipedia (bottom). Because two versions of Wikipedia are created separately, it is difficult to extract aligned complex-simplified sentence pairs. For the screenshot of parallel articles from Newsela, I put an article written by an editor for different grade levels. The article on the left is for grade 9, and the one on the right is for grade 3. Because Newsela editors create paired articles in the document level, some sentences in the left article are misaligned sentences from the article on the right.

As shown above, due to various reasons, sentence pairs can be poorly aligned. To rewrite a text in a different readability level, editors may drastically reconstruct its words, sentences, or even paragraphs. In consequence, sentences in paraphrased articles may not accurately pair with the original ones. The misalignment can become even worse when sentences are automatically extracted and aligned with simple lexical-based features such as Jaccard similarity [102, 93]. The misalignment can come from poorly performed extraction and alignment algorithms and from a poorly aligned article itself, as shown in Figures 3 and 4.


Machine learning

From Wikipedia, the free encyclopedia

For the journal, see Machine Learning (journal).
"Statistical learning" redirects here. For statistical learning in linguistics, see statistical learning in language acquisition.

Machine learning (**ML**) is the study of computer algorithms that improve automatically through experience.[1] It is seen as a part of artificial intelligence. Machine learning algorithms build a model based on sample data known as "training data", in order to make predictions or decisions without being explicitly programmed to do so.[2] Machine learning algorithms are used in a wide variety of applications, such as email filtering and computer vision, where it is difficult or unfeasible to develop conventional algorithms to perform the needed tasks.

Machine learning

From Simple English Wikipedia, the free encyclopedia

Machine learning gives computers the ability to learn without being explicitly programmed (Arthur Samuel, 1959).[1][2] It is a subfield of computer science.

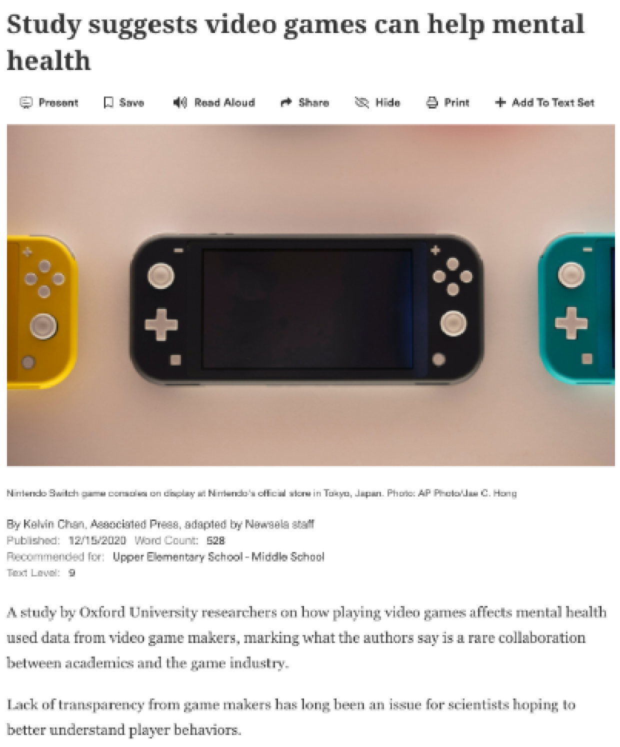
The idea came from work in artificial intelligence.[3] Machine learning explores the study and construction of algorithms which can learn and make predictions on data.[4] Such algorithms follow programmed instructions, but can also make predictions or decisions based on data.[5]:2 They build a model from sample inputs.

Machine learning is done where designing and programming explicit algorithms cannot be done. Examples include spam filtering, detection of network intruders or malicious insiders working towards a data breach,[6] optical character recognition (OCR),[7] search engines and computer vision.

Figure 3: Screenshot of parallel articles for Wikipedia

Such less-aligned sentence pairs confuse the model. My earlier experiment found that such noise seriously affects the readability of model prediction sentences. The relevance of the sentence pairs needs to be measured to assess the influence of this noise.

Measuring the relevance of two sentences is complex, and there are multiple relevance measurement models based on different configurations. "Universal Sentence Encoder"[7] is an earlier sentence embedding model trained on a large variant of natural language processing (NLP) tasks and corpora. "Sentence BERT" [64] fine-tuned the bidirectional encoder representations from transformers (BERT) method so that its sentence embedding contains



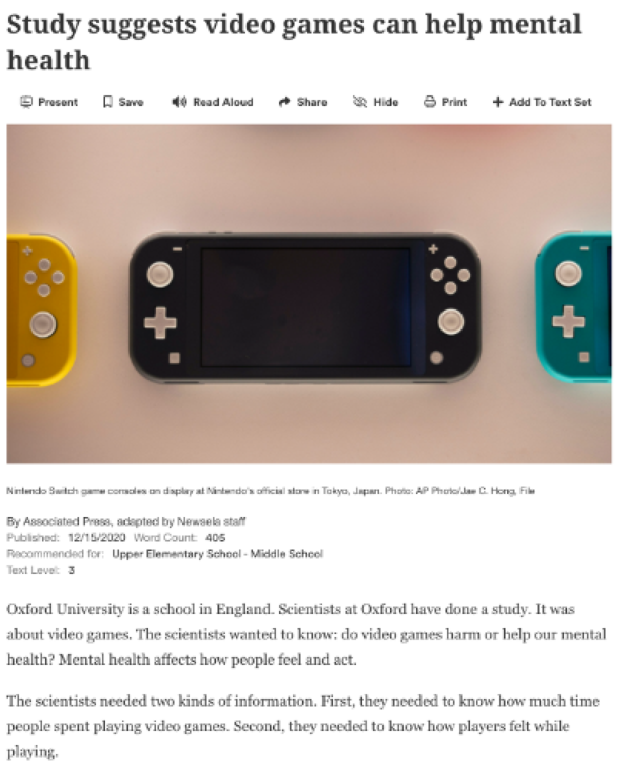
Study suggests video games can help mental health

Nintendo Switch game consoles on display at Nintendo's official store in Tokyo, Japan. Photo: AP Photo/Jae C. Hong

By Kelvin Chan, Associated Press, adapted by Newsela staff
Published: 12/15/2020 Word Count: 528
Recommended for: Upper Elementary School - Middle School
Text Level: 9

A study by Oxford University researchers on how playing video games affects mental health used data from video game makers, marking what the authors say is a rare collaboration between academics and the game industry.

Lack of transparency from game makers has long been an issue for scientists hoping to better understand player behaviors.



Study suggests video games can help mental health

Nintendo Switch game consoles on display at Nintendo's official store in Tokyo, Japan. Photo: AP Photo/Jae C. Hong, File

By Associated Press, adapted by Newsela staff
Published: 12/15/2020 Word Count: 405
Recommended for: Upper Elementary School - Middle School
Text Level: 3

Oxford University is a school in England. Scientists at Oxford have done a study. It was about video games. The scientists wanted to know: do video games harm or help our mental health? Mental health affects how people feel and act.

The scientists needed two kinds of information. First, they needed to know how much time people spent playing video games. Second, they needed to know how players felt while playing.

Figure 4: Screenshot of parallel articles for Newsela

information from both word embedding and sentence embedding.

Both models aim to encode sentences into a single embedding vector that specifically transfers learning to other NLP tasks. The model is efficient and results in accurate performance on diverse transfer tasks, including semantic textual similarity tasks. Because the model is trained on a large variant of NLP tasks and corpora, I believe that the generated embedding is accurate and helpful for sentence encoding and can decide whether two sentences are correlated sufficiently.

Given a complex sentence s$_{comp}$ and a simplified sentence s$_{simp}$, the model encodes them into vec$_{comp}$ and vec$_{simp}$, respectively. Then, I calculate the cosine similarity c(vec$_{comp}$, vec$_{simp}$) (Equation 12), which indicates the "semantic" similarity between s$_{comp}$ and s$_{simp}$. Although the sentence structure and word usage in s$_{comp}$ and s$_{simp}$ are quite different, they still have a high c(vec$_{comp}$, vec$_{simp}$) score. Therefore, I use the c(vec$_{comp}$, vec$_{simp}$) score as an indicator to check whether s$_{comp}$ and s$_{simp}$ are correctly aligned.

$$c(vec_{comp}, vec_{simp}) = \frac{vec_{comp}\ vec_{simp}}{||vec_{comp}||\ ||vec_{comp}||} \tag{9}$$

Different from both "Universal Sentence Encoder" and "Sentence BERT," the BERT Score [97] computes a similarity score at the token level. More specifically, the score is calculated by comparing each token in the complex sentence with each token in the simplified sentence. The BERT score computes the precision and recall for each word in the complex and simplified sentence and uses the F1 score as an indicator to check whether s$_{comp}$ and s$_{simp}$ are correctly aligned.

$$Recall = \frac{1}{|comp|} \sum_{i \subset comp} max(vec_{comp_i} vec_{simp_j}) \tag{10}$$

$$Precision = \frac{1}{|simp|} \sum_{j \subset simp} max(vec_{comp_i} vec_{simp_j}) \tag{11}$$

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} \tag{12}$$

I used the above three methods to measure relevance in the following analysis. A higher value indicates a higher relevance, and a lower value indicates a lower reference. Lower relevance is always due to poorly aligned sentence pairs. However, extremely high relevance also indicates poor simplification.

3.1.2 Substitution Measurement

Substitution means replacing difficult phrases or words with simplified synonyms. It plays a central role in reducing reading difficulty.

To measure substitution, I adopt the paraphrase database (PPDB) as discussed in Section 2.2.1.1. PPDB provides substitution mapping rules that can help to check whether sentence pairs contain a valid substitution.

Given a complex sentence s_{comp} and a simplified sentence s_{simp}, I examine each n-gram (n includes 1,2 and 3) in s_{comp}. If any n-gram is included as a complex form in PPDB, I check whether the simplified form is in s_{simp} (and also check that its complex form does not occur in s_{simp}). In this way, I can validate that there is at least one substitution in the sentence pairs s_{comp} and s_{simp}. Thus, I compute the ratio of words in the complex sentences replaced by PPDB to measure the substitution rewriting operation.

3.1.3 Dropping Measurement

Dropping refers to removing unimportant or redundant parts of a sentence to make it more concise.

To measure dropping, I adopt an edit-distance algorithm [36]. A drop in a sentence may come from substitution (a substitution is composed of a drop and an insertion). By feeding PPDB into the edit-distance algorithm, the shortest edit distance is calculated between two sentences s_{comp} and s_{simp} (Table 7). The shortest edit distance path represents the sequence of edit actions that change from s_{comp} to s_{simp}, and the list of edit actions includes adding, deleting, replacing, and keeping. Similarly, I compute the ratio of words from complex sentences that are removed to measure the dropping rewriting operation.

Table 7: Sample of dropping measurement

Source	Sentences
Complex	Many programs are limited to parents , or are tied to work .
Simplified	Many programs are only for parents .
Edit	KEEP:Many, KEEP:programs, KEEP:are, ADD:only, ADD:for, DEL:limited, DEL:to, KEEP:parents, DEL:,, DEL:or, DEL:are, DEL:tied, DEL:to, DEL:word, KEEP:.

3.1.4 Reordering Measurement

Reordering indicates the interchanging of parts or components in a sentence to simplify its structure and syntax.

To measure the reordering, I compare the syntax and sentence structure in s$_{comp}$ and s$_{simp}$. More specifically, I must check whether the relationships among words have changed.

Following Xu et al.'s [93] method to analyze syntax patterns, I use a dependency parser [57] to extract relationships for each word and compute its signature. As seen in Table 8, given the original sentence "It had a population of 1,369 in the 2001 Census" and the translated sentence "In 2001, there were 1,369 inhabitants," I generate the signature for each sentence using a dependency parser. The signature is a set of relationships from each word, including a head word, a relationship, and a tail word. For example, "in $\xrightarrow{pobj}$ 2001" shows that the relation from "in" to "2001" is "pobj" (the object of a preposition). I then calculate the Jaccard similarity (Equation 13) for two sets of signatures.

$$J(sign1, sign2) = \frac{|sign1 \cap sign2|}{|sign1 \cup sign2|} \tag{13}$$

3.1.5 Splitting Measurement

The splitting operation divides a long sentence into several shorter sentences to reduce the complexity of the original sentence.

Table 8: Samples from Google Translation System

Original	It had a population of 1,369 in the 2001 Census .
Signature	in $\xrightarrow{pobj}$ 2001, had $\xrightarrow{dobj}$ population, population $\xrightarrow{prep}$ of, population $\xrightarrow{prep}$ in, ...
Translated	In 2001, there were 1,369 inhabitants .
Signature	in $\xrightarrow{pobj}$ 2001, were $\xrightarrow{prep}$ in, were $\xrightarrow{expl}$ there, were $\xrightarrow{attr}$ 1,369, ...

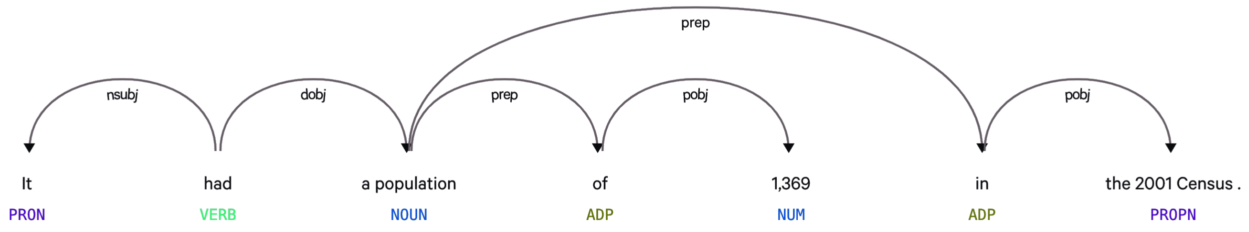


Figure 5: Dependency Parser Output

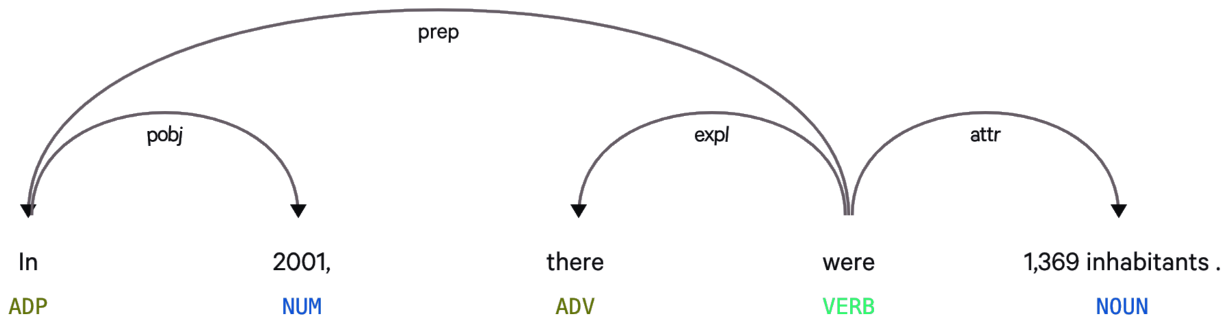


Figure 6: Dependency Parser Output

Measuring the splitting is straightforward. Given a complex sentence s_{comp} and a simplified sentence s_{simp}, I can compare the number of sentences on each side. If the sentences on the complex is larger than the sentences on the simplified side, I treat current sentence pairs as contributing to the splitting rewriting operation.

More specifically, I use an NLP toolkit Spacy [30] to preprocess s_{comp} and s_{simp}, which reveals how many sentences are in s_{comp} and s_{simp}. If the number of sentences in s_{simp} is larger than the number in s_{comp}, I mark the sentence pairs as contributing to the splitting operation. The measurement of the splitting operation is a boolean value that indicates whether the complex sentence will be split.

3.2 Analyzing the Existing Training Datasets

This section analyzes existing training datasets, including Wikipedia, Newsela, Wiki-Auto, and Newsela-Auto. At the first glance, existing training corpora contains poorly aligned sentence pairs. Furthermore, existing training corpora are also limited in several ways and contribute to only some operations. The analysis prompts me to study the remaining research questions.

Wikipedia is the first data corpus of interest to researchers. Simple English Wikipedia [1] and English Wikipedia [2] are two versions of English Wikipedia. Articles on the Simple English Wikipedia are usually simpler than articles on English Wikipedia and typically only provide basic information. The basic presentation style in Simple English Wikipedia makes it an ideal choice for beginners learning English. As a result, researchers have collected aligned articles from English Wikipedia and Simple English Wikipedia and found alignment sentences in aligned articles [102].

However, as Xu et al.[93] indicate, the aligned sentences from English Wikipedia and Simple English Wikipedia are sub-optimal because (1) they are prone to automatic sentence alignment errors, (2) they contain a large proportion of inadequate simplifications, and (3)

[1]https://simple.wikipedia.org
[2]https://www.wikipedia.org

they generalize poorly to other text genres. Therefore, they argue that focusing on Wikipedia limits simplification research. Xu et al. introduce a new simplification dataset, Newsela [3]. Newsela is a teaching content platform that enhances reading participation and education for each topic. Newsela also provides articles that have been rewritten four times for different grades by Newsela's professional editors. This work forms a parallel corpus that allows for the alignment of sentences at different reading levels. In contrast to Simple Wikipedia, which was created without a well-defined objective, Newsela is intended to help teachers prepare curricula that match the English language skills required at each grade level.

Jiang et al.[91] argued existing complex-simplified sentence pairs from the above two corpora fail to capture paraphrases and the context of surrounding sentences because those pairs were extracted with the help of lexical-based features, such as the Jaccard coefficient or the cosine distance of term frequency-inverse document frequency (TF-IDF) vectors. Instead, they created a small, high-quality, manually annotated, sentence-aligned corpus: Newsela-Manual with 50 article sets and Wiki-Manual with 500 article pairs. Then, they trained a neural CRF model, equipped with a BERT as a base model, to capture semantic similarity and leverage the similar order of content. Using such a model, they constructed two large corpora, Wiki-Auto and Newsela-Auto. Experiments show the Wiki-Auto and Newsela-Auto perform better than previous alignment approaches.

For reference, Xu et al. [94] provide a validation and test dataset called Turk. In the Turk dataset, eight simplified reference sentences for each complex sentence are used as a factual basis, and all these sentences are generated by Amazon Mechanical Turk workers. The Turk dataset contains 2,000 data samples for validation and 356 samples for testing. Turk is a reliable data set because (1) it is human-generated, and (2) each complex sentence has multiple, equally good simplified forms.

Fernando et al. [1] argued Turk put most of its attention on the lexical simplification. In their work, they extended the Turk dataset by asking Amazon Mechanical Turk workers to write simplified sentences to maximize sentence rewriting operations. Their dataset, ASSET, extended the Turk dataset with the same complex sentences and provided ten simplified reference sentences for each complex sentence.

[3]https://newsela.com

In the following, I treat Turk and ASSET as references because both are written by humans and provide multiple simplified sentences. I analyze the current training corpora, including Wikipedia, Newsela, Wiki-Auto and Newsela-Auto, based on these references.

3.2.1 Relevance Analysis

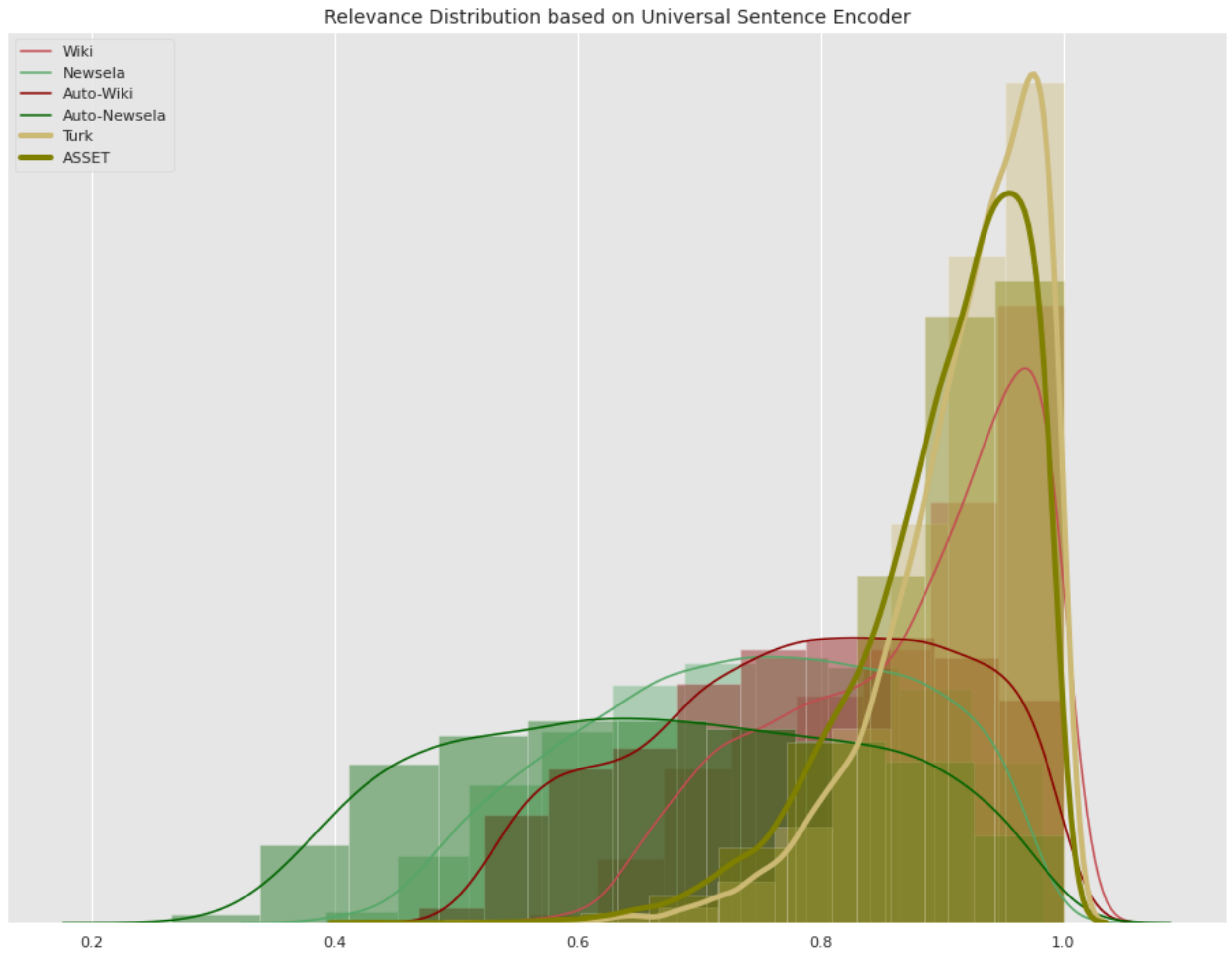


Figure 7: Relevance Analysis of Using Universal Sentence Encoder (Histogram)

Figure 7, 9, 11 (Histogram) and 8, 10, 12 (BoxPlot) show the distribution of relevance simplification rewriting operation measurement for all the datasets. Both Turk and ASSET have strong relevance scores, and almost all the sentence pairs are above 0.9 similarity scores. The ASSET relevance score is slightly lower than that of Turk because sentence pairs in

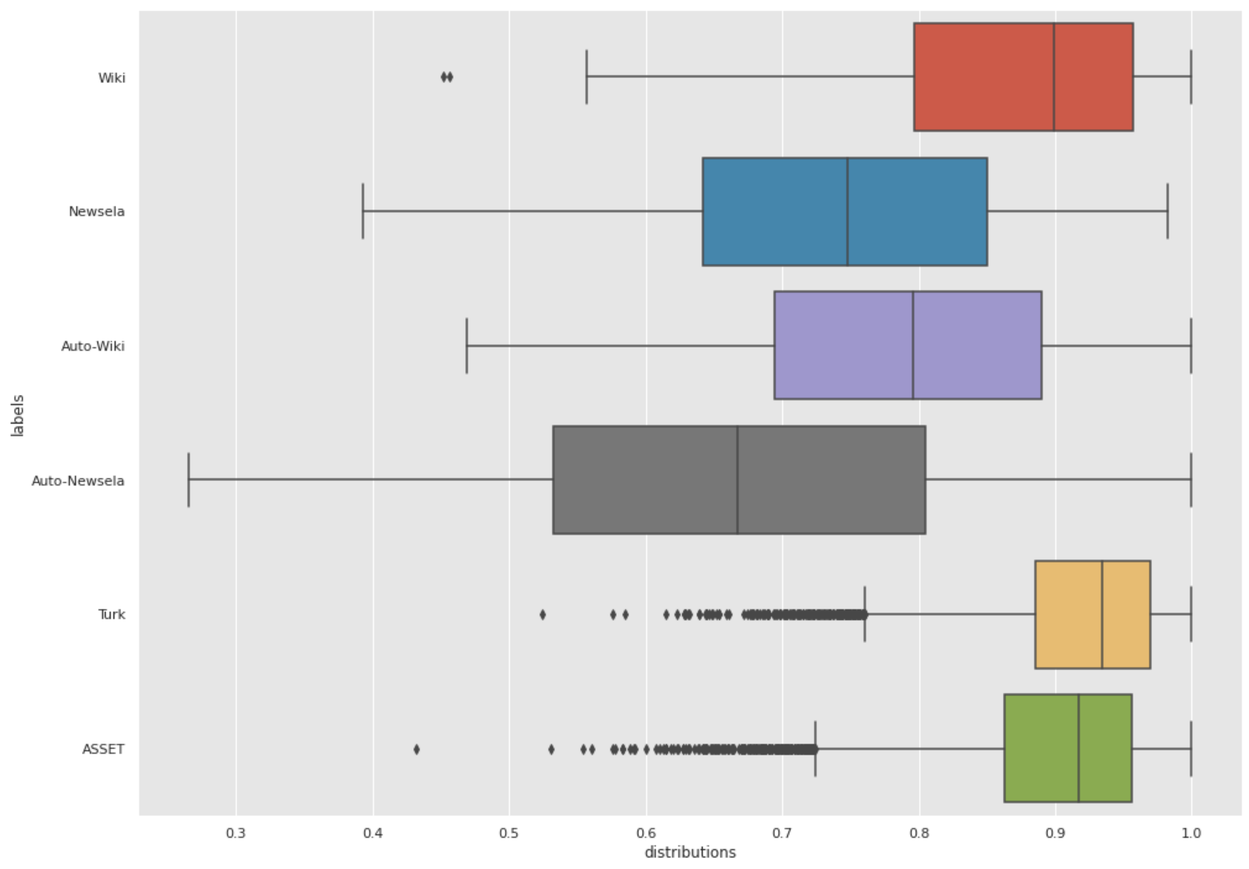


Figure 8: Relevance Analysis of Using Universal Sentence Encoder (BoxPlot)

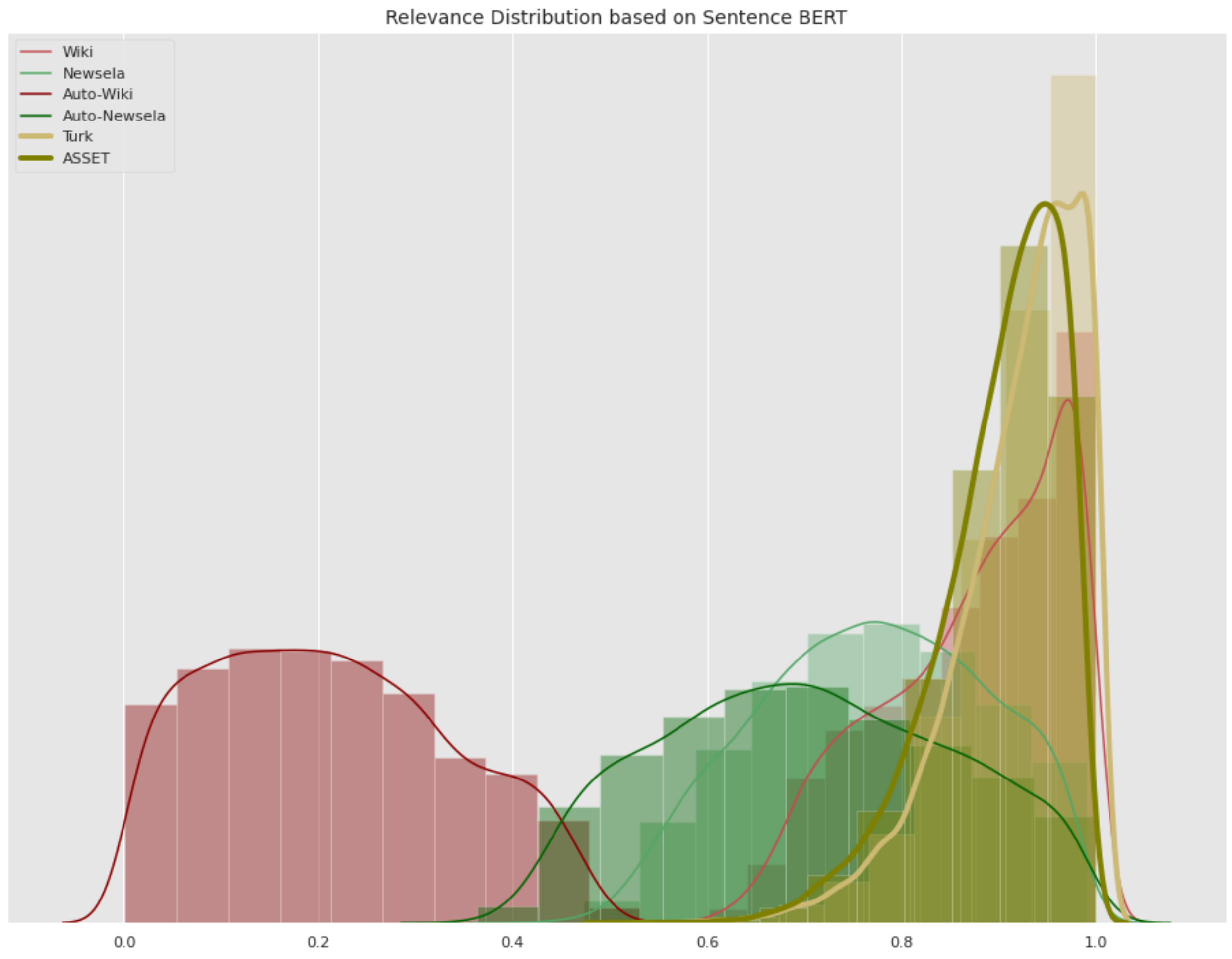


Figure 9: Relevance Analysis of Using Sentence BERT (Histogram)

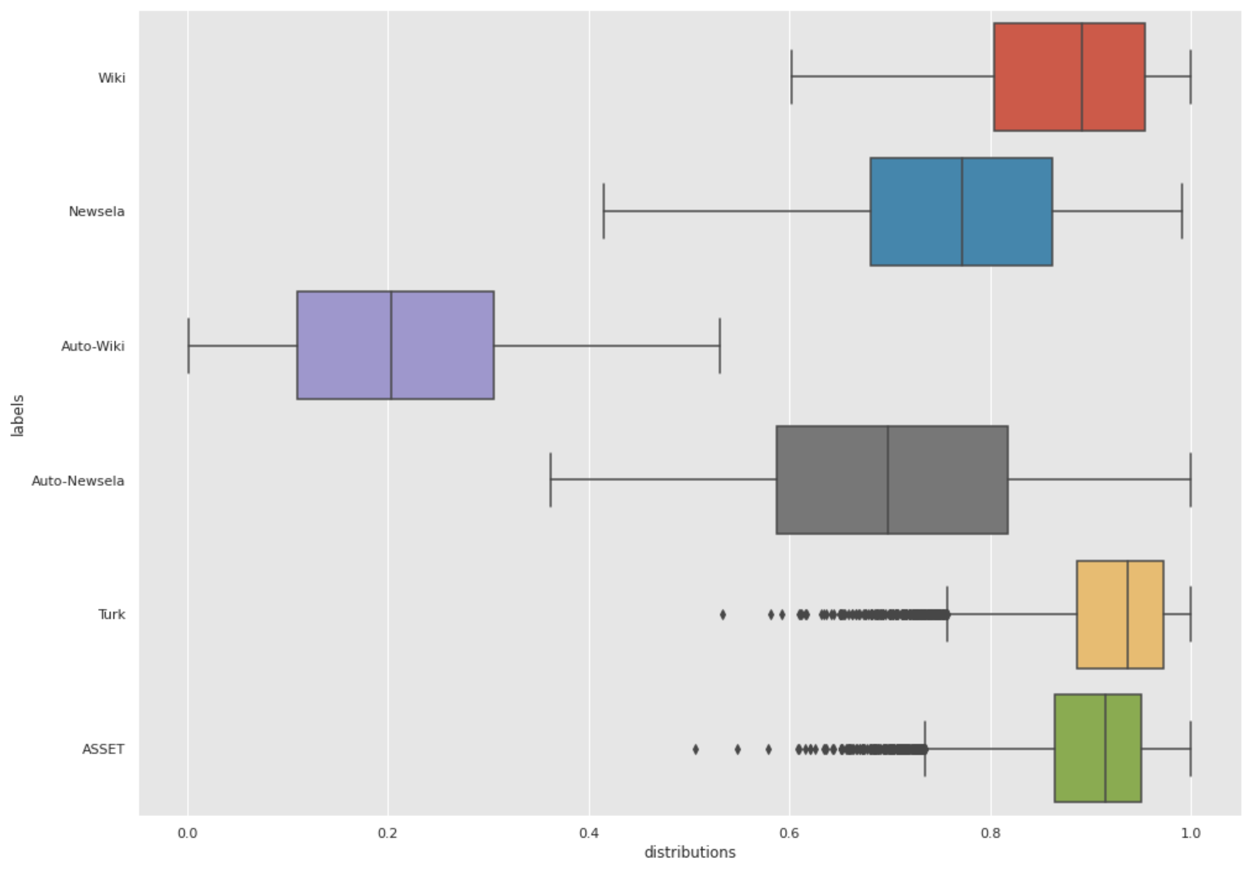


Figure 10: Relevance Analysis of Using Sentence BERT (BoxPlot)

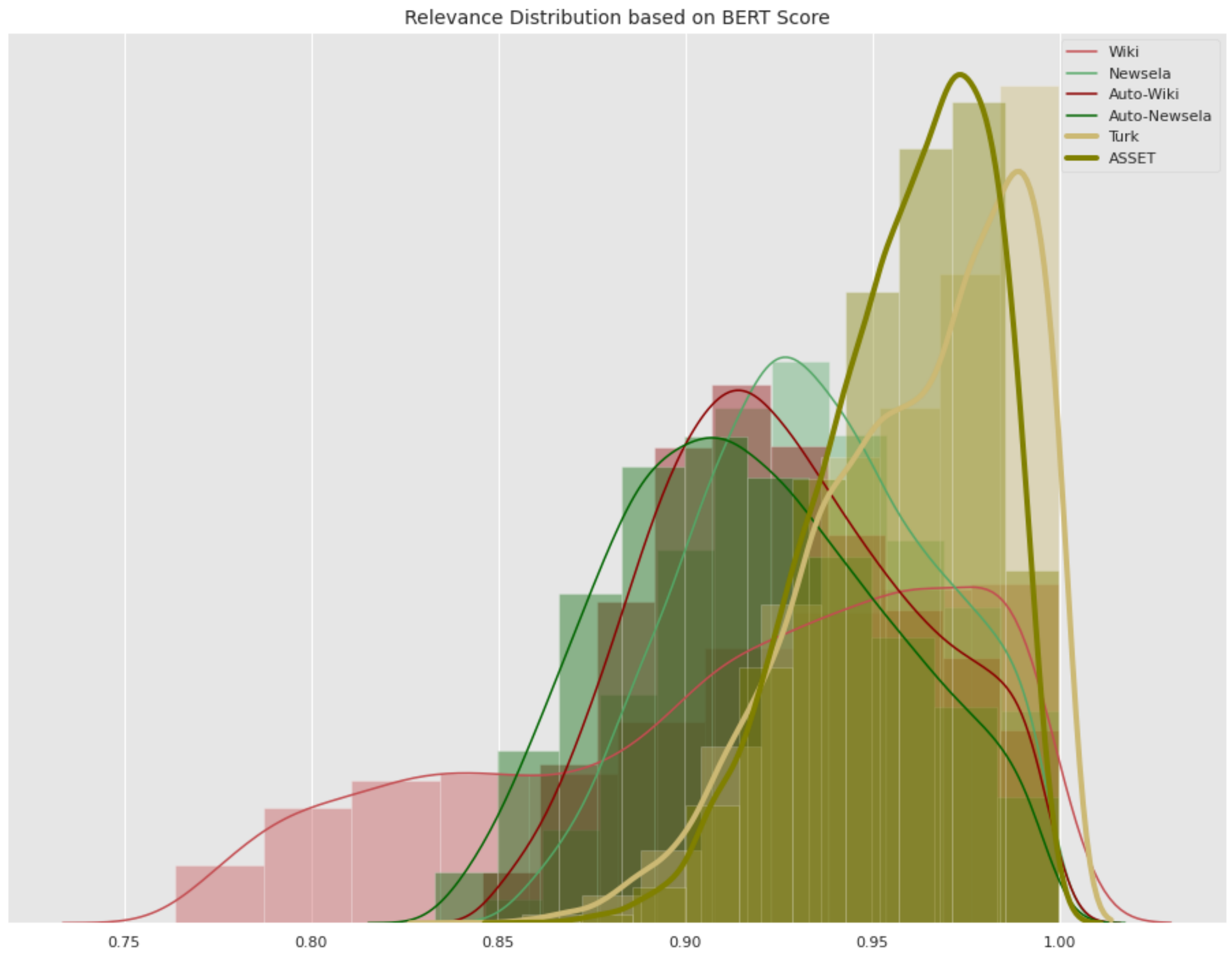


Figure 11: Relevance Analysis of Using BERT Score (Histogram)

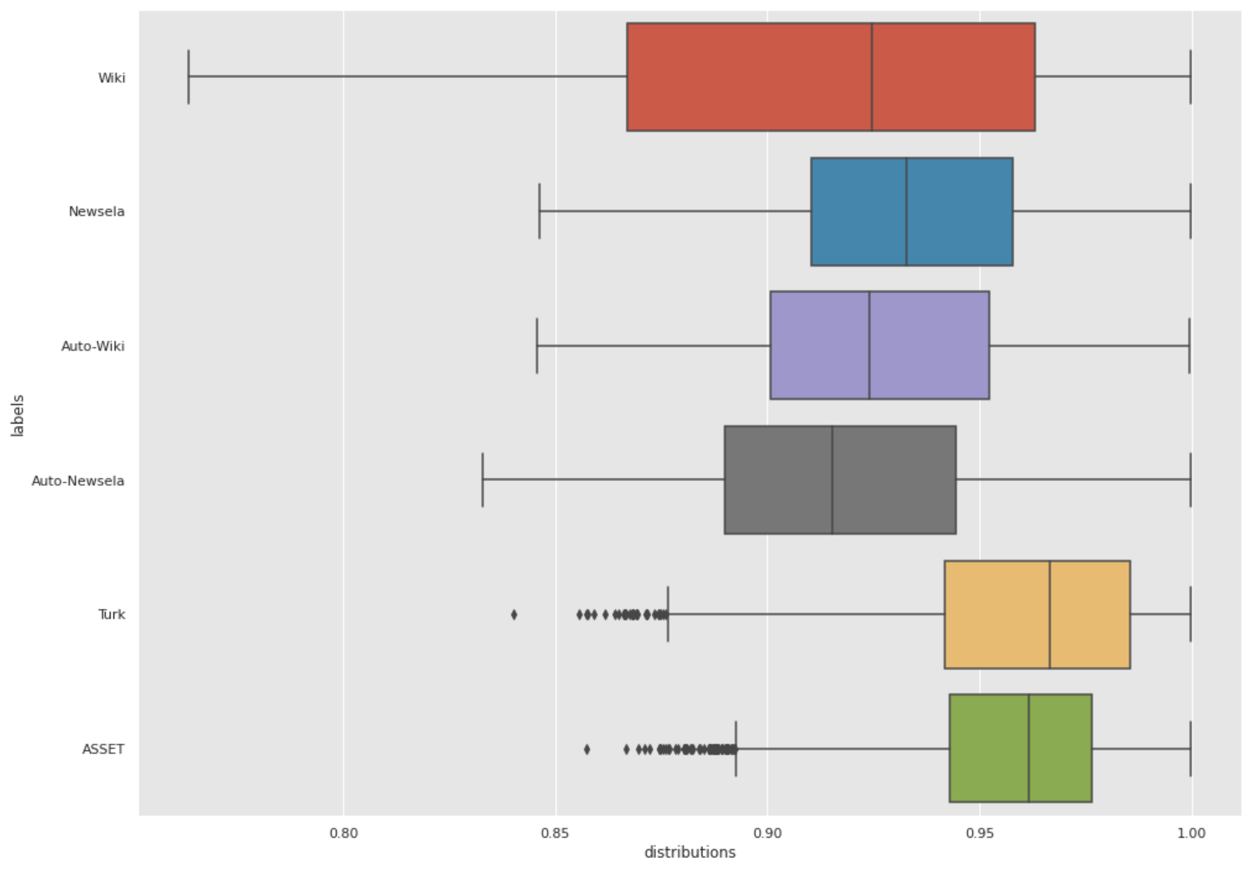


Figure 12: Relevance Analysis of Using BERT Score (BoxPlot)

ASSET contain more rewriting operations.

However, the distribution of other training corpora differs significantly. Most of the low relevance is because of less correlated sentence pairs. Tables 10 and 9 show samples of less correlated pairs from Newsela and Wikipedia, respectively. These less correlated samples confuse the model and teach the model to generate irrelevant simplified sentences. As I mentioned before, to rewrite a text in a different readability level, editors may drastically reconstruct its words, sentences, or even paragraphs. In consequence, sentences in paraphrased articles may not accurately pair with the original ones. It is difficult to extract high-quality sentence pairs like TURK and ASSET. This challenge becomes even more daunting when sentences are automatically extracted and aligned with simple features. Therefore, I believe the most important thing is to construct a corpus equipped with strongly correlated and complex simplified sentence pairs.

Table 9: Less correlated samples from Wikipedia.

Source	Sentences
Complex	Dudley went straight out of college to work as a teacher in 1880.
simplified	Dudley worked as a teacher .
Complex	La Sauve is a commune in the Gironde department in Aquitaine in south-western France .
simplified	It is found in the region Aquitaine in the Gironde department in the southwest of France .

3.2.2 Substitution Analysis

Figure 13 (Histogram) and 14 (BoxPlot) show the distribution of substitution rewriting measurement for different corpora. Higher substitution indicates more words in the complex sentence are replaced, which indicates simpler outputs. Turk and ASSET outperform most of the training corpora. The sentence pairs in the current, publicly available training corpora substitute a short amount of words, especially for the Newsela dataset. However, Newsela-Auto performs slightly better than Turk and ASSET.

Table 10: Less correlated sample from Newsela.

Source	Sentences
Complex	Retired players Dave Christian , Reed Larson and William Bennett filed a class action lawsuit in federal court on Tuesday alleging that the league has promoted fighting and downplayed the risk of head injuries that come from it .
simplified	They filed a lawsuit .
Complex	He is the executive director of georgiacarry.org , a group that pushed for the bill passage .
simplified	He is with georgiacarry.org .

3.2.3 Dropping Analysis

Figure 15 (Histogram) and 16 (BoxPlot) show dropping rewriting distribution for different corpora. A higher value indicates more words in the complex sentence are dropped. However, many dropped words remove much information, which leads to a significant information loss in the simplified sentence. A small number of dropped words indicates insufficient simplification, which is less helpful for reducing the sentence complexity. Therefore, I treat two human-written datasets as references and compare these references with other datasets. Wiki, Wiki-Auto and Newsela-Auto drop more words, and only Newsela has a similar distribution of dropped words to the referenced dataset.

3.2.4 Splitting Analysis

Table 11 shows the ratio of samples contributing to splitting simplification rewriting operation. I expect a certain amount of the sample to contribute to splitting operations so the model can learn to split a long sentence. ASSET, a dataset created by Amazon Mechanical Turk workers with certain requirements, contains 30 percent of the corpus that contributes to splitting rewriting operations, which indicates such a splitting rewriting operation cannot

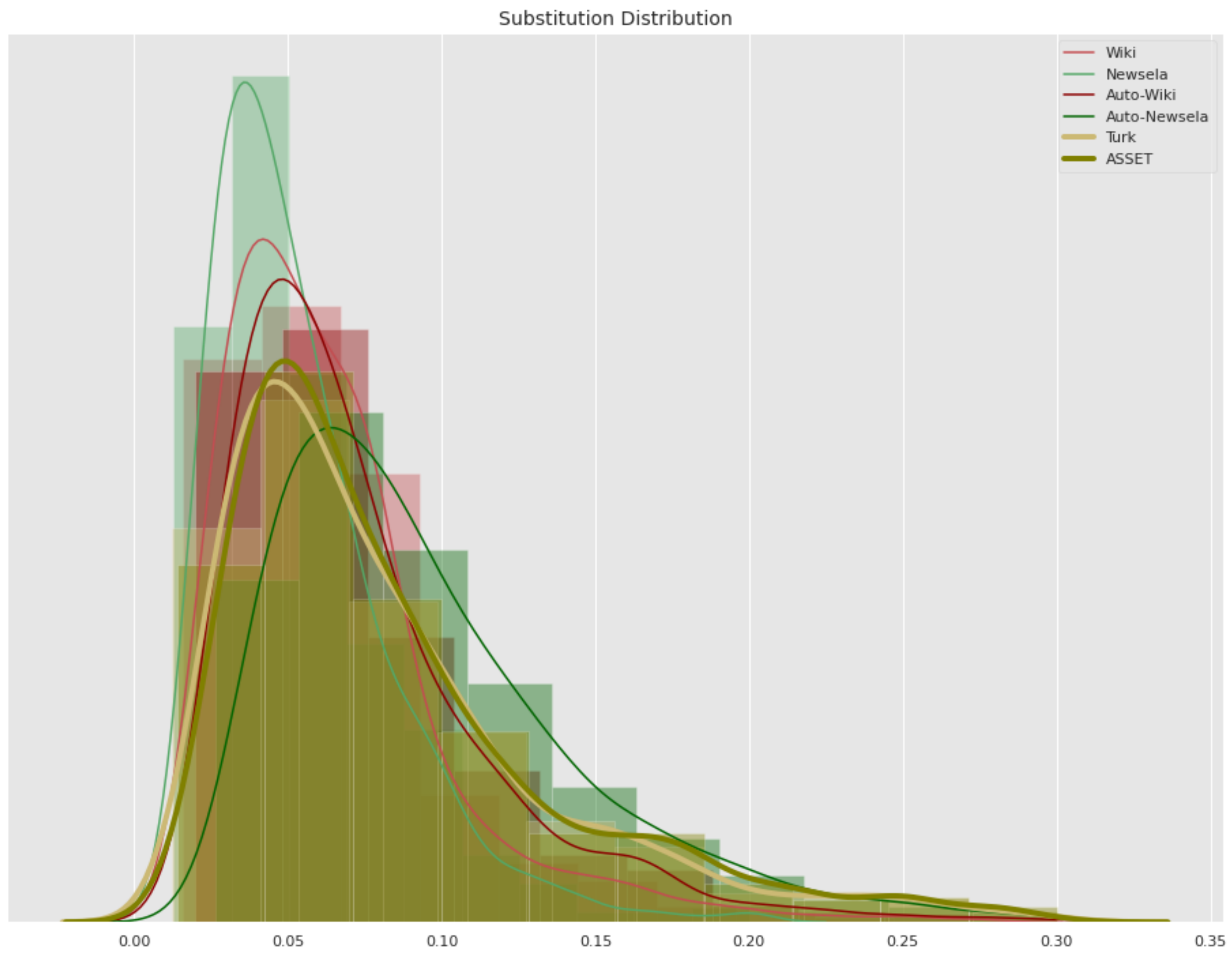


Figure 13: Substitution Analysis (Histogram)

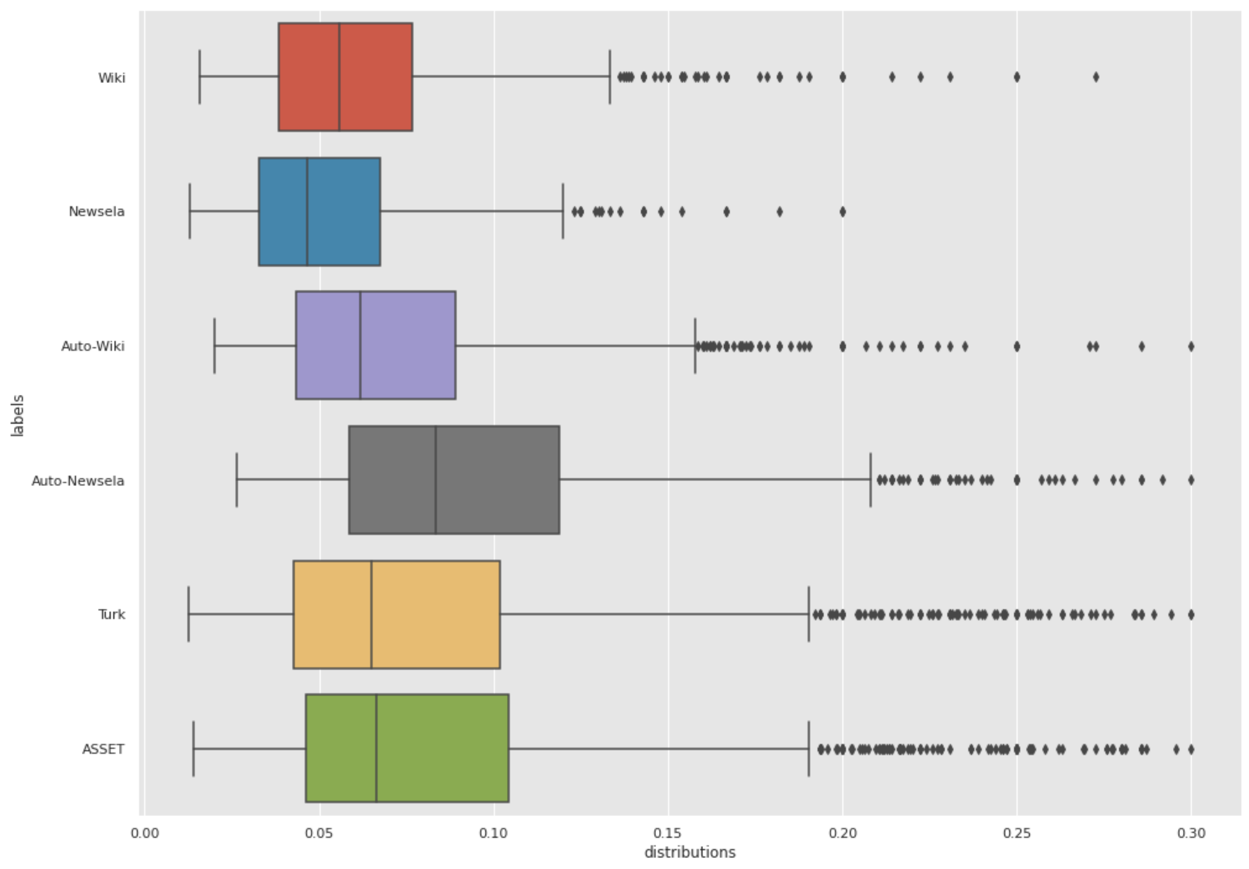


Figure 14: Substitution Analysis (BoxPlot)

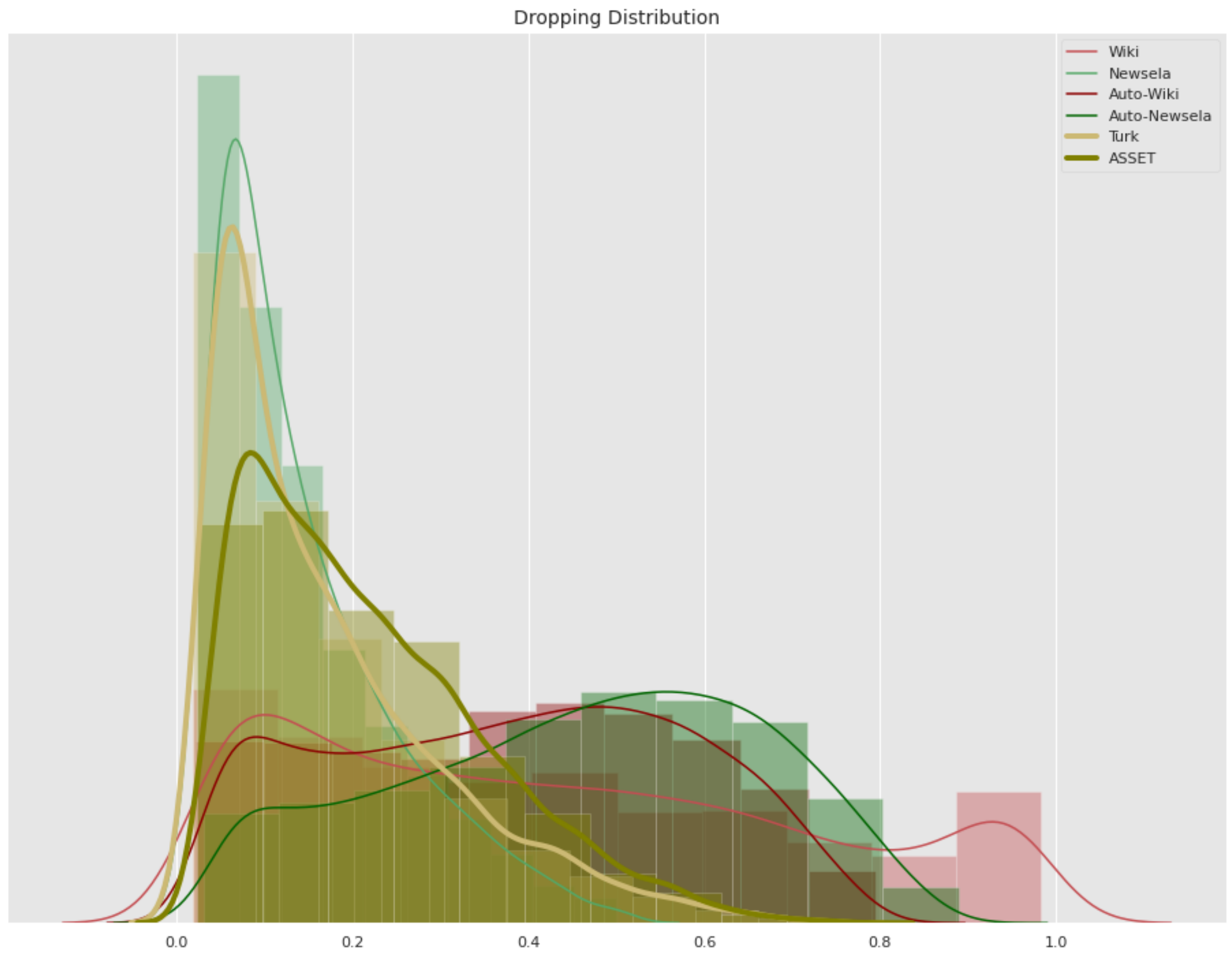


Figure 15: Dropping Analysis (Histogram)

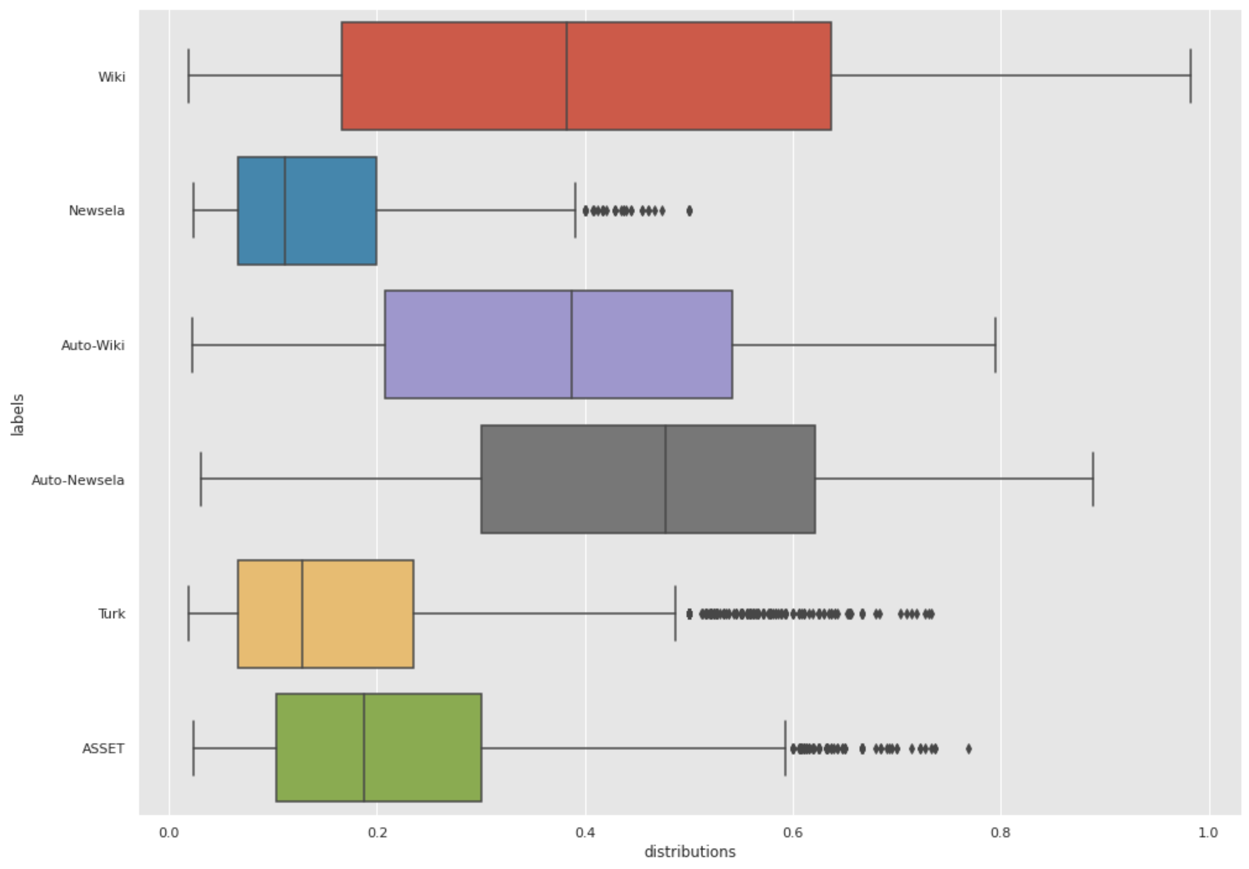


Figure 16: Dropping Analysis (BoxPlot)

Table 11: Splitting Analysis

Corpus	Split Ratio
Wikipedia	0.102
Newsela	0.002
Wiki-Auto	0.009
Newsela-Auto	0.016
Turk	0.044
ASSET	0.310

be ignored. However, not all the training corpora have enough samples that teach the model to perform splitting. [4]

3.2.5 Reordering Analysis

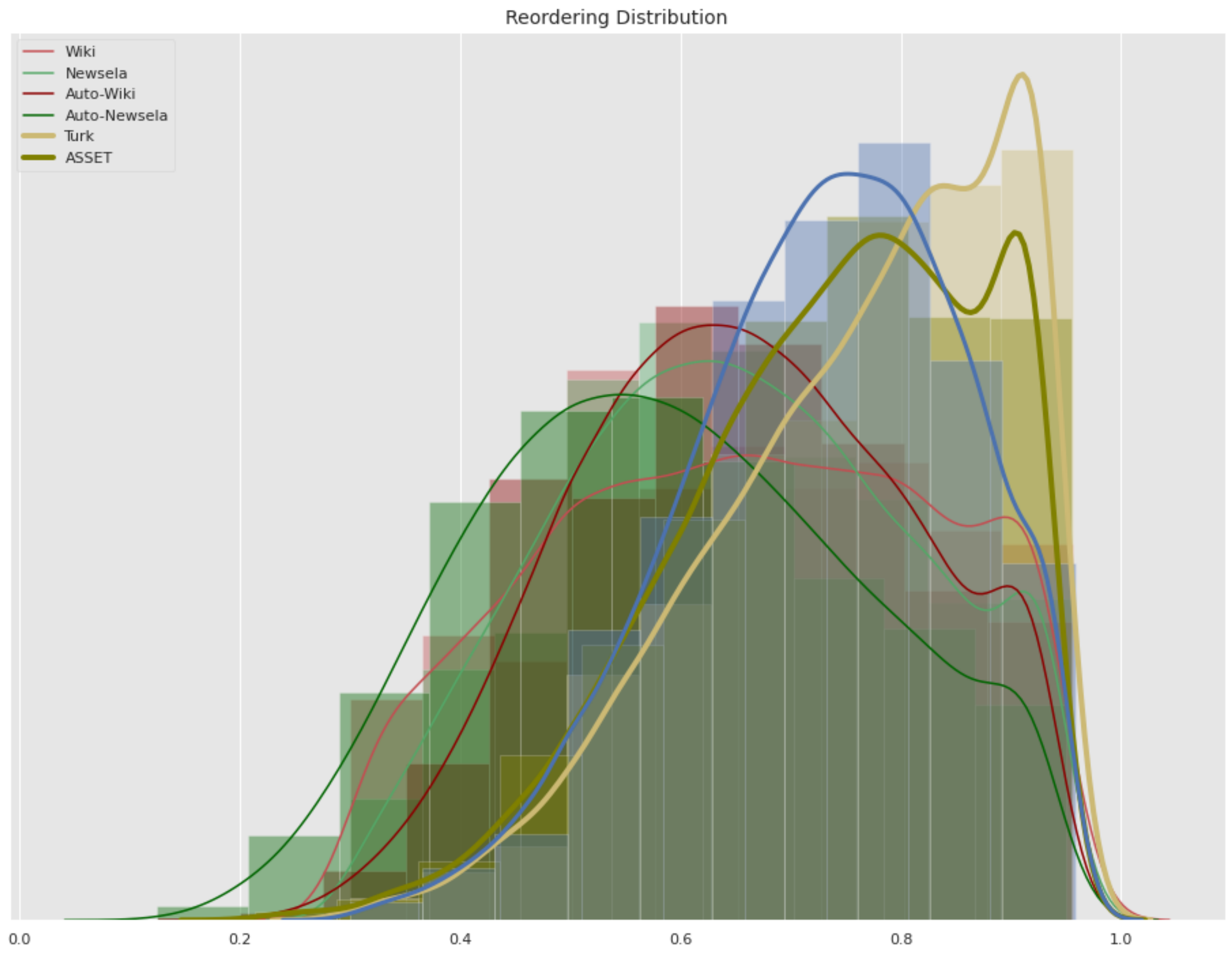


Figure 17: Reordering Analysis (Histogram)

Figure 17 (Histogram) and 18 (BoxPlot) shows distribution of reordering simplification rewriting measurements for different corpora. A lower value indicates more sentence struc-

[4]Using regular expression sentence splitter (the regular expression: $(?<!\because)(?<![A-Z][a-z]\dot{)}(?<=\dot{|}))$ may be too strict to split sentences. I found the ratio of split sentences are lower than the result from user study conducted by [33] based on a small subset of data. But it is impossible for me to conduct user studies for all training datasets.

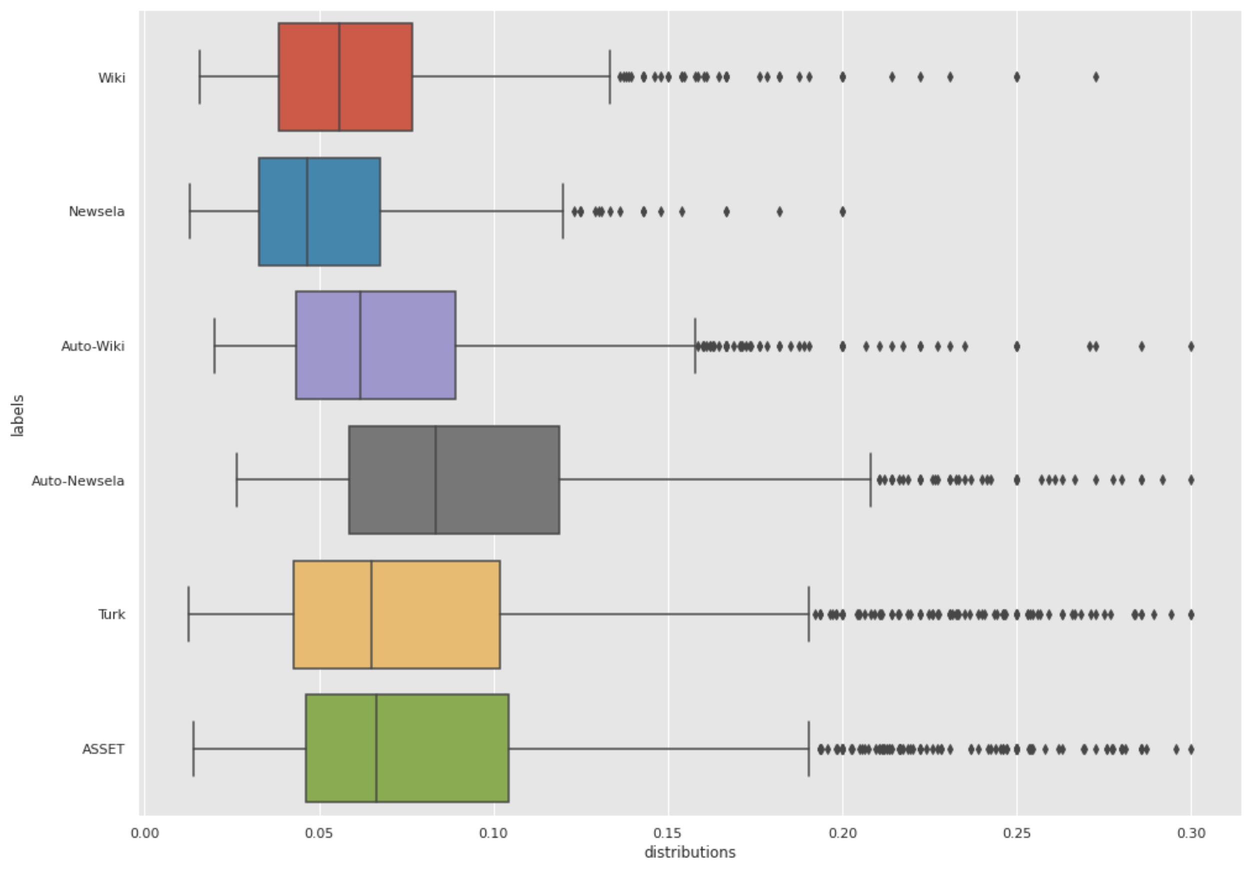


Figure 18: Reordering Analysis (BoxPlot)

ture changes. Like the analysis of dropping operation, a large amount of sentence structure change leads to a noisy or less correlated simplified sentence. A small amount of sentence structure change indicates insufficient simplification, which is less helpful for reducing the sentence complexity.

I treat two human-written datasets as references and compare these references with other datasets. The training corpora, Wiki, Newsela, Wiki-Auto, and Newsela-Auto, differ considerably from Turk and ASSET.

3.2.6 Summary

Based on the above analysis, I summarize three important aspects of the findings regarding the current publicly available training corpora, Wiki, Newsela, Wiki-Auto, and Newsela-Auto, by comparing them with two human-generated corpora, Turk and ASSET.

1. All training corpora contains a certain amount of poorly aligned sentence pairs. These poorly aligned sentence pairs result from either poor algorithmic extraction from parallel articles or articles that are not very parallel. Although Wiki-Auto and Newsela-Auto are equipped with an advanced alignment algorithm that uses multiple NLP technologies and has trained a large amount of data, such a problem is tough to avoid if extracting sentence pairs from existing parallel articles. This work prompts me to investigate constructing a training dataset on the sentence level and avoid extracting sentence pairs from the available articles. In this way, I can relieve the problem of poorly aligned sentence pairs from extraction algorithms and poor parallel articles.
2. All training corpora are limited in several ways and contribute to only some operations. Infrequent cases are always treated as noise if they are merely trained using sentence pairs. As a result, a model trained on these resources favors those operations that lead to inadequate simplification. At the same time, the splitting operation is largely ignored by all the training corpora. A model trained on these resources may completely ignore splitting. This work prompts me to investigate different approaches, either by generating a training dataset that contributes to those operations or relying on existing linguistic and simplification resources, to improve different operations in the sentence simplification

systems.

3. Excessively focusing on one operation may not be of interest to some users. The different operation contribution rates in these two datasets lead to different styles of simplified sentences. Users in different settings may prefer various forms of simplified sentences. This work prompts me to investigate an approach that allows for the insertion of a style-related parameter to enable different styles of sentence simplification.

4.0 Constructing a new Corpus: `SimSim`

4.1 Generating Sentence Pairs

All the training corpora are limited in several ways and contribute to only some operations. Infrequent cases are always treated as noise if they are merely trained using sentence pairs. As a result, a model trained on these resources favors those operations, leading to inadequate simplification. Furthermore, discrepancies between training and testing data may cause models to generalize poorly.

Therefore, my first goal is to create a training corpus automatically and focus on the four simplification operations. Generating pairs of sentences is a prerequisite. **The first demand of such pairs of sentences is semantically related.**

The crux of training a well-generalizing simplification model is to teach it the essential skills for simplifying sentences. Thus, I propose to refine existing training pairs and construct novel ones by simulating various simplification transformations. I can use the current paraphrase system to work toward this goal. One of the most reliable systems is the large-scale machine translation system.

4.1.1 Seed Sentence Bank

Before performing back-translation, I build a collection of seed sentences, based on which I will perform a series of simulations and build new training pairs for simplification. To compare the existing training corpora, I re-use the sentences in Wikipedia and Newsela and treat them in my seed sentence bank. More specifically, I used complex sentences from Wikipedia and Newsela. Besides, I found some simplified sentences are still useful in Simple Wikipedia, I also incorporated a subset of simplified sentences in Wikipedia if their correlation score is higher than a threshold. In the following steps, all the simulations will be based on the seed sentence bank.

4.1.2 Back-Translation

Due to various reasons discussed in the previous chapter, it is difficult to extract complex-simplified sentence pairs from existing online resources to meet the requirements of strong semantic relevance. Instead, the existing powerful paraphrase models provide an alternative way to generate sentence pairs. In this section, I use the back-translation approach to generate sentence pairs that mitigate the problems with current training corpora.

Back-translation uses a translation system to translate the non-English side of a parallel text to obtain English-English paraphrase pairs [67, 69], and I follow a similar approach. I use multiple languages as a pivot and translate a given complex sentence into a sentence with a pivot language. I then translate that sentence back into English using the same translation system. I highlight three benefits of adopting back-translation for generating paraphrased sentences:

1. Given results of prior works [86], back-translation can ensure a basic level of alignment between sentences;
2. Because back-translation will impose certain language changes to the original text, I can collect a diverse set of candidate sentences by using different systems and pivot languages;
3. I can scale the size of the training dataset by applying this method on large text corpora, even in different languages. It imposes language diversity during translation, and trivial details may change, but the gist remains untouched.

To support more diverse outputs, in which more diverse outputs have higher probability and support more simplification rewriting operations, I select GNMT (Google's Neural Machine Translation System) as the translation system because (1) it is free and high quality, and (2) it supports 103 languages. Theoretically, given one complex sentence, back-translation via GNMT provides 103 high-quality, semantically related, paraphrased sentences.

I use GNMT to translate each sentence into 103 pivot languages, then translate them back to English. By doing this, I collect a pool of sentence pairs. One sample is shown in Table 12. Given the original sentence,

It is situated at the coast of the Baltic sea, where it encloses the city of Stralsund.

Back-translation via both Chinese and Greek as pivot languages provides the simplest outputs. Output through Chinese as the pivot language simplifies the preposition and output via Greek as the pivot language simplifies the words “situated” and “’coast.” The other outputs are sub-optimal; the output using Italian as pivot language generates a less fluent sentence.

Table 12: Sample from GNMT.

Pivot	Sentences
Original	It is situated at the coast of the Baltic sea , where it encloses the city of stralsund .
Chinese	It is located on the coast of the Baltic Sea and surrounds the city of Stralsund .
Greek	It is located on the shores of the Baltic Sea, where it encloses the city of Stralsund .
Italy	It is located on the Baltic Sea coast, where the city of Stralsund is located .
Japanese	It is located on the Baltic Sea coast and surrounds the city of Stralsund .
Hindi	It is situated on the banks of the Baltic sea, where it surrounds the town of Stralsund .

As shown above, the pool of paired sentences does not necessarily consist of complex-simplified sentence pairs. Translation requires conveying a complete semantic sense of a given sentence via another language; such a process does not guarantee the translated sentence is a simplified sentence. To make sure the sentence pairs are helpful to improve readability, I focus on selecting sentence pairs using Generative Pre-trained Transformer 2 (GPT-2).

4.1.3 Selecting Sentence Pairs Using GPT-2

After collecting the pool of sentence pairs, I use the unconditional language model tool to verify the sentence pairs are complex-simplified sentence pairs. The unconditional language model is trained in an unsupervised approach by feeding a huge amount of text data. In this way, similar to GNMT, the model gives a higher score to the most commonly-used sentences, and such commonly-used sentences are always simpler. Such an unconditional language model is more effective than the GNMT because the unconditional language mode does not have a constraint to convey the complete semantic sense of a given sentence compared to GNMT.

More specifically, I use GPT-2, a large-scale unconditional language model that can generate coherent text paragraphs, achieve the most advanced performance on many language modeling benchmarks, and perform basic reading comprehension, machine translation, question answering, and summary operations without task-specific training. Besides the data strategy, GPT-2 is a large, transformer-based unconditional language model with 1558 million parameters. The GPT-2 was trained simply to predict the next word in a huge amount of text from the Internet. As shown in Figure 19, GPT-2 appends a special token "BOS" at the beginning of the sentence and uses this sentence as the input of the GPT-2 model. The output is the original sentence. In this way, the input and output sentences are shifted by one token. For each token, GPT-2 predicts the next token giving all previous tokens. Figure 19 shows the prediction of the last three tokens of the sentence "Admission to Tsinghua is competitive." When GPT-2 learns to predict each token in the sentence, it observes the information of previous tokens. For example, when predicting the word "competitive," the GPT-2 observes the previous tokens "admission to Tsinghua is" and learns to predict the next token.

Regarding the dataset that GPT-2 used, human filter the dataset choices to emphasize the diversity of content and preserve the text quality. For example, they only considered a web page if the person who recommended it on Reddit had at least three karma. Such a data filtering metric can be thought of as a heuristic indicator for data quality. In this way, users can be certain that the data (1) is the focus of a diverse domain of content, (2) fit most user interests and (3) is of higher data quality.

Such a large model trained with a high-quality training dataset will be able to capture diverse genres of text.

To quantitatively validate the level of the sentence is commonly-seen, I average the negative log-likelihood loss of predicting each token in the sentence and call it the "GPT-2 loss." A lower loss value indicates the sentence is commonly seen, whereas a higher loss indicates a rarely seen sentence.

Table 13 shows the loss for the sentences listed in Table 12. Simplified sentences always achieved a lower GPT-2 loss.

Therefore, I use GPT-2 to validate the sentence pairs as complex-simplified pairs. More specifically, if the loss of the paraphrased (back-translated) sentence is smaller than the original sentence, I treat it as complex-simplified sentence pairs. For the pool of paired sentences, I filter out the pairs with a larger GPT-2 loss for the paraphrased (back-translation) sentence.

Figure 20 shows a sample of GPT loss per token for sentences "It is situated at the coast of the Baltic sea, where it encloses the city of Stralsund" and "It is located on the Baltic Sea coast and surrounds the city of Stralsund". The GPT-2 losses of the first two tokens are the same for the two sentences; however, the GPT-2 loss of the token "located" is much smaller than that of the token "situated." The different tokens also affect the GPT-2 loss of the following tokens. For example, the GPT-2 loss of "coast" in the second sentence is smaller than that in the first sentence. It may be because the phrase "located on the coast" is more frequent than "situated at the coast."

Among all 103 candidate languages, I select the one with the lowest GPT-2 score as the target sentence to form a training pair. Particularly, if GPT-2 score deems the back-translated sentence is less natural than the original one, it will be discarded. In this way, I construct the first version training corpus and call it `SimSim`, for "simulate simplification." Though the sentence pairs constructed in the previous section are well-aligned, they can hardly be used for training a simplification model because all essential simplification operations are not distilled in the data. I perform the following steps to simulate each simplification operation individually and to distill those simplification operations into our data.

....

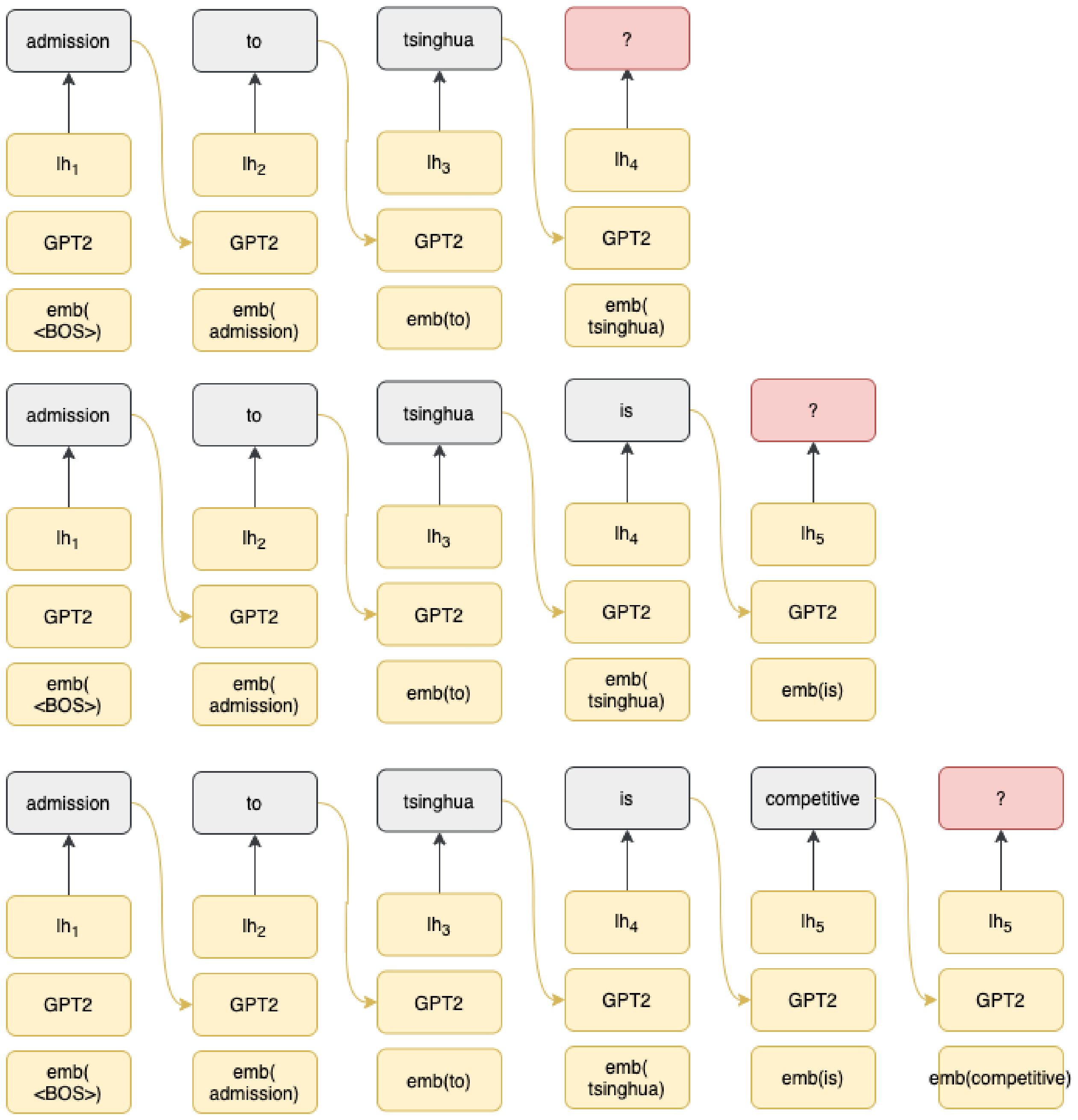


Figure 19: GPT-2 Prediction

Table 13: Sample from GNMT.

Pivot	Sentences	GPT-2 Loss
Original	It is situated at the coast of the Baltic sea , where it encloses the city of stralsund .	3.8020
Chinese	It is located on the coast of the Baltic Sea and surrounds the city of Stralsund .	2.8642
Greek	It is located on the shores of the Baltic Sea, where it encloses the city of Stralsund .	2.8379
Italy	It is located on the Baltic Sea coast, where the city of Stralsund is located .	2.9493
Japanese	It is located on the Baltic Sea coast and surrounds the city of Stralsund .	3.1864
Hindi	It is situated on the banks of the Baltic sea, where it surrounds the town of Stralsund .	3.0487

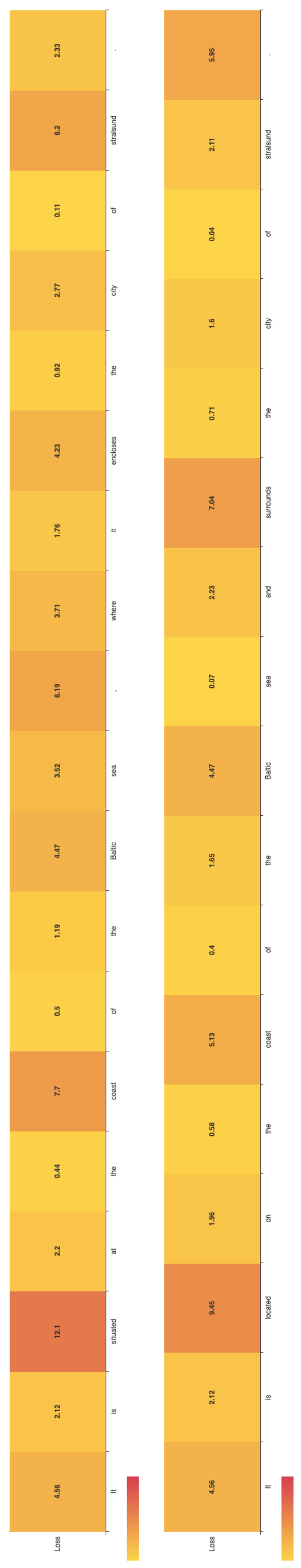


Figure 20: GPT-2 Loss for sample sentences.

4.2 Simulating Simplification Operations

The previous pipeline reduces the misalignment error and makes sure the sentence pairs in my dataset are strongly semantically correlated, then **the second demand of such pairs of sentences are simulating four simplification rewriting operations.**

4.2.1 Simulating Substitution and Reordering

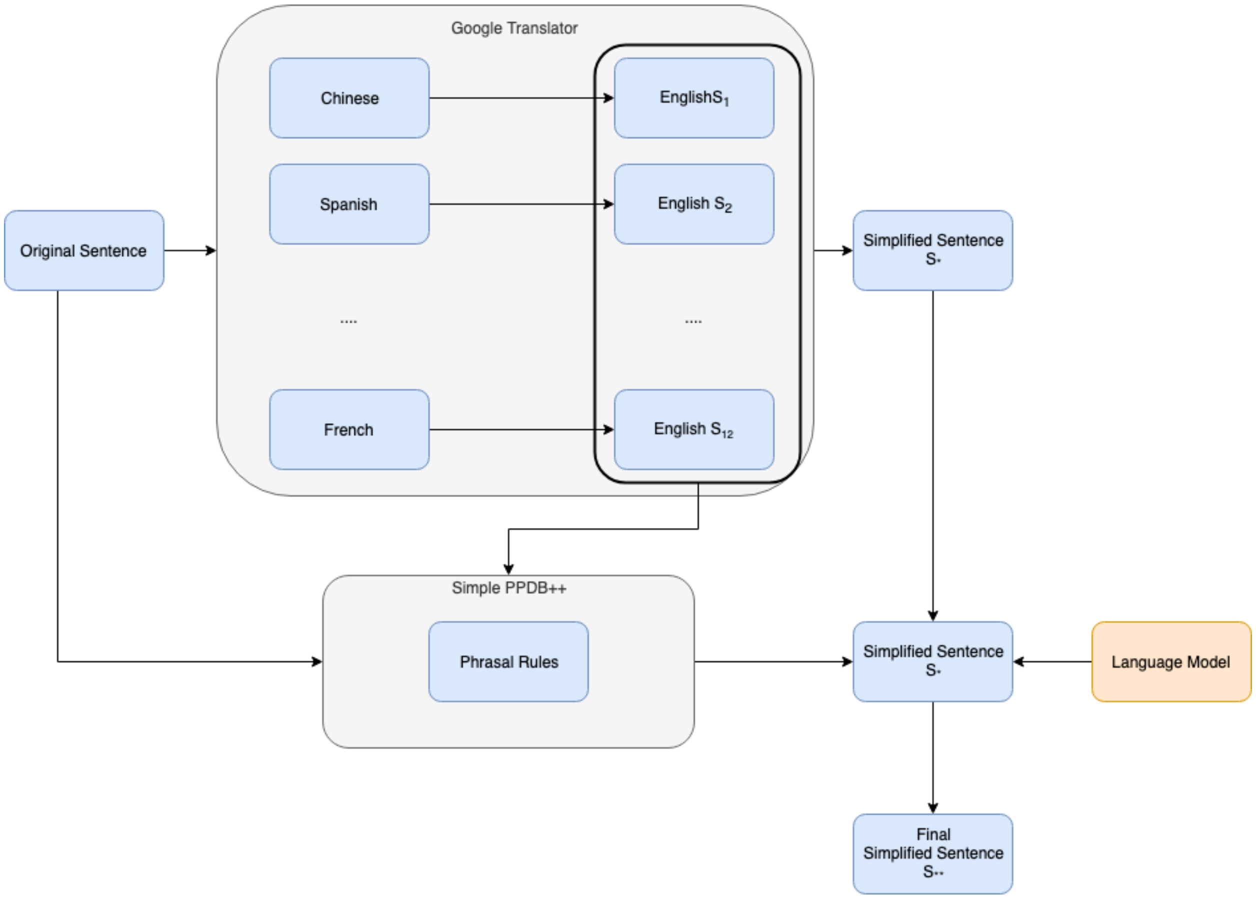


Figure 21: GPT-2 Loss for sample sentences.

I propose to apply the paraphrasing rules from the PPDB to simulate the substitutions and impose substitution transformations into the paraphrased pairs. However, simply replacing words that appear in the PPDB is not practical because substitution operations should be context-dependent. Meanwhile, the number of options for replacing each token is huge, and most of them are not practical for the current sentence.

To reduce the number of options of applied rules, I filter out the rules not used in the 103 paraphrased pairs and only focus on a subset of rules as candidates. For example, table 3 shows "situated" can be replaced by three candidate words "located," "implanted," and "stationed" from the PPDB. The three options need to be applied based on the context sentence containing the word "situated." Given the paraphrased pairs from back-translations as shown in Table 13, I only find the token "located" is presented in the paraphrased sentences. Therefore, I filter out the other two options, "implanted" and "stationed," and focus on the replaced target "located" for the token "situated." As shown in Figure 21, my pipeline first extracts the paraphrased sentence S* with the lowest score as the target sentence. Then I apply all the processed PPDB rules to the target sentence S* to form S**. To ensure the applied rules are proper, I use GPT-2 again to evaluate the quality of sentences after applying rules. For example, I apply the rule **coast** $\rightarrow$ **shore** to the sentence "It is located on the **coast** of the Baltic Sea and surrounds the city of Stralsund." Then, I compare the GPT-2 scores of the original sentence and replaced the sentence ("It is located on the **shore** of the Baltic Sea and surrounds the city of Stralsund"). If the GPT-2 score improves after the substitution, I will accept this transformation and form a new data pair. In this way, I simulate substitution rewriting operation, and this pipeline improves the substitution operation for the SimSim.

4.2.2 Simulating Dropping

To distill the dropping operation into the data, I follow the previous approach [41] and augment the data by randomly removing certain functional phrases from the simplified sentence. To study the Turk and ASSET validation dataset, I find that prepositional, adjective or adverb phrases are the most common phrases that are dropped. This dropped behavior does not significantly change the meaning of the original sentence. Therefore, I randomly removed these phrases during training, and I expect the model can learn to drop words appropriately.

4.2.3 Simulating Splitting

I find back-translation rarely paraphrases a sentence by splitting it into multiple shorter ones. I think that proper splitting transformation can help improve the readability by reconstructing the structure of a sentence. To incorporate the splitting operation into our data, I include Wikisplit [4] in our dataset. The Wikisplit dataset is constructed automatically based on the Wikipedia revision history, and it can serve as a useful resource for models to learn the splitting operation. I put Wikisplit sentence pairs in the seed bank and apply the above process on the target-side sentences to mix the splitting transformation with other operations.

4.3 Analysis of SimSim

By simulating different operations with the above process, I present the second version of SimSim for text simplification. In total, it contains 1676k complex-simplified sentence pairs, which is significantly larger than previous datasets. In this section, I analyze SimSim and compare all the measurements in the previous chapters to the existing training datasets.

4.3.1 Relevance Analysis

Figure 22, 24 and 26 (Histogram) and 23, 25 and 27 (BoxPlot) show the distribution of relevance rewriting operation for all the previous training corpora including SimSim. SimSim achieves the highest relevance score, especially for the relevance score computed by BERT Score. This is mostly because SimSim is constructed by back-translation, which avoids misalignment error from extraction and alignment algorithms and poorly aligned articles. The back-translation approach significantly improves relevance measurement.

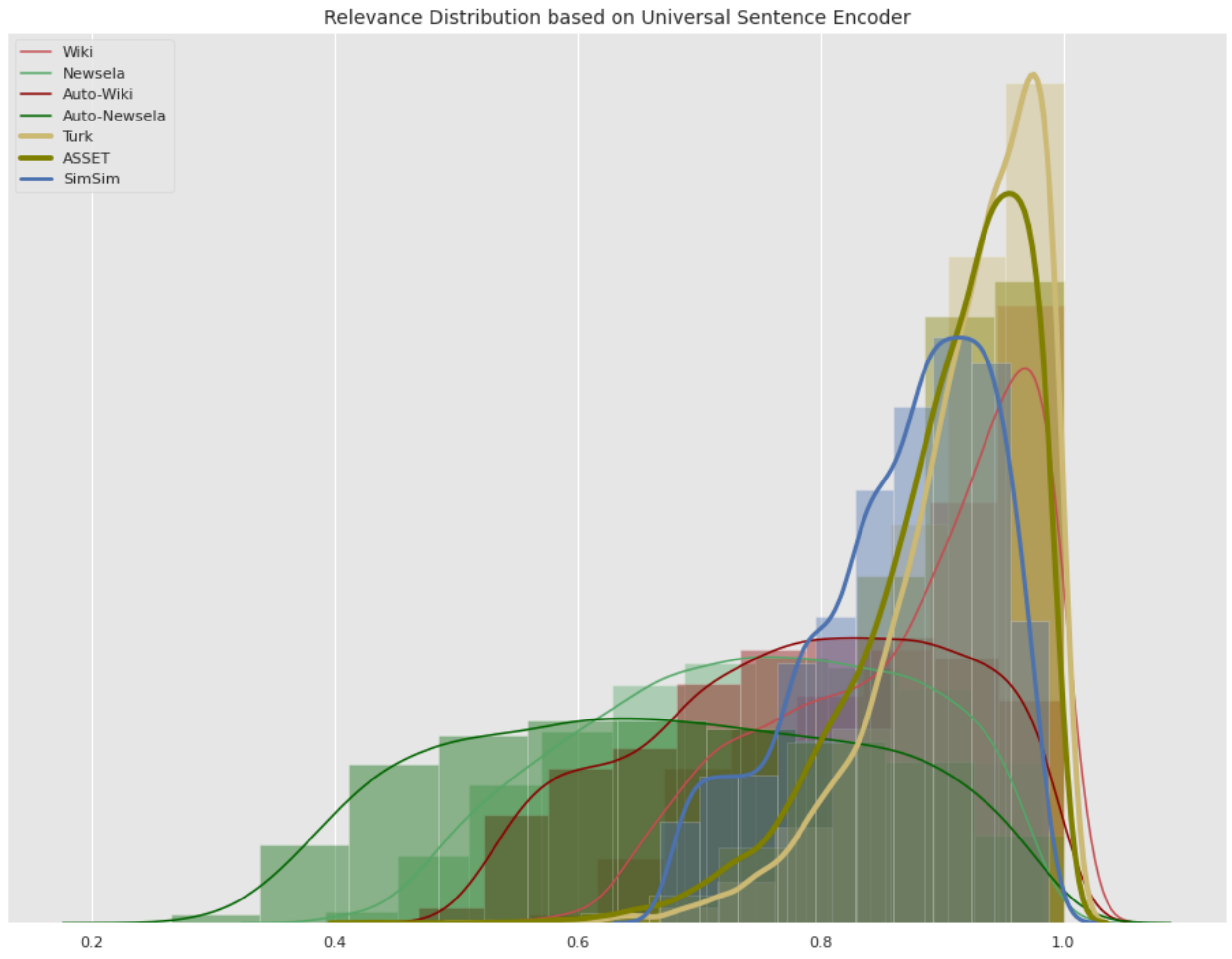


Figure 22: Relevance Analysis of Using Universal Sentence Encoder (Histogram)

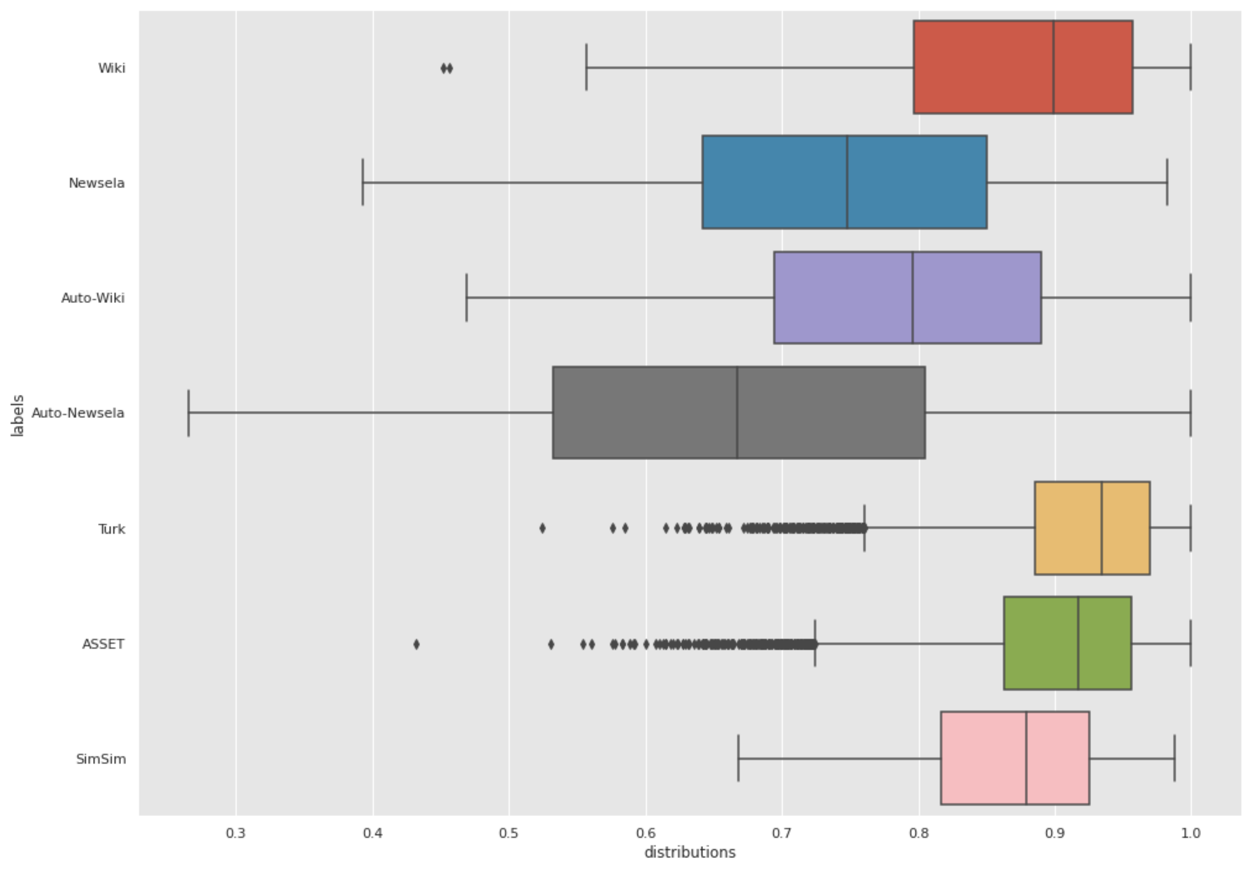


Figure 23: Relevance Analysis of Using Universal Sentence Encoder (BoxPlot)

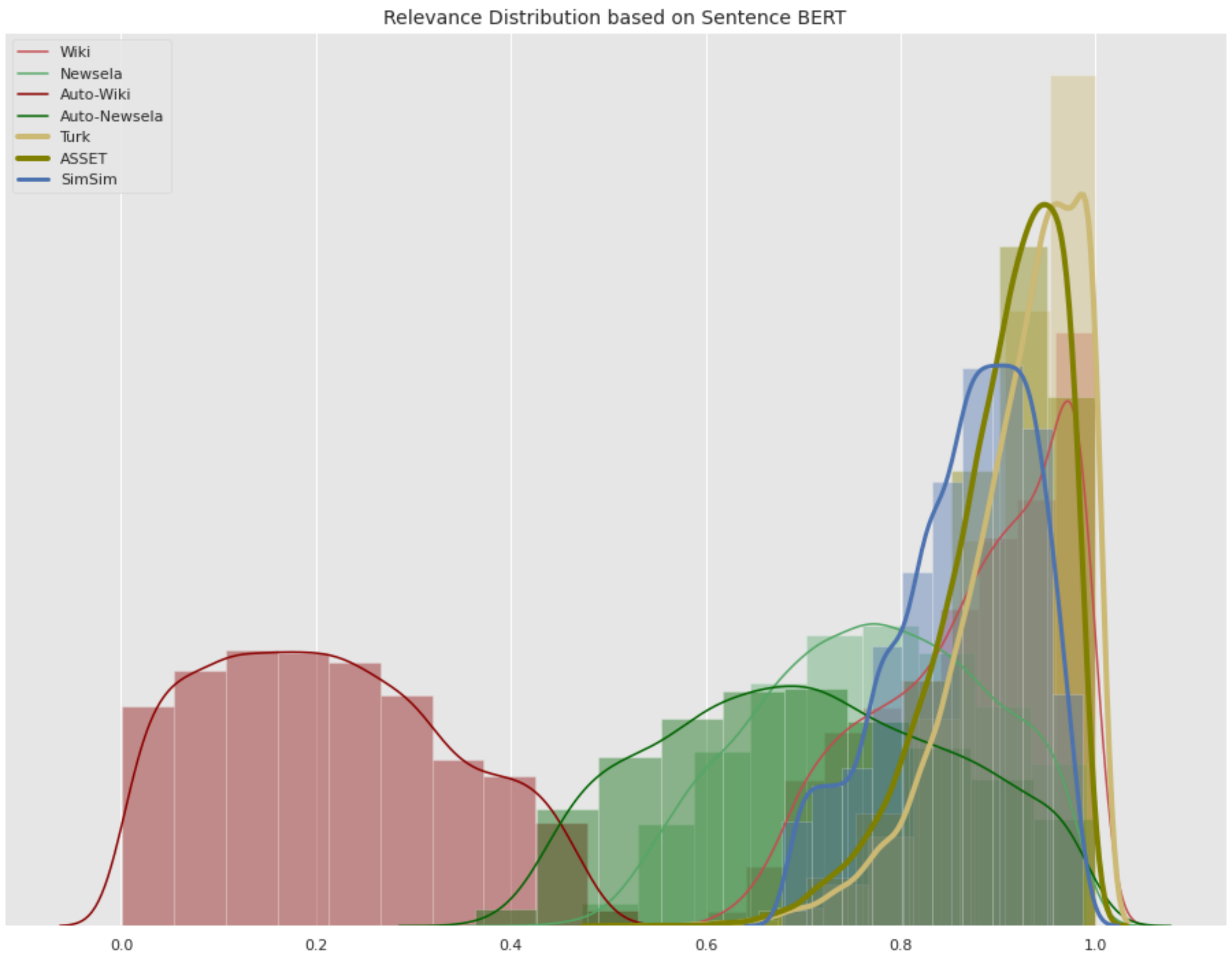


Figure 24: Relevance Analysis of Using Sentence BERT (Histogram)

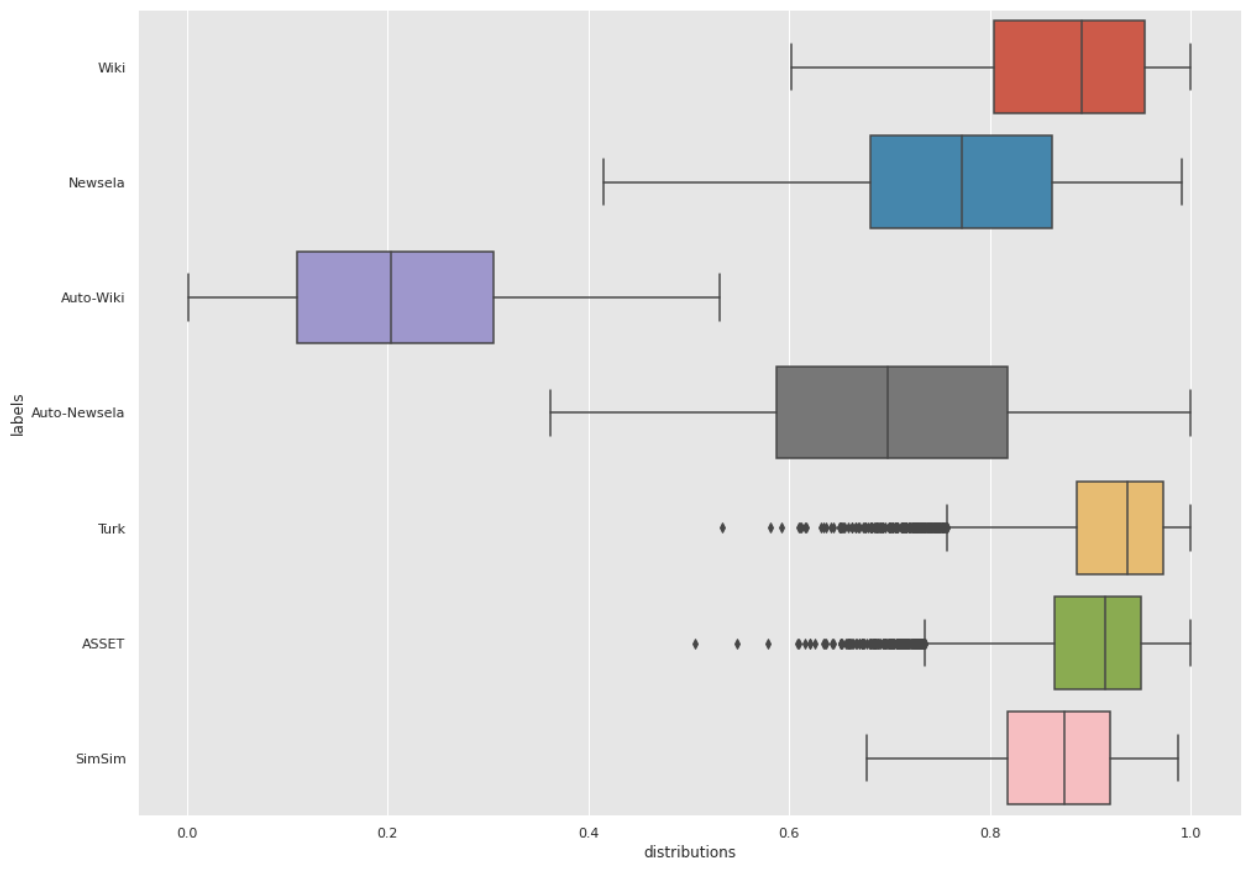


Figure 25: Relevance Analysis of Using Sentence BERT (BoxPlot)

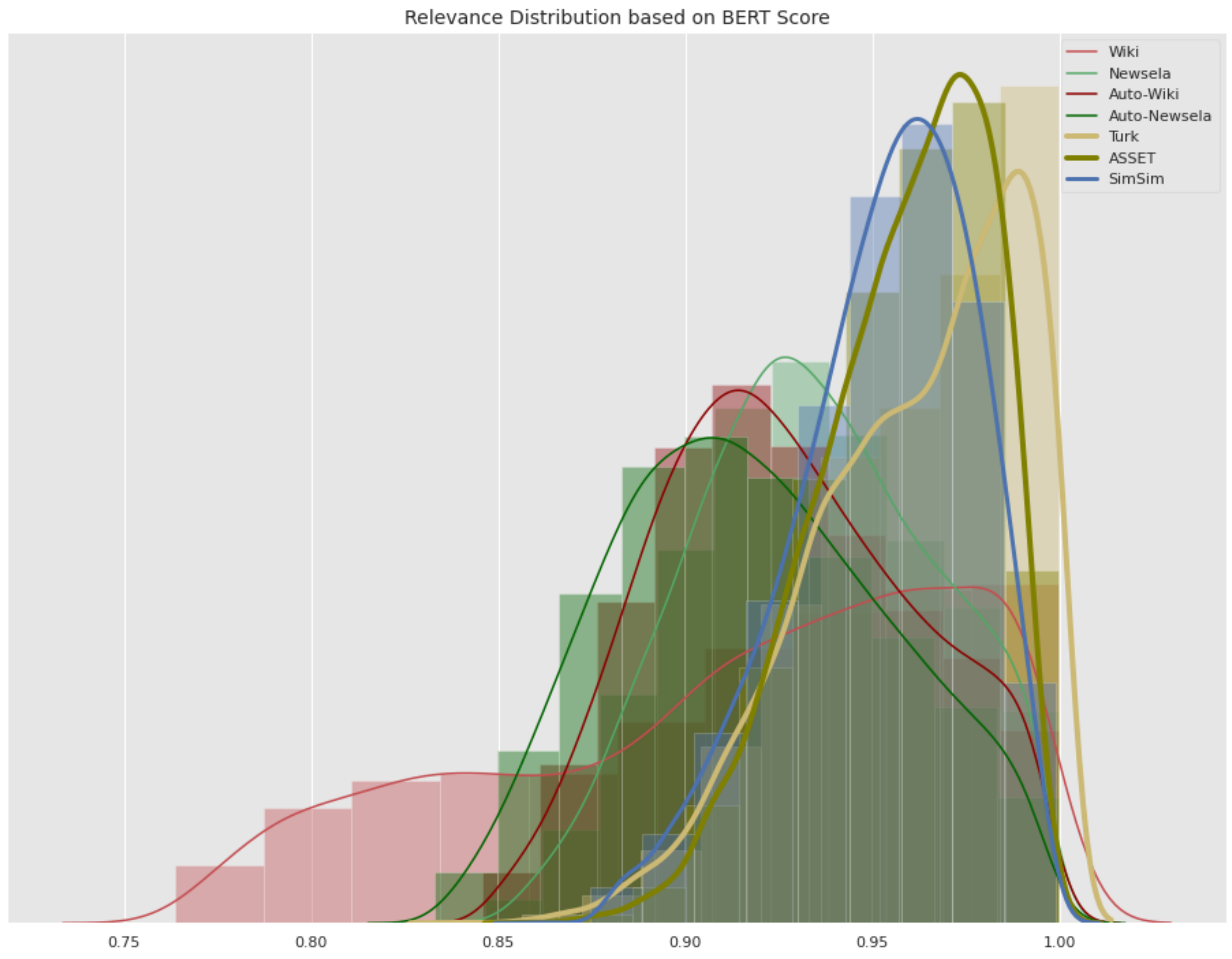


Figure 26: Relevance Analysis of Using BERT Score (Histogram)

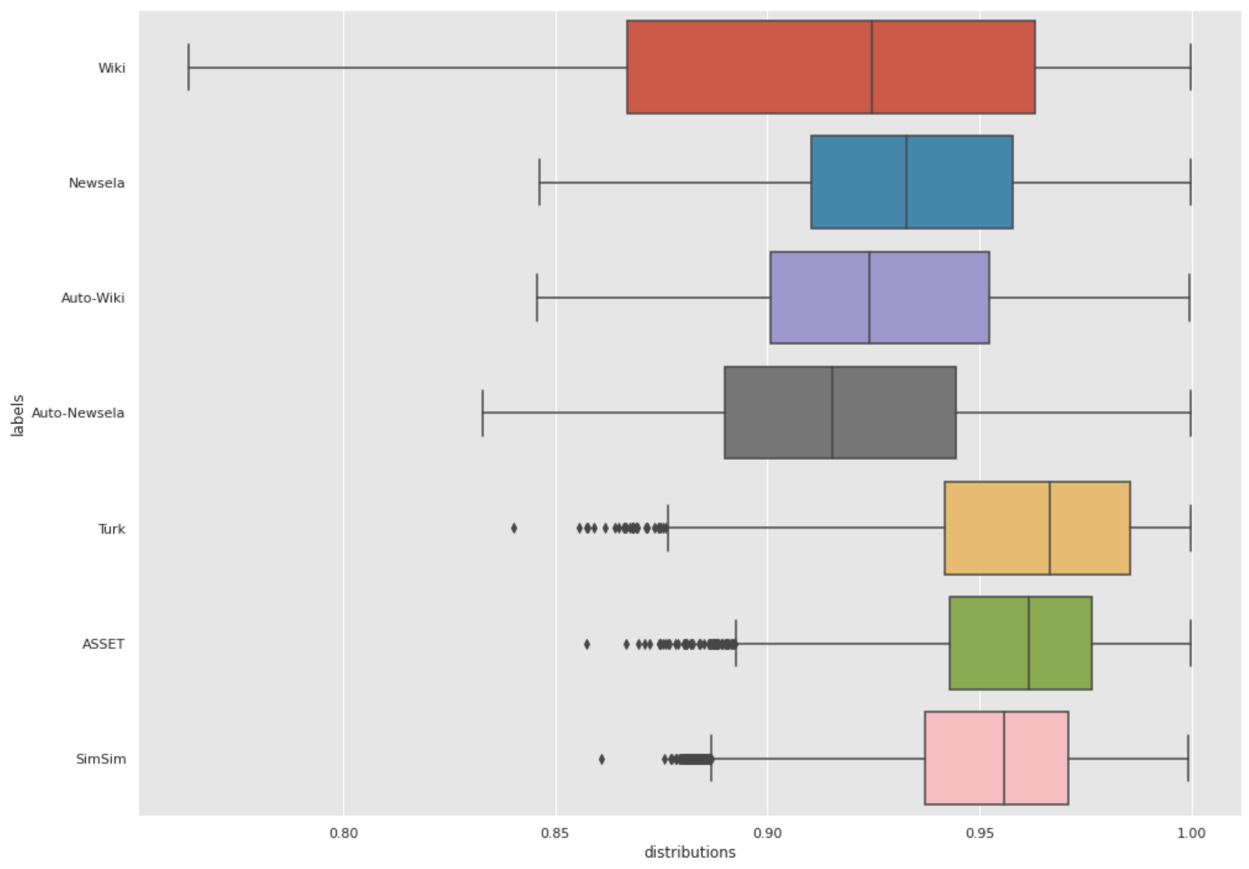


Figure 27: Relevance Analysis of Using BERT Score (BoxPlot)

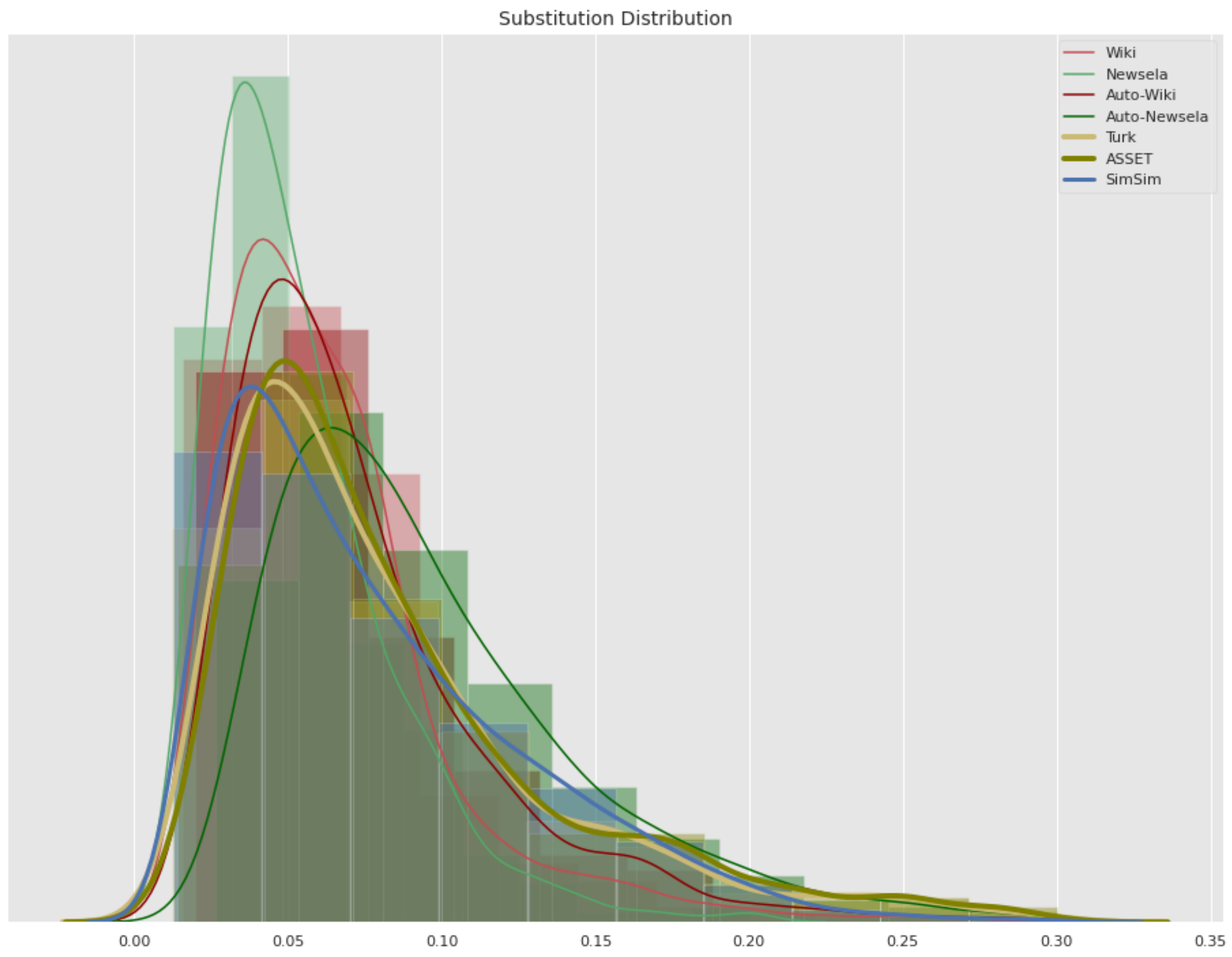


Figure 28: Substitution Analysis (Histogram)

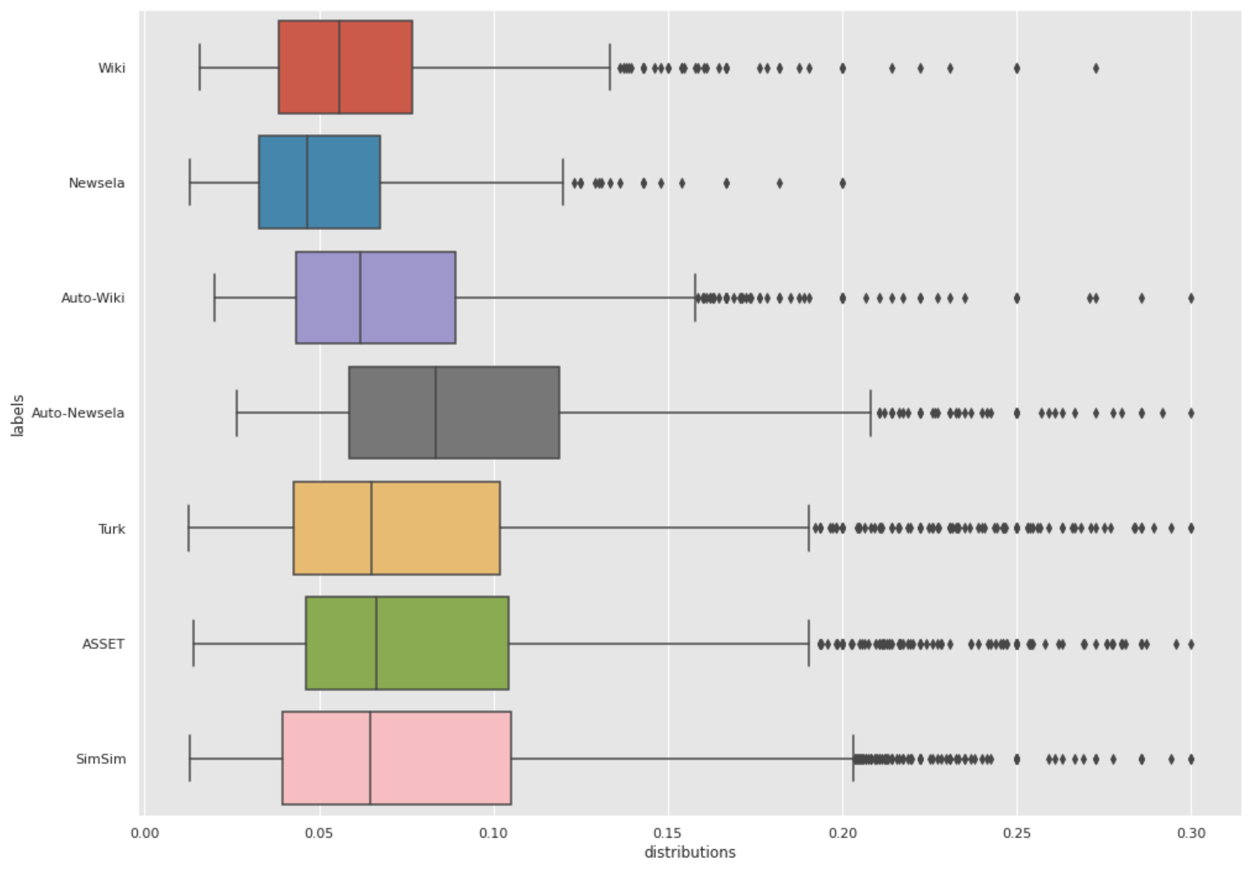


Figure 29: Substitution Analysis (BoxPlot)

4.3.2 Substitution Analysis

Figure 28 (Histogram) and 29 (BoxPlot) show the distribution of substitution rewriting operations for comparing SimSim to other training datasets. By simulating the substitution operation and adding PPDB transformation information into SimSim, I found SimSim achieves a similar distribution to that of Turk and ASSET. However, Newsela-Auto, which substitute more words, performs better in substitution operation.One of my future jobs is to treat Newsela-Auto as part of the seed sentence bank to further improve substitution operation.

4.3.3 Dropping Analysis

Figure 30 (Histogram) and 30 (BoxPlot) show the distribution of dropping rewriting operations for comparing SimSim to other training datasets. By simulating the dropping operation, I found SimSim achieves a most similar distribution to that of Turk and ASSET.

4.3.4 Splitting Analysis

Table 14 shows the ratio of samples contributing to splitting rewriting operation. SimSim contains 39.9 percent of sentence pairs contributing to splitting writing operations. I believe SimSim contains a sufficiently large number of samples to teach the model how to split.

4.3.5 Reordering Analysis

Figure 17 (Histogram) and 18 (BoxPlot) show reordering rewriting distribution for different corpora. Like simulating the substitution rewriting operation, after simulating the reordering rewriting operation, SimSim achieves a most similar distribution to that of Turk and ASSET.

4.3.6 Conclusion

As shown in the above analysis, SimSim demonstrates (1) a closer distribution on multiple dimensions to the human-annotated Turk and ASSET dataset than the others, suggesting

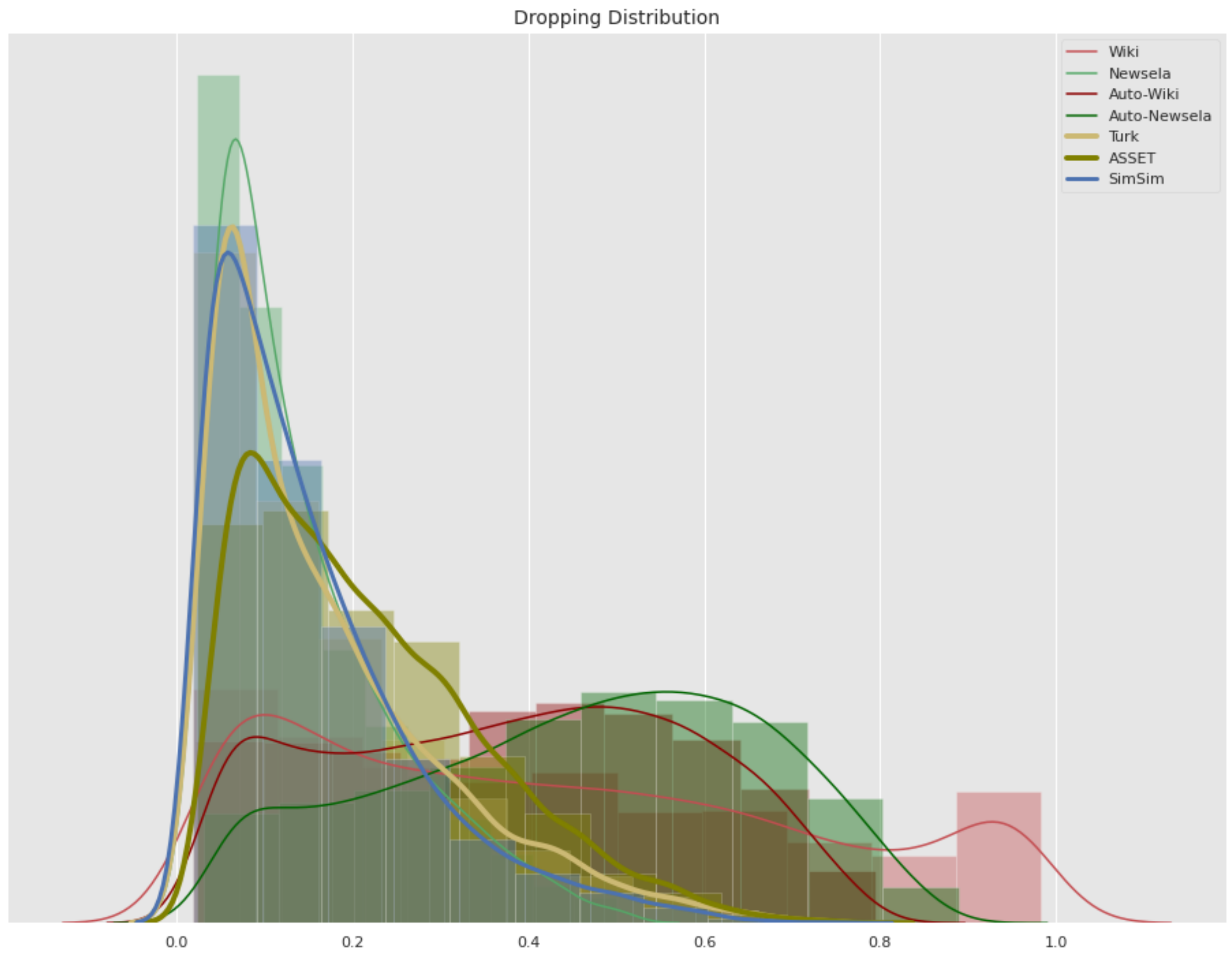


Figure 30: Dropping Analysis (Histogram)

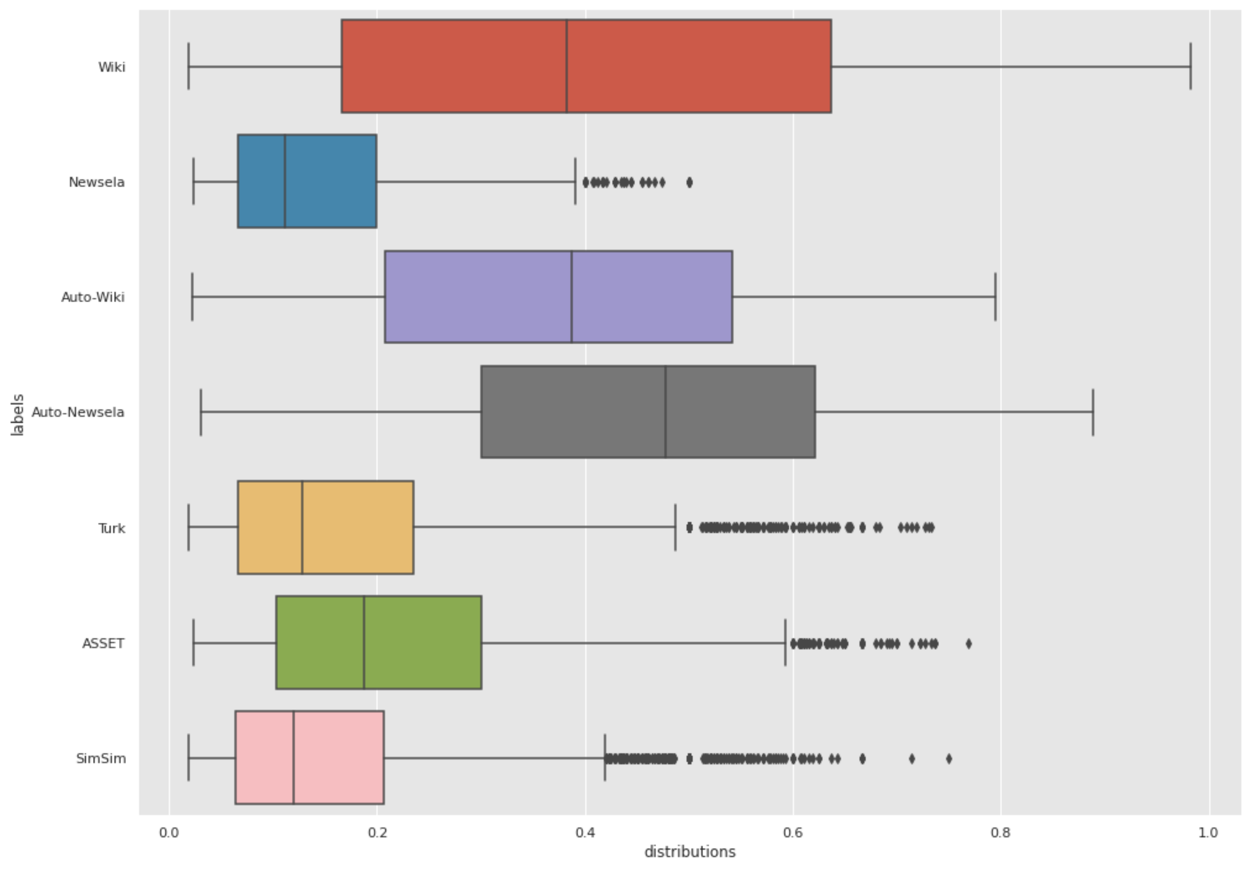


Figure 31: Dropping Analysis (BoxPlot)

Table 14: Splitting Analysis

Corpus	Split Ratio
Wikipedia	0.102
Newsela	0.002
Wiki-Auto	0.009
Newsela-Auto	0.016
Turk	0.044
ASSET	0.310
SimSim	0.399

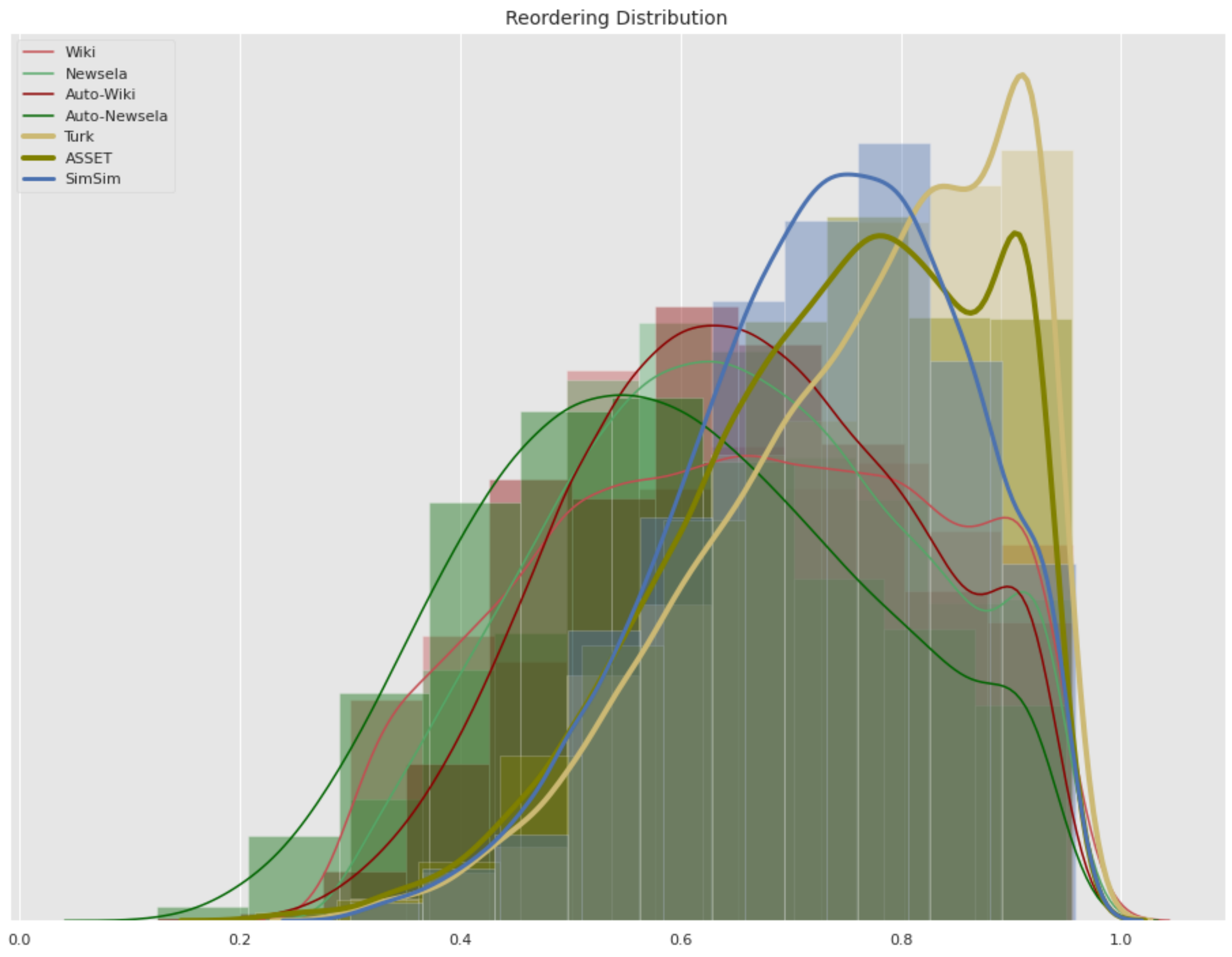


Figure 32: Reordering Analysis (Histogram)

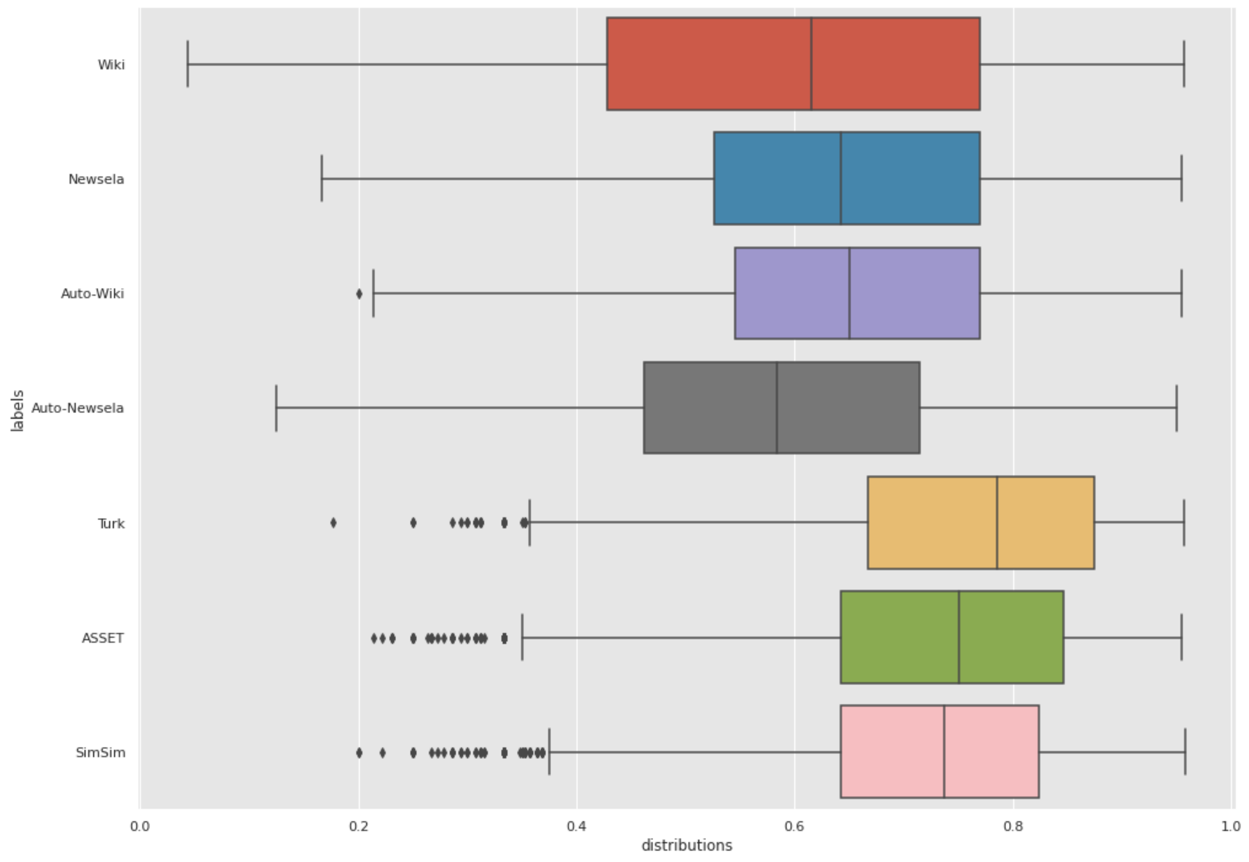


Figure 33: Reordering Analysis (BoxPlot)

that SimSim may serve as a better dataset for training automatic models; (2) a sufficiently large number of samples to teach the model how to perform certain operations, such as substitution and splitting.

These analyses simply compare the distribution of different simplification operation measurements. The model may behave differently. Therefore, in the following sections, I concentrate on model designs to further improve applying these simplification rewriting operations.

5.0 Model Design and Experiments for SimSim

This section focuses on designing a model and conducting experiments for SimSim.

5.1 Data Pipeline

5.1.1 Tokenization

After tokenizing SimSim, the dataset contained 885,263 unique words, with most lower frequency words being name entities. Therefore, following [98], I tagged and anonymized name entities using a special token with the format **NE@N**, where NE includes { PERSON, LOCATION, ORGANIZATION, NUMBER } and N indicates the N^{th} distinct NE entity type. As Table 15 shows, the sentence "It is situated at the coast of the Baltic sea , where it encloses the city of Stralsund" was changed to "It is located on the coast of LOCATION@1 and surrounds the city of LOCATION@2" by replacing the location name entities with anonymized name entity tags. I did not believe the pipeline would affect sentence simplification because I rarely needed to substitute name entities in complex sentences. Furthermore, I found that anonymizing name entities substantially simplified the model optimization because the model did not need to learn the semantic meaning of those name entities.

Table 15: Sample of Anonymized sentence

	Sentences
Original	It is situated at the coast of the Baltic sea , where it encloses the city of stralsund .
Anonymized Sentence	It is located on the coast of the LOCATION@1 and surrounds the city of LOCATION@2 .

After anonymizing the name entities, the dataset was reduced to 157,424 unique words, indicating a word count much smaller than the original. However, this vocabulary size remained too large for a deep learning model for several reasons:

1. Unrecognized Name Entities: I used Stanford CoreNLP Name Entities Recognizer tools [50] to tag and anonymize name entities. However, the recall performance of such tools is imperfect, especially given the fact that new name entities appear every day, precluding any guarantee that all name entities will be correctly tagged.
2. Misspelled Words: Misspelled words are infrequent. For example, the word "tranformed" does not exist in the English language; the correct spelling is "transformed," which requires inserting a single letter into the misspelled word. I expected the model to understand such words by checking for misspelled words automatically.
3. Composite Words: Words can comprise a combination of frequently used words. For example, "grapefruit-flavored" comprises three frequently used words: "grape," "fruit," and "flavored." I expected the model to be able to learn word compositions.
4. Other infrequent words: Some rarely used words, for example, "macropedia," are found infrequently but can be easily understood by checking elements such as the root and suffix; in the case of "macropedia," they are "macro" and "pedia."

Based on this analysis, I expected the tokenization to understand word composition. Following [51], I used subword tokenization [70] for this study's vocabulary. Based on the intuition that various word classes can be translated using units smaller than words—for instance, names, compounds, cognates, and loanwords—the subword tokenizer splits words into subwords, which are then defined by another vocabulary.

The subword vocabulary is learned through an unsupervised approach based on a large corpus, usually Wikipedia. When training the subword vocabulary, users provide an expected vocabulary size, and the byte pair encoding (BPE) [23] algorithm optimizes a subword vocabulary that tokenizes the provided corpus into the smallest number of subwords. That is, rather than explicitly checking English suffixes, prefixes, and roots, subword tokenization checks word units using a data-driven approach. This subword embedding in this work produced a total of 30,522 subwords.

Table 16: Samples of Subword Tokenization

Word	Tokenized Subwords
tranform	tran ##form
grapefruit-flavored	grape, ##fr, ##uit, -, flavor, ##ed
macropedia	macro, ##ped, ##ia
allentown	allen, ##town
alaaeldin	ala, ##ael, ##din
19801230	1980, ##12, ##30

Table 16 shows samples of subword tokenization. [1] Although the word "tranform" is misspelled, its subword units "tran" and "##form" enable understanding. Meanwhile, the composite word "grapefruit-flavored" and the infrequently used word "macropedia" can also be split into meaningful subword units. Additionally, subword tokenization is useful for unrecognized name entities. For example, location name entities, such as "allentown," and person name entities, "alaaeldin," can also be split into smaller subword units to enable understanding. Notably, even a number such as "19801230" can be segmented meaningfully to be recognized as a date.

As Table 17 shows, for each sentence, I used the subword tokenizer to tokenize sentences into sequences of subword units, with each word unit represented by its index in the subword vocabulary.

Tagging and anonymizing name entities in SimSim does not guarantee that name entities in a simplified sentence will exist in the complex sentence. Table 18 shows how name entities can differ between complex and simplified sentences as a result of incorrect GNMT translation. For example, GNMT confuses "MacFarlane" and "McFarlane," two spellings of a person's name entity. Due to the different spellings, the name entities recognizer tool treats them as two individual name entities. The mismatched name entities in the training

[1] The table only shows sampled subword tokenization. Some words in this table are replaced with certain name entity tags.

Table 17: Sample of tokenization pipeline

	Sentences
Original	It is situated at the coast of the Baltic sea , where it encloses the city ofstralsund .
Anonymized Sentence	It is situated at the coast of the LOCATION@1 , where it encloses the city LOCATION@2 .
Subword Tokenized Sentence	it is situated at the coast of the location@1 where it en ##cl ##oses the city location@2 .
Subword Tokenized Index	1068, 1062, 3406, 1071, 1055, 2080, 1056, 1055, 151, 69, 1132, 1068, 3428, 19512, 26513, 1055, 1162, 152, 71

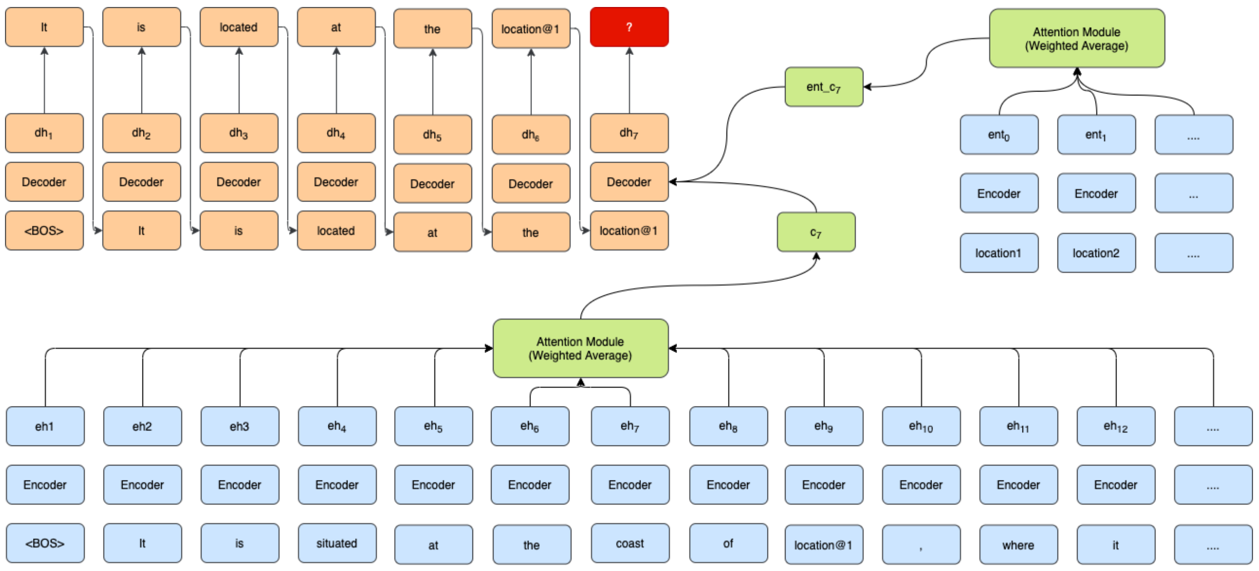


Figure 34: Overview of Conditional Language Model Guilded By Name Entities

data confuse the model, leading it to generate name entities that did not exist in the complex sentence.

Table 18: Sample of Anonymized Complex-Simplified Sentence Pairs

	Sentences
Complex Sentence	MacFarlane also has a character on Family Guy named after Hartman : named Dr. Elmer Hartman .
Anonymized Complex Sentence	person3 also has a character on family guy named after person1 : named dr. person2 .
Simplified Sentence	McFarlane 's Family Guy has a character by the name of Hartman : Dr. Elmer Hartman .
Anonymized Simplified Sentence	person0 's family guy has a character by the name of person1 : dr. person2 .
Mapping	McFarlane → PERSON0 Hartman → PERSON1 Elmer Hartman → PERSON2 MacFarlane → PERSON3

To avoid having the model generate nonexistent name entities, I extended the conditional language model introduced in Section 2.1.2 and proposed name entities to guide the conditional language model. As Figure 34 demonstrates, I input a list of name entities tags into the model to guide it to generate only name entities existing in the simplified sentence. The decoder observed the information from the complex sentences and information from the name entities tag list using the same attention mechanism. Accordingly, the model learned to copy the name entities tag in the name entities tag list into the simplified sentence. At the inference stage, I used the name entities list to replace name entities in the complex sentence; thus, the model would only generate name entities existing in the complex sentence and would not generate nonexistent name entities.

5.1.2 Integrating with PPDB

Although SimSim already simulated a substitution rewriting operation, the sentence simplification model itself inferred such substitutions from the dataset. This section introduces a deep learning model architecture based on Transformer, which, as discussed in Section 2.1.2, allows the simplification model to be explicitly aware of substitution.

First, I preprocessed PPDB, as described in Section 2.2.1.1. For each complex sentence in the corpus, I extracted candidate rules and ranked them according to their simplification scores. The following paragraphs discuss the integration of mapping rules into deep learning models.

The format of the substitution mapping rule only reveals which word or phrase is the simplified version of a given complex word or phrase, meaning that it cannot directly integrate into a differential deep learning model. Additionally, direct replacement is not an optimal solution because the application rule is context-dependent, prompting researchers to investigate other approaches to integrating PPDB into deep learning models.

My previous model [99] proposed using external memory to integrate PPDB mapping rules into deep learning models (Figure 35). Augmented memory is a collection of key-value pairs for each PPDB mapping rule, with the key denoting the context and the value denoting the output. In my implementation, augmented memory training is divided into two stages.

The first stage aims to update the augmented memory. Each time the model predicts a word recognized by PPDB (for example, if the model predicts the word "winner" and PPDB recognizes "winner" as a simplified form of "recipient"), it updates the key-value pair in the augmented memory. I use a hidden state for the second multi-head attention (first layer) as the key because hidden states encode information about both complex sentences and generate a simplified sentence. Then, I update the augmented memory using the final hidden state of the "Decoder" as the value.

The second stage aims to combine the augmented memory with the model prediction. After the augmented memory is updated, a "Combiner" merges the information from the "Decoder" with the augmented memory to make a word prediction. I use a feed-forward neural network to implement the "Combiner."

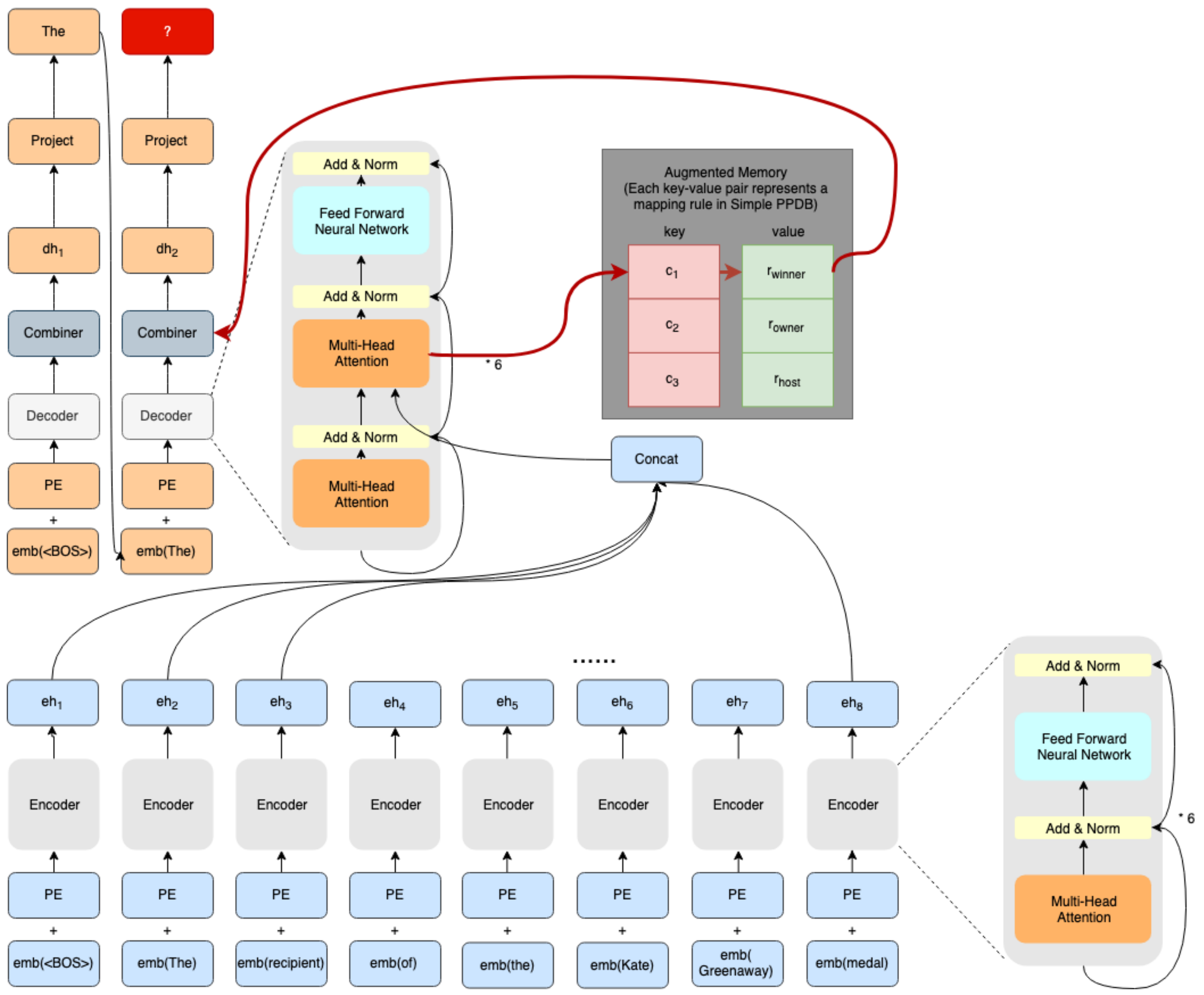


Figure 35: Overview of Augmented Memory Deep Learning Architecture in Zhao et al.[99]

However, there are two major drawbacks to this approach.

1. The augmented memory requires the context of each mapping rule to be encoded explicitly, therefore the memory can only store the mapping rules that occur in the training dataset.
2. Due to the format of key-value pairs in the augmented memory, both of which are single vectors, the memory can only effectively model the mapping rules with a single word in a simplified form. However, there are many mapping rules in Simple PPDB for which the simplified form is a phrase (i.e., it contains more than one word).

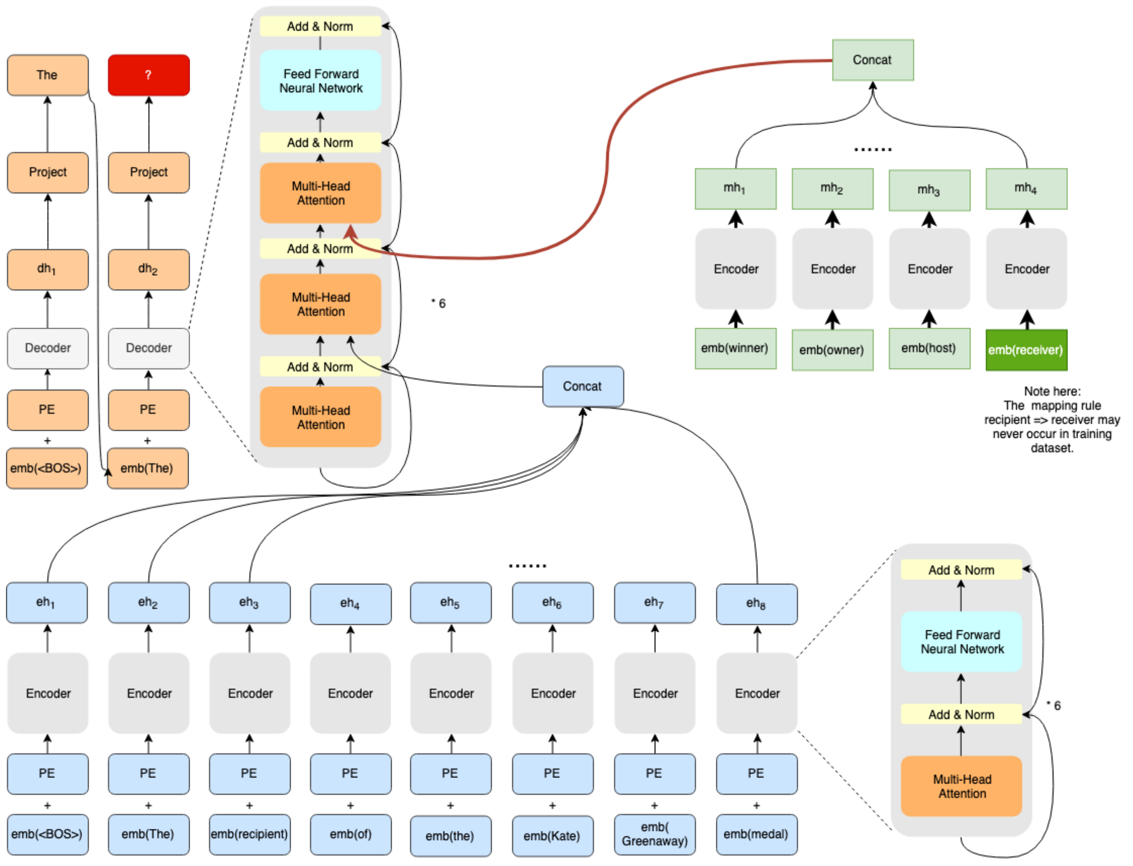


Figure 36: Overview of my proposed Augmented Memory Deep Learning Architecture

To resolve these drawbacks, I propose the new augmented memory deep learning architecture depicted in Figure 36. Rather than encoding the context explicitly, this approach

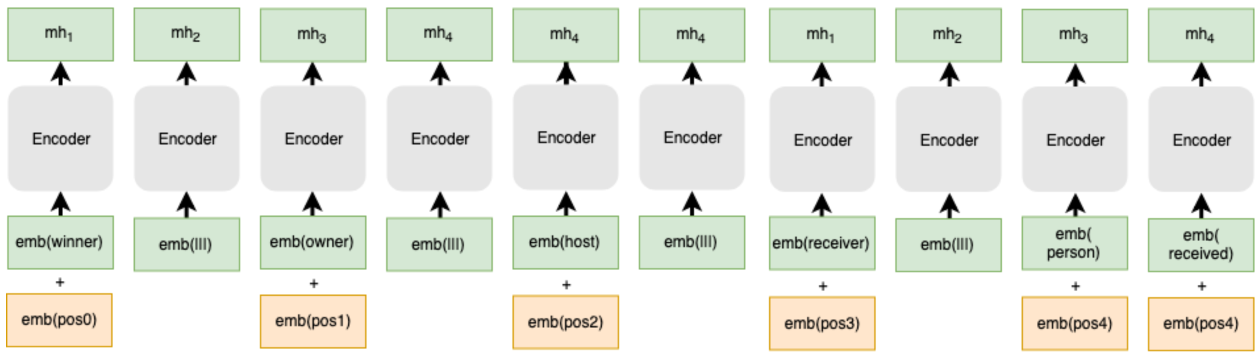


Figure 37: Overview of my proposed PPDB Encoder

uses another "Encoder" with the same structure as that for complex sentences; this encodes candidate words extracted from candidate mapping rules. As such, a simplified word that may not occur in the mapping rules of the training dataset can still be encoded. I insert another multi-head attention sub-layer in the "Decoder" to obtain information from the augmented memory.

Employing this architecture, I presume that the "Decoder" can coordinate information from three sources: (1) words in complex sentences, (2) words in mapping rules recognized by Simple PPDB, and (3) words that were previously generated in the simplified sentence. Finally, the "Decoder" predicts a word after coordinating information from these sources.

Another benefit of this architecture is that it enhances the flexibility of encoding words in the mapping rules. The details of the augmented memory are presented in Figure 37, demonstrating that separating words with different mapping rules requires learning the embedding of a separator |||. This allows a phrase to be encoded, with different phrases separated by |||. As the words are ranked according to their simplicity score, I also learned positional embedding for the nth position (n ranges 1–15). This enabled the model to pay more attention to higher-ranked words.

5.2 Evaluation

5.2.1 Comparison to Other Models

This section enumerates recently published baselines with high SARI.

- **PBMT-R** [**90**] Phrase-based machine translation system that re-ranks candidates, favoring dissimilar candidates based on their Levenshtein distance to the complex sentences.
- **SBMT-SARI** [**53**] Syntax-based machine translation model integrated with PPDB [59] and finetuned towards SARI.
- **DRESS-LS** [**98**] Encoder-Decoder model trained with reinforcement learning, rewarding higher SARI candidates.
- **DMASS+DCSS** [**99**] Encoder-Decoder model integrating with PPDB [59] by allocating augmented memory.
- **NTS+SARI** [**55**] Encoder-Decoder model revising the beam search to favor high SARI.
- **NSELSTM-S** [**82**] Encoder-Decoder model with memory-augmented Neural Semantic Encoder, finetuned towards SARI.
- **ACCESS** [**51**] Encoder-Decoder model focusing on controlling several attribute candidates for promoting high SARI. This model previously achieved the best score.
- **DMASS2** My best performed model, which is an Encoder-Decoder model integrating with PPDB by allocating augmented memory and training using the SimSim dataset as discussed in previous section.

Table 19 compares the performance of several recently published models.

Due to the fact that SARI is the most reliable metric for the sentence simplification task [94], I would like to focus on a more detailed discussion regarding SARI results. To further examine SARI, the impact of F1 scores for three operations used to calculate SARI scores is discussed in this section.

DMASS2 demonstrated state-of-the-art SARI performance, which benefited from correctly adding words through our created corpus `SimSim` emphasizing the simulation of substitution.

Table 19: Performance of different models on the Turk dataset.

Model	FKGL	Factors in FKGL		SARI	F1 for operations of SARI			Rule Utilization		
		WLen	SLen		Add	Delete	Keep	Prec	Recall	F1
NTS-SARI	9.05	1.35	22.38	37.51	3.36	38.46	70.71	15.58	8.40	10.14
PBMT-R	8.35	1.30	22.08	38.56	5.73	36.93	73.02	22.32	10.44	13.16
SBMT-SARI	7.49	1.18	23.50	39.96	5.97	41.43	72.51	27.46	19.01	20.09
DRESS-LS	6.92	1.35	16.76	37.27	2.82	42.21	66.78	9.54	4.76	5.89
NSELSTM-S	-	-	-	36.88	-	-	-	-	-	-
EditNTS	7.30	1.35	18.94	38.22	3.36	39.15	72.13	10.60	4.78	6.18
DMASS+DCSS	8.04	1.29	21.64	40.45	5.72	42.23	73.41	32.93	19.88	22.41
ACCESS	6.50	1.26	18.58	41.87	7.28	46.07	72.58	28.06	14.38	17.13
DMASS2	8.41	1.30	22.22	43.07	12.43	44.75	72.02	74.63	58.32	62.03

ACCESS represents a previous state-of-the-art model in terms of SARI score, which uses a controlling approach to bias model prediction to better fit human needs. Its developers introduced several metrics to reduce the complexity of words and sentences, forcing the model to generate sentences with less complex words and sentences. The goal of this approach appears similar to the simulating simplification operation, with this simulation taking place at the model level, which is more effective. However, this method does not integrate external knowledge, resulting in lower SARI. The next chapter introduces our control approach.

SBMT-SARI achieves a high F1 score for adding words and integrates external knowledge bases, performing well in terms of correctly adding new words but performing poorly at choosing to delete or keep words. By analyzing its predictions, SBMT-SARI acts aggressively to substitute as many words as possible, which can produce inaccurate simplifications.

PBMT-R behaves similarly to SBMT-SARI, acting aggressively to substitute as many words as possible by re-ranking candidates and preferring dissimilar model predictions based on Levenshtein distance to complex sentences. Such preferences lead to selecting sentences with more substitutions.

DMASS+DCSS uses a PPDB integration method similar to DMASS2; however, as analyzed in the previous chapter, the Wikilarge training dataset performs simplification operations poorly; this leads the DMASS+DCSS to perform not as well as DMASS2.

Rule utilization computes the precision, recall, and F1 for certain PPDB rules applied during model prediction, including whether they are applied. Both DMASS2 and the DMASS+DCSS model perform well in terms of rule utilization due to PPDB integration. As previously discussed, SBMT-SARI acts aggressively to substitute as many words as possible in the sentence. Such aggressive behavior promotes relatively high recall performance but diminishes precision and F1 performance.

FKGL measures the simplicity of a sentence by its sentence length (WLen) and average word length (WLen). However, because this measurement does not consider ground truth simplification references, a high FKGL may be counteracted by a loss of information and readability. Therefore, FKGL and two of its factors—WLen and SLen—are treated as supplementary metrics.

ACCESS achieves the best FKGL performance because its controlling approach forces the model to generate short sentences. For ACCESS, sentence length is a factor controlling model prediction. At the inference stage, it controls various factors to generate short sentences. Notably, DRESS-LS also performs well in terms of FKGL, with its policy gradient used to reword shorter sentences, thus producing lower FKGL scores.

Using an integrated external knowledge base, both SBMT-SARI and DMASS2 can generate shorter words to simplify sentences. However, DMASS2 performs worse in terms of FKGL. This is because SimSim is constructed by back-translation, which is not effective at generating short sentences. I attempted to resolve this by simulating a dropping operation, however, improvements were limited, as discussed in the next sections.

5.2.2 Ablation Study

This section enumerates models trained on our dataset SimSim in different stages. For ablation study, I compared the performance of these models.

- **Stage 1** As discussed in Section 4.1, the initial stage involved the dataset being constructed purely by the back-translation approach.

Table 20: Ablation Study of Encoder-Decoder model on the Turk dataset.

Model	FKGL	Factors in FKGL		SARI	F1 for operations of SARI			Rule Utilization		
		WLen	SLen		Add	Delete	Keep	Prec	Recall	F1
EncDec-Stage 1	8.85	1.34	22.12	36.33	4.53	32.79	71.66	0.68	0.68	0.68
EncDec-Stage 2	8.83	1.32	22.60	40.15	7.52	38.64	74.32	20.28	10.39	12.46
EncDec-Stage 3	8.31	1.30	22.03	41.07	8.33	41.97	72.89	27.85	16.62	19.07
DMASS2-Stage 4	8.41	1.30	22.22	43.07	12.43	44.75	72.02	74.63	58.32	62.03

- **Stage 2** As discussed in Section 4.1, the second stage involved the dataset being constructed using the back-translation approach, enabling the translation of complex sentences into 103 simplified sentences, before using GPT-2 to select the most fluent sentence.
- **Stage 3** As discussed in Section 4.2, the third stage involved the dataset being further improved by simulating four simplification operations.
- **Stage 4** As discussed in Section 5.1.2, the fourth stage involved explicit integration with PPDB by adding external memory.

Figure 20 and 21 show the model's performance using the SimSim dataset at Stage 1, Stage 2, Stage 3 and Stage 4. SARI and rule utilization, except for FKGL, improved gradu-

Table 21: Ablation Study of Encoder-Decoder model on the ASSET dataset.

Model	FKGL	Factors in FKGL		SARI	F1 for operations of SARI			Rule Utilization		
		WLen	SLen		Add	Delete	Keep	Prec	Recall	F1
EncDec-Stage 1	8.85	1.34	22.12	49.34	17.10	66.48	64.44	15.11	7.58	8.97
EncDec-Stage 2	8.83	1.32	22.60	52.20	19.34	69.89	67.38	20..56	11.15	13.70
EncDec-Stage 3	8.31	1.30	22.03	52.37	19.18	71.01	66.92	29.67	18.21	20.84
DMASS2-Stage 4	8.41	1.30	22.22	52.47	12.43	72.35	65.56	44.32	34.00	35.65

ally from Stage 1 to Stage 4. Improvement by further simulations of simplifications was limited. Investigation revealed that, although I had simulated operations into the dataset which contributed further to the simplification operations, the deep learning model would average the contributions of the entire training dataset, weakening contributions to simplification operations. For example, although simulating substitution and reordering operations enabled simplifying word usage and sentence structure for some sentences, a considerable number of sentences were missing this simulation because of the limitations of back-translation and GPT2. While simulating a dropping operation randomly removes prepositional phrases, it does not guarantee that all prepositional phrases in the training dataset will be removed. Nonetheless, this remains superior to human-written sentences because humans will not simplify each part of a sentence.

My finding of the model trained by a dataset purely constructed by back-translation is similar to related works. Sentence pairs provided by back-translation are not necessarily complex-simplified sentence pairs, which leads to poor performance in sentence simplification evaluation.

The improvement of the models in Stage 4 compared to Stage 3 was significant. For Stage 4, I revised the model to explicitly check the candidate words contributing to substitution operations, significantly improving the substitution operation. As shown in Table 20, the improvement is reflected by scores for both SARI and rule utilization. However, the FKGL performance did not improve because substitutions do not shorten sentences.

5.2.3 Comparison to Other Corpus

Table 22 compares baseline model performances using recently published training datasets. Wiki-Auto was constructed by aligning sentences in the Complex and Simple Wikipedia using a neural CRF model featuring a BERT as a base model. Different traditional alignment approaches using lexical-level features, such as the Jaccard coefficient or the cosine distance of TF-IDF vectors, enabled this approach to capture semantic similarity and leverage similarly ordered content. Consequently, Wiki-Auto features much more cleanly aligned complex-simplified sentence pairs.

Table 22: Performance of different Corpora.

Model	FKGL	Factors in FKGL		SARI	F1 for operations of SARI			Rule Utilization		
		WLen	SLen		Add	Delete	Keep	Prec	Recall	F1
Wiki-Auto	7.67	1.35	19.0	39.64	5.18	41.61	72.13	1.01	1.35	1.13
SimSim	8.31	1.30	22.03	41.07	8.33	41.97	72.89	1.58	2.03	1.69

However, given the limitations of Complex and Simple Wikipedia, the simplification is not sufficient: Wiki-Auto's contributions to simplification operations remain limited, leading to significantly worse SARI and rule utilization performance than when using SimSim. However, Wiki-Auto performed better for FKGL because sentences in Wiki-Auto are generally short. As discussed, due to SimSim back-translation construction, complex and simplified sentences are always similar in length.

6.0 Tunable Sentence Simplification Models

In the last chapter, I proved that the model DMASS2 trained by SimSim achieves the best SARI score. The improvement is reflected mostly in the adding (substituting) of simplified words. Although I introduced a dropping simulation operation into the SimSim construction, the model simply averages all simplification operations that still perform poorly in generating a short sentence.

On the other hand, depending on the setting, users may prefer various forms of simplified sentences related to different simplification operations. Therefore, a deep-learning model architecture that allows the insertion of style-related parameters for different styles of sentence simplification is critical. For example, excessive focus on the dropping operation may lose the essential meaning of the original complex sentence, which is harmful to meaning preservation; however, some users prefer such a style of sentence simplification. On the contrary, a slight dropping operation and additional splitting do not harm meaning preservation, and they generate another option for sentence simplification.

Therefore, in this chapter, I construct a tunable sentence-simplification model by allowing for the insertion of style-related parameters. This model allows users to provide style-related features, and it can generate certain sentence styles based on user input. The style-related features can be connected to the four simplification rewriting operations.

6.1 Tunable Sentence Simplification

To enable tunable sentence generation in a general NLP (natural language processing) domain, two approaches can be applied: policy gradient [63] and prefix constraint [75].

The policy gradient method uses a reinforcement learning approach and treats user-provided, style-related features as the rewards. After training, the policy gradient prefers to generate a sentence that favors user-preferred styles.

Meanwhile, the prefix constraint adds a restriction (i.e., a constraint) to the model. The constraint is a user-provided, style-related feature, and the model learns the dependency between the ground truth sentence and the constraints. In the inference stage, users can tune the constraints to generate a sentence that favors user-preferred styles.

6.1.1 Policy Gradient

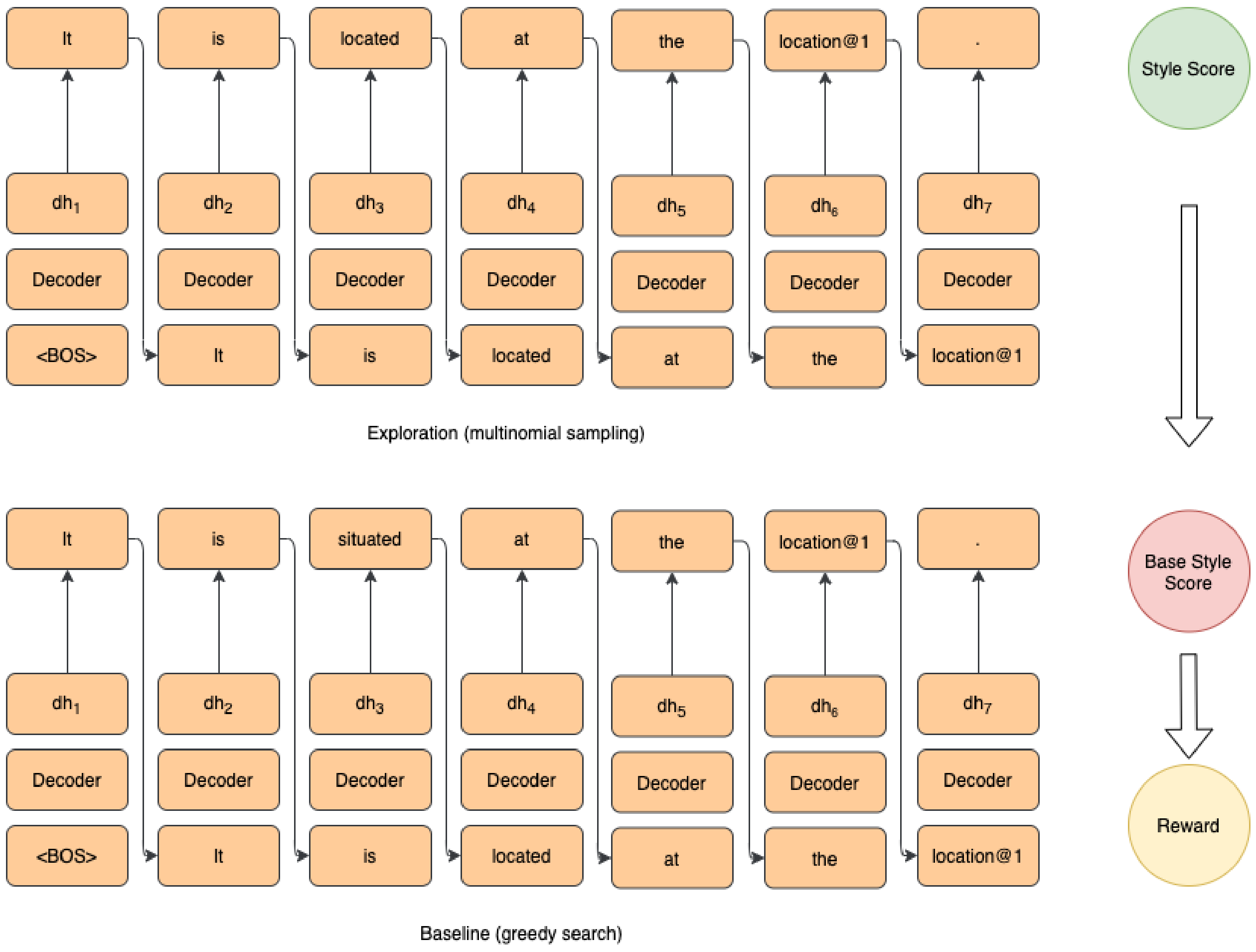


Figure 38: Overview of Architecture of Policy Gradient

Figure 38 shows how the reward is computed using the policy gradient method. In this method, the model tries to compare two sentences—the sample sentence and the baseline sentence—and compute the reward.

The baseline sentence indicates the standard performance of the text-simplification system. The base style score is always computed using a generated sentence from greedy search [66] or by averaging existing scores from a collection of generated sentences [63].

The sample sentence refers to the sentence explored by the model through multinomial sampling. When generating each word, instead of selecting the word with the highest probability, the sampling algorithm explores an alternative sentence. The sample style score is computed based on this sentence.

The policy gradient algorithm compares two sentences and computes the reward by subtracting two scores.

If the sample sentence is a successful exploration that generates a higher style score, the reward is positive. If the exploration is not successful, the reward is negative. The reward is then applied to determine weights for the objective function, and the optimization favors generation with a higher style score.

However, this approach has several weaknesses:

1. The sentence exploration is not stable because of a large vocab size. In my experiment, most of the sample sentences are not readable.
2. The reward favors certain simplification rewriting operations without considering the sentence readabilities. Due to the limitations of sentence exploration, a sample sentence contributes to certain simplification rewriting operations, but unreadable sentences are still rewarded during the training.
3. Because the style score is treated as a reward, and because the policy gradient method uses rewards to reassign weights for the objective function, such tuning occurs during the training stage. Each time the user needs to change the style, it is necessary to re-train the model.

Due to these limitations of using policy gradient methods to tune to sentence styles, I will also try to use prefix constraint methods.

6.1.2 Prefix Constraint

As shown in Figure 39, the prefix constraint method places an additional token, "style score," in front of the sentence. Because of the association between the style score and

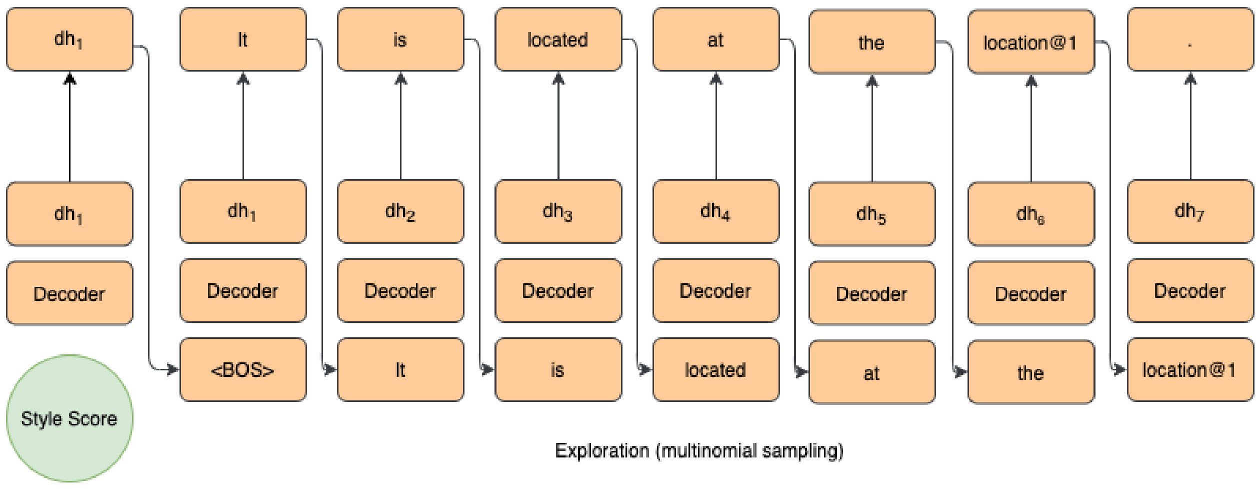


Figure 39: Overview of Architecture of Prefix Constraint

the ground truth sentence, the model becomes aware of the dependency between these two sentences via the self-attention mechanism.

Compared to the policy gradient model, there are several benefits to the prefix constraint method:

1. Sentence generation is more stable because there is no need to sample words. Generation in prefix constraint needs to consider both the language model and association with the style score.
2. Because the syntax score is used in the generation stage, it is not necessary to re-train the model if the user needs to change the style.

6.2 Tunable Sentence Simplification Models

This approach is also helpful for constructing a truly useful sentence simplification system that can perform all suitable operations. Because either generated data or existing linguistic and simplification resources may only apply to certain operations, and different resources may conflict with each other, a style featuring maximized performance for each operation is ideal for reducing sentence complexity.

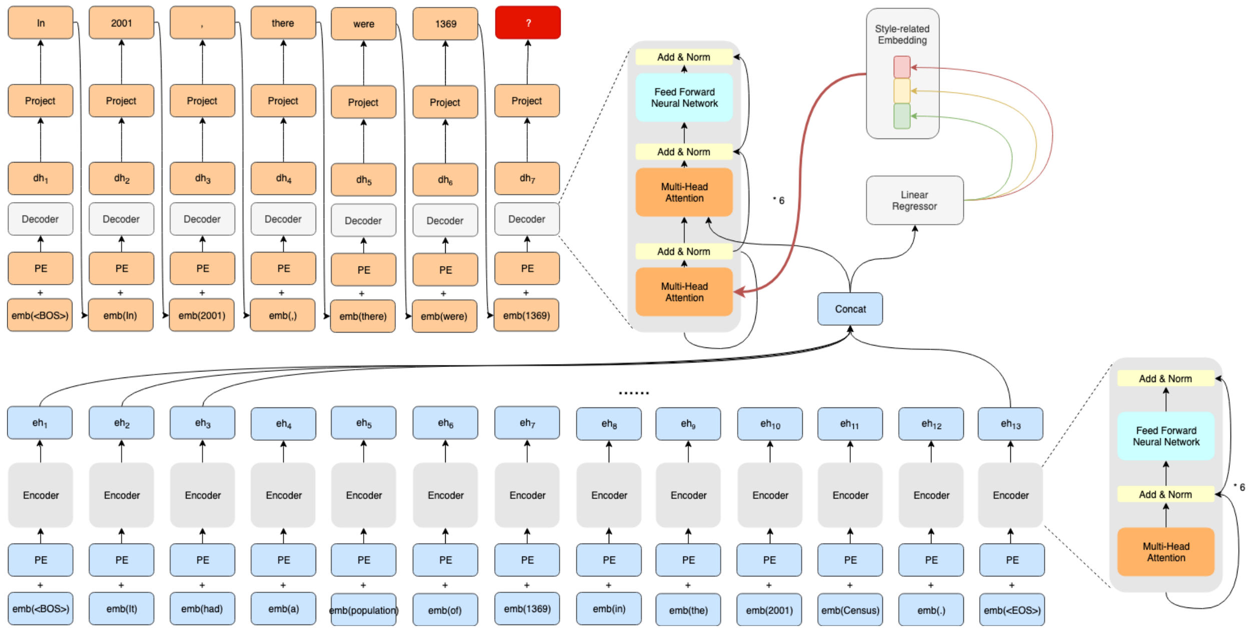


Figure 40: Overview of Architecture to Enable Styles of SS In Training

Inspired by prefix constraint [71] methods, I use style-related embedding to instruct the model as to which style it needs to generate (upper right corner of Figure 40).

Style-related embedding is a vector that encodes the style of a simplified sentence. Each style is encoded into a scalar value, as discussed in the previous section. To make the style-related embedding compatible with the deep-learning architecture, I tiled it to be the same size as the hidden state. For example, style-related values of {0.1, 0.5, 0.3, 0.9} would be tiled as {0.1, 0.1, 0.5, 0.5, 0.3, 0.3, 0.9, 0.9} if the size of the hidden state were 8. The style-related embedding is fed into multi-head attention.

In the training stage, when the ground truth simplified sentence is visible, I can calculate the style-related embedding and feed it to the model. The model will learn the dependency between the values in style-related embedding and the behavior of generation. In the testing/inference stage, the user can provide style-related values and put them into the style-related embedding; the model will then generate a sentence according to the user input.

For some style-related values, such as sentence length, it is hard for users to provide a global requirement for all simplified sentences. The ideal length of a simplified sentence may depend on the particular complex sentence. Therefore, to improve user convenience, I instruct a linear regressor to predict the ideal length of a simplified sentence. Thus, in training time, the linear regressor learns to predict the length of the simplified sentence. In testing/inference time, users are free to scale the predicted length (e.g., by a multiple of 0.8).

I followed the teacher-forcing approach to optimize the model. For each word in the sentence, I used the ground truth sentence from a prior time step as input and the current word as output for prediction. When predicting each word, the model will consider the following three pieces of information:

1. The complex sentence
2. The simplified sentence in the prior time step
3. The style-related embedding

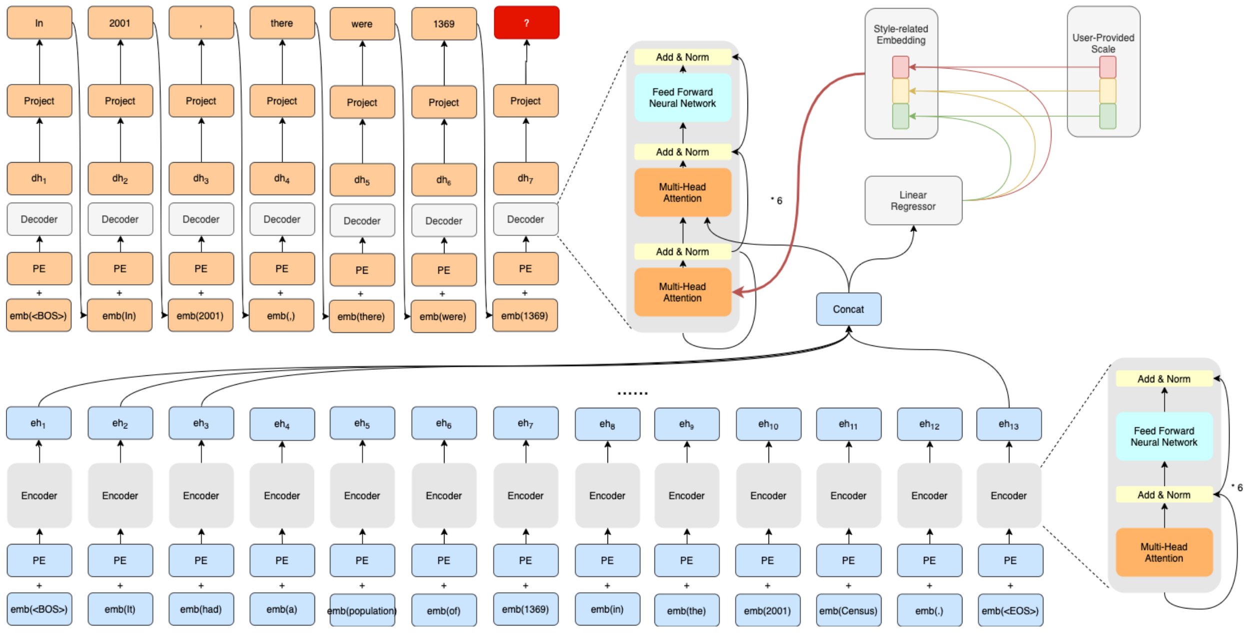


Figure 41: Overview of Architecture to Enable Styles of SS In Inference

In the inference stage, users are allowed to change the style-related embedding. As shown in Figure 41, one option is to provide a scale vector indicating a multiplier applied to style-related embedding. For example, a scale greater than 1.0 always promotes the current style, and a scale less than 1.0 reduces the current style. In the real use case, we can tune these style scores by referencing certain development datasets to obtain appropriate sentence simplification styles.

In the next sections, I will introduce the details of tuning styles that contribute to each simplification rewriting operation.

6.2.1 Tuning the Substitution Rewriting Operation

To promote the substitution rewriting operation, I use the substitution measurement mentioned in Section 3.1.2. Thus, I compute the ratio of words in the complex sentences that have been replaced by PPDB and treat the ratio as the style-related parameters to the substitution-rewriting operation.

Table 23 and 24 show typical examples of the different sentences generated by tuning the substitution rewriting operations. Inhibiting substitution always reduces the substitution of words in the simplified sentence and generates sentences similar to the complex sentence.

Promoting substitution improves the substitution of words in the simplified sentence. Because my model is integrated with PPDB, generation with no tuning already works well. Further tuning to favor the substitution rewriting operation will be prone to error. In the first example, the model substituted the word “able-bodied” to “canbodied” because it noticed the word “can” could be substituted for “able.” In the second example, the model substituted “award” for “medal.”

It is not difficult to tune word usage in sentence generation. In fact, there are studies to tune word usage in other NLP domains [100]. Therefore, my confidence in tuning the substitution of the sentence simplification system is high.

Table 23: Sample of Influence of Tuning Substitution Rewriting Operation

Style	Sentences
Complex Sentence	jeddah is the principal gateway to mecca , islam 's holiest city , which able-bodied muslims are required to visit at least once in their lifetime .
No Tuning	jeddah is the main gateway to islam 's holiest city , which allows able-bodied muslims to visit at least once in their lifetime .
Promoting Substitution	jeddah is the main gateway to islam 's holiest city that canbodied muslims must visit at least once during their lifetime .
Inhibiting Subsitution	jeddah is the principal gateway to mecca , islam 's holiest city , which able-bodied muslims are required to visit at least once in their lifetime .

Table 24: Sample of Influence of Tuning Substitution Rewriting Operation

Style	Sentences
Complex Sentence	since 2000 , the recipient of the kate greenaway medal has also been presented with the colin mears award to the value of $ 5000 .
No Tuning	since 2000 , the winner of the kate greenaway medal has also been given with the colin mears award to the value of $ 5000 .
Promoting Substitution	since 2000 , the winner of the kate greenaway medal has also been given with the colin mears medal of $ 5000 .
Inhibiting Substitution	since 2000 , the recipient of the kate greenaway medal has also been presented with the colin mears award to the value of the number of 5000 .

6.2.2 Tuning the Dropping Rewriting Operation

To promote the dropping rewriting operation, I used the measurement mentioned in Section 3.1.3.

Table 27 and 28 show typical examples of the different sentences generated by tuning the dropping rewriting operation. Different levels of promoting this operation drop different numbers of words. Because my prefix constraint is soft, the generated sentence is always readable. Notably, most of the dropping operation removes prepositional, adjectival, and adverbial phrases. These are learned from SimSim in simulated dropping, which is discussed in Section 4.2.2. Because of this, compared with earlier works that generate short sentences, my tuning of the dropping rewriting operation has a smaller chance of deleting important information from the sentence.

It is also not difficult to tune sentence length in sentence generation. There are studies to tune sentence length in other NLP domains [46]. Therefore, my confidence in tuning the

Table 25: Sample of Influence of Tuning Dropping Rewriting Operation

Style	Sentences
Complex Sentence	jeddah is the principal gateway to mecca , islam 's holiest city , which able-bodied muslims are required to visit at least once in their lifetime .
No Tuning	jeddah is the main gateway to islam 's holiest city , which allows able-bodied muslims to visit at least once in their lifetime .
Promoting Dropping	jeddah is the main gateway to mecca , the most holy city in islam ,able-bodied muslims are required to visit at least once .
Further Promoting Dropping	jeddah is the main gateway to mecca , the most holy city in islam .

Table 26: Sample of Influence of Tuning Dropping Rewriting Operation

Style	Sentences
Complex Sentence	since 2000 , the recipient of the kate greenaway medal has also been presented with the colin mears award to the value of $ 5000 .
No Tuning	since 2000 , the winner of the kate greenaway medal has also been given with the colin mears award to the value of $ 5000 .
Promoting Dropping	since 2000 , the winner of the kate greenaway medal has also been given the colin mears award to $ 5000 .
Further Promoting Dropping	since 2000 , the winner has also been given the colin mears award .

dropping of the sentence simplification system is high.

6.2.3 Tuning the Splitting Rewriting Operation

To promote the splitting rewriting operation, I used the measurement mentioned in Section 3.1.5. The measurement of the splitting rewriting operation is a Boolean value that indicates whether the complex sentence is split.

Table 27: Sample of Influence of Tuning Splitting Rewriting Operation

Style	Sentences
Complex Sentence	jeddah is the principal gateway to mecca , islam 's holiest city , which able-bodied muslims are required to visit at least once in their lifetime .
No Tuning	jeddah is the main gateway to islam 's holiest city , which allows able-bodied muslims to visit at least once in their lifetime .
Promoting Splitting	jeddah is the main gateway to mecca , the holiest city in islam . able-bodied muslims are required to visit at least once in their lifetime .

Table 27 and 28 show typical examples of the varied sentences generated by tuning the splitting rewriting operation. After promoting this operation, the model split the sentence into two or more short sentences. Similar to before, the styles of sentence splitting are learned from the Wikisplit data set, which, in turn, are learned from SimSim in simulating splitting, as discussed in Section 4.2.3.

6.2.4 Tuning the Reordering Rewriting Operation

As in reordering simplification rewriting operations, I used the measurement mentioned in Section 3.1.4 to promote my reordering rewriting operation. However, I found the model generated similar sentences when promoting the reordering rewriting operations.

Table 28: Sample of Influence of Tuning Splitting Rewriting Operation

Style	Sentences
Complex Sentence	since 2000 , the recipient of the kate greenaway medal has also been presented with the colin mears award to the value of \$ 5000 .
No Tuning	since 2000 , the winner of the kate greenaway medal has also been given with the colin mears award to the value of \$ 5000 .
Promoting Splitting	since 2000 , the winner of the kate greenaway medal has also been given the colin mears award . the kate greenaway medal was given to the value of \$ 5000 .

After analyzing the reason for this, I found that the model was unaware of the syntax of the generated sentence. The sentence simplification model works via a statistical approach in which the model predicts the probabilities:

$$\begin{aligned}\mathcal{P}(w_1, w_2, ...w_n|w_{complex}, style) =& \mathcal{P}(w_n|w_1, w_2, ...w_{n-1}, w_{complex}, style)\\ &\mathcal{P}(w_{n-1}|w_1, w_2, ...w_{n-2}, w_{complex}, style)....\\ &\mathcal{P}(w_2|w_1, w_{complex}, style)\mathcal{P}(w_1|w_{complex}, style)\end{aligned} \tag{14}$$

As shown in the above formula, $w_{complex}$ indicates the words in the complex sentence and *style* indicates the style-related parameters. The model predicts next-word probabilities based on the complex sentence and style-related parameters and selects the word with the highest probability during generation.

This setting applies to shallow styles, such as word usage and sentence length, because it is easier to model $\mathcal{P}(w_1, w_2, ...w_n|w_{complex}, style)$, but the model is not aware of the syntactical functionality of each word during generation. Therefore, it is necessary to add syntax information to the probabilities by modeling $\mathcal{P}(w_1, w_2, ...w_n|w_{complex}, style, syntax)$.

6.3 Syntax-Aware Tunable Sentence Simplification

In this section, I introduce my syntax-aware tunable sentence simplification model. As discussed in the previous section, the model needs to be aware of the syntactical information to tune the reordering simplification rewriting operation.

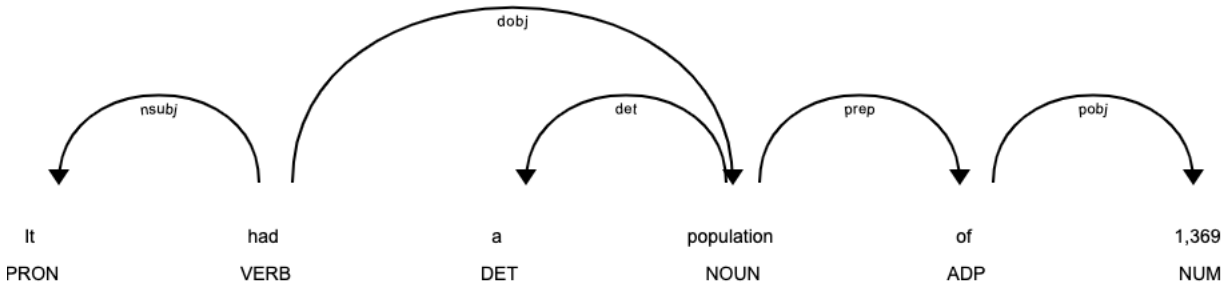


Figure 42: Dependency Parser Output

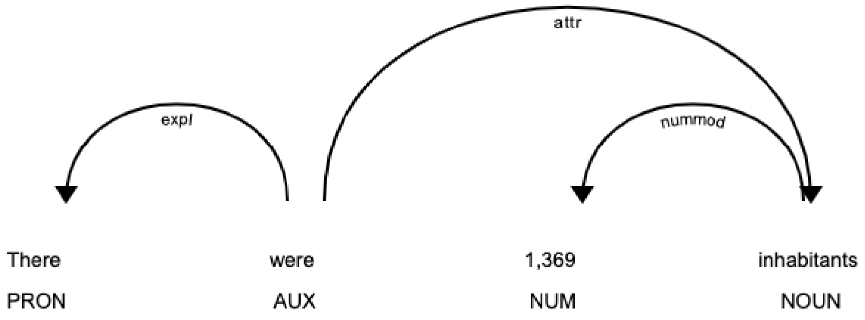


Figure 43: Dependency Parser Output

As mentioned in Section 3.1.4, I use a dependency parser [57] and treat the results as the representation of syntax. Figure 42 and 43 show the dependency parser results for a complex-simplified sentence pair. For both of them, the syntax representations follow a hierarchical structure starting with a "ROOT" (the words "had" in Figure 42 and "were" in Figure 43) and follow different paths to different words with certain syntactical functions.

To make such hierarchical structures compatible with deep-learning models, I stack the syntactical roles of each word for the complex sentence, as shown in Figure 44. The syntax representation of the complex sentence is a stack of syntactical roles on different levels. For example, the syntactical role for word "1,369" on the first level is dobj because "1,369" is the direct object of "had." Then, the syntactical role for "1,369" on the second level is prep because "1,369" has a prepositional relationship with the word "population." In this work, I only encode syntactical roles from the first two levels into the model.

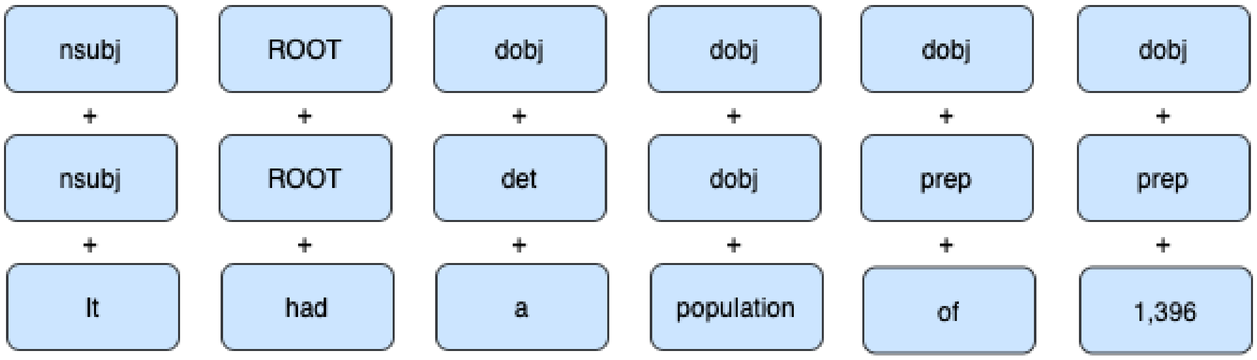


Figure 44: The Syntax Representation for complex sentence

The model learns a syntax token embedding and feeds the sum of token embedding and syntax token embedding into the Transformer Encoder, as shown in Figure 45. In this way, the model is aware of the syntax information of the complex sentence.

To produce syntax-aware sentence generation, I revised the Transformer Decoder to interleave generated syntax role token and word token. In Figure 45, the model first generates the syntax role tokens and then generates word tokens based on those syntax role tokens.

In this way, the current model predicts the following probabilities:

$$\begin{aligned}\mathcal{P}(s_1, w_1, s_2, w_2, ...s_n, w_n|w_{complex}, style) =& \mathcal{P}(w_n|w_1, w_2, ...w_{n-1}, s1, s2, ...s_{n-1}, w_{complex}, style)....\\ & \mathcal{P}(w_{n-1}|w_1, w_2, ...w_{n-2}, s1, s2, ...s_{n-2}, w_{complex}, style)....\\ & \mathcal{P}(w_2|w_1, s_1, w_{complex}, style)\mathcal{P}(w_1|w_{complex}, style)\end{aligned} \tag{15}$$

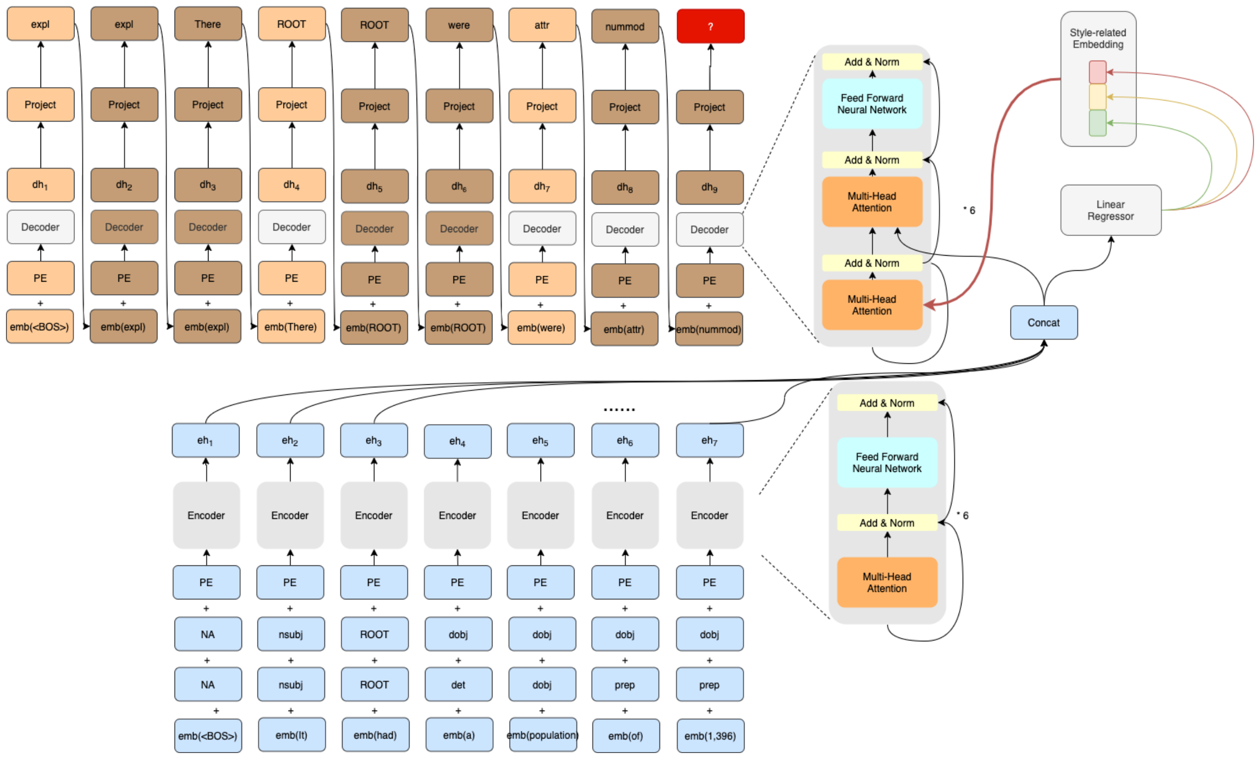


Figure 45: The Syntax-Aware Tunable Sentence Simplification

Table 29: Sample of Influence of Tuning Reordering Rewriting Operation

Style	Sentences
Complex Sentence	there he died six weeks later , on january january 888 .
Promoting Reordering	he died six weeks later , on january 13 , 888 .
Further Promoting Reordering	he died on january 13 , 888 , six weeks after his death .

Table 30: Sample of Influence of Tuning Reordering Rewriting Operation

Style	Sentences
Complex Sentence	each version of the license is given a distinguishing version number .
Promoting Reordering	each version of the license is given a distinct number .
Further Promoting Reordering	there is a distinct version number for each version of the license .

Table 31: Sample of Influence of Tuning Reordering Rewriting Operation

Style	Sentences
Complex Sentence	new south wales 's largest city and capital is sydney .
Promoting Reordering	the biggest city and capital of the new south wales region is sydney .
Further Promoting Reordering	sydney is the biggest city and capital of the new south wales .

Tables 29, 30, and 31 show typical examples of the different sentences generated by tuning the reordering rewriting operation. The different levels that facilitate this operation update the sentence structure to different levels. Similar to before, a slight promotion of reordering rewriting operations can improve the readability of complex sentences. However, excessive promotion of the rewrite operation will bring greater risks.

Table 32: Sample of Influence of Tuning Reordering Rewriting Operation

Style	Sentences
Complex Sentence	small value inductors can also be built on integrated circuits using the same processes that are used to make transistors .
Promoting Reordering	it is also possible to build small-value inductors on integrated circuits using the same processes used to make transistors .

Table 33: Sample of Influence of Tuning Reordering Rewriting Operation

Style	Sentences
Complex Sentence	therefore , these pdfs can not be distributed without further manipulation if they contain images .
Promoting Reordering	therefore , it is not possible to distribute these pdfs without further manipulation if they contain images .

Unlike tuning the substitution, dropping, and splitting rewriting operations, tuning the reordering rewriting operation can update the complex sentence in different ways, some of which are overlapped with other rewriting operations. Here, I enumerate different types of reordered sentences:

1. **Adding a Clause**: As shown in Tables 32 and 33, the reordered sentence adds a clause by moving the keyword “possible” to the front to make it clearer.

Table 34: Sample of Influence of Tuning Reordering Rewriting Operation

Style	Sentences
Complex Sentence	new south wales 's largest city and capital is sydney .
Promoting Reordering	sydney is the biggest city and capital of the new south wales .

Table 35: Sample of Influence of Tuning Reordering Rewriting Operation

Style	Sentences
Complex Sentence	origin irmo was chartered on christmas eve in 1890 in response to the opening of the columbia , newberry and laurens railroad .
Promoting Reordering	in 1890 , the origin irmo was chartered on christmas eve . this is a response to the opening of the columbia , newberry and laurens railway .

Table 36: Sample of Influence of Tuning Reordering Rewriting Operation

Style	Sentences
Complex Sentence	these attacks may have been psychological in origin rather than physical .
Promoting Reordering	these attacks may have been of psychological origin , not of physical origin .

Table 37: Sample of Influence of Tuning Reordering Rewriting Operation

Style	Sentences
Complex Sentence	the britannica was primarily a scottish enterprise , as symbolised by its thistle logo , the floral emblem of scotland .
Promoting Reordering	the britannica was primarily a scottish enterprise . it was symbolized by the coat of arms .

Table 38: Sample of Influence of Tuning Reordering Rewriting Operation

Style	Sentences
Complex Sentence	alessandro (" sandro ") mazzola (born 8 november 1942) is an italian former football player .
Promoting Reordering	mazzola is a former football player .

Table 39: Sample of Influence of Tuning Reordering Rewriting Operation

Style	Sentences
Complex Sentence	as of 2000 , the population was 89,148 .
Promoting Reordering	as of 2000 , the city had a population .

2. **Updating the Order**: As shown in Tables 34 and 35, the reordered sentence moves the word "sydney" to the beginning, which is a more natural expression.
3. **Substitution**: As shown in Table 36, the reordered sentence replaces "rather than' with "not of," a simpler expression. It overlaps with the substitution rewriting operation because sometimes the replacement phrase is also a change in sentence structure.
4. **Splitting**: As shown in Table 37, the reordered sentence split the long sentence into two shorter sentences. It overlaps with the splitting rewriting operation because sometimes the splitting is also a change in sentence structure.
5. **Dropping**: As shown in Table 38, the reordered sentence removes unimportant components from a complex sentence. It overlaps with the dropping rewriting operation because sometimes the dropping is also a change in sentence structure.
6. **Adding**: As shown in Table 39, the reordered sentence adds the subject "city." Although it may have nothing to do with this complex sentence, the word "city" makes the sentence clearer by providing more context.

6.4 Evaluation

This section evaluates the effectiveness of my tuning models in the sentence simplification system.

Table 40: the Effectiveness of Tuning Rewriting Operations. * denotes p < 0.001

Style	Pearson Correlation
Substitution	0.90*
Dropping	0.99*
Reordering	0.96*

Regarding the substitution, dropping, and reordering rewriting operations, I calculated the Pearson correlation between the tuning scale and the specified rewriting operation measurement. As shown in Table 40, they all significantly correlated with the tuning scales.

Dropping may be the easiest style to tune because the model only needs to learn when to stop generation. With the help of the syntax-aware tunable sentence simplification model, the effectiveness of tuning the reordering rewriting operation is also high. Due to the limitation of available PPDB rules, tuning the substitution effect will be slightly worse. For the splitting operation, I force the model to split complex sentences and the model split 98.61 % of the sentences.

However, I found that the tuning style is not helpful in improving SARI because it is difficult to find a combination of tuning parameters (scale) that achieves a significantly better SARI score. This finding conflicts with the conclusion of ACCESS [51]. In their work, they use a WikiLarge dataset, and tuning may exclude the noisy sentence. But noisy sentences are rare in SimSim, so the tuning has hardly improved in my case. Another limitation is my linear regressors are not accurate enough to provide appropriate style scores. Such limitation makes tuning towards better SARI harder.

6.5 Summary

With the help of syntax-aware tunable sentence simplification, the experiment shows that my tunable sentence simplification model is effective for all four simplification rewriting operations. My syntax-aware tunable sentence simplification model can determine different types of sentence structure simplification, particularly reordering operations.

7.0 Discussion and Conclusion

7.1 Achievements

In this work, I explored sentence simplification through four simplification rewriting operations: **substitution**, **dropping**, **reordering**, and **splitting** [102, 98]. The substitution operation replaces difficult phrases or words with simplified synonyms. The dropping operation removes unimportant or redundant parts of a sentence to improve conciseness. The reordering operation reorders components to simplify sentence structure and syntax. The splitting operation divides a long sentence into several shorter sentences to reduce the original sentence's complexity. My dissertation focused on these four operations; the remaining work is based on the four rewriting operations.

I designed measurements to analyze the existing training datasets and determined that all of the datasets (1) contain a number of poorly aligned sentence pairs, (2) are limited in several ways and contribute to only some operations, and (3) excessively focus on one operation that may not be of interest to some users. These conclusions prompted me to construct my own dataset SimSim.

The initial SimSim is constructed with back translation and the unconditional language model GPT-2. Then, SimSim is further improved for the substitution simplification rewriting operation with the help of PPDB. The WikiSplit dataset is used to enhance SimSim for the splitting simplification rewriting operation. The dropping simplification rewriting operation of SimSim is also improved using the rule-based approach by randomly removing words. The dataset analysis shows that SimSim demonstrates (1) a closer distribution on multiple dimensions to the human-annotated Turk and ASSET dataset than the others, suggesting that SimSim may serve as a better dataset for training automatic models; and (2) a sufficient number of samples to teach the model certain operations, such as substitution and splitting.

Although some noise was introduced from the data construction, I introduced my model design to reduce the influence of fault from data construction. The experiment shows that models trained by SimSim achieve state-of-the-art performance regarding SARI. My ablation studies also demonstrate the way constructing SimSim improves the performance gradually. However, those simplification simulations also have certain limitations due to the limited resources available online, especially for dropping operations.

Due to the limitations of the back translation, the model occasionally neglected certain simplification rewriting operations (such as name entities guilded sentence simplification model). This shortcoming prompted me to explore tunable sentence simplification models. My initial tunable sentence simplification model worked well except for tuning the reordering operation; however, encoding syntax information into the model improved its effectiveness. With the help of syntax-aware tunable sentence simplification, the experiment shows that a tunable model is effective for all four simplification rewriting operations. My model can determine different types of sentence structure simplification, particularly reordering operations.

The sections that follow discuss limitations and future work.

7.2 Discussion

7.2.1 Analyzing the Existing Corpora and the Constructed `SimSim`

As discussed in Section 3.1.1, the relevance analysis indicates that all existing corpora contain a certain number of poorly aligned sentence pairs resulting from edits intended to rewrite a text for a different readability level, which may drastically reconstruct its words, sentences, or even paragraphs. Consequently, sentences in paraphrased articles may not accurately pair with the original sentences. The misalignment may worsen when sentences are automatically extracted and aligned with simple lexical-based features such as Jaccard similarity. The misalignment can come from poorly performed extraction and alignment algorithms or from a poorly aligned article itself. Such misalignment error is unavoidable if extracting from existing one-line resources due to limited resources. My approach collects sentence pairs in another way, namely, using existing paraphrase systems to generate simplified sentences. There are different choices of such paraphrase systems: In this work, I used

Google Neural Machine Translation and a back-translation approach to rewrite a text at the sentence level. Although this method provided me strongly aligned sentence pairs, the changes from paraphrased sentences are limited. Most of the sentence pairs contributed to substitution and reordering operations; a limited number of pairs contributed to dropping and splitting rewriting operations. Therefore, the improvement from my data at stage 1 and 2 is smaller as shown in Secion 5.2.2. One of my future works will check multiple paraphrase systems to generate varying sentence pairs to include more rewriting operations in the data. One related work [48] trained separate models for splitting and dropping rewriting operations.

In this work, I explored several approaches to further edit generated sentence pairs to include more rewriting operations.

Substitution analysis results indicate more words will be substituted because substitution always results in a simpler output. I found the SimSim performs slightly better than other existing corpora except for Newsela-Auto. The inferior performance of SimSim is because GPT-2 fails to capture multiple substitution mapping. GPT-2 evaluates a text at the sentence level and may neglect a substituted word or phrase. Table 41 shows one example from the testing dataset that GPT-2 assigned a low loss to a clearly complex sentence. The sentence pairs contain three substitution mappings, including flexible→moveable, hollow→empty, and interior→insides. Future work will involve fine-tuning GPT-2 with human-provided, aligned sentence pairs to encourage consideration of the substitution mapping that would result in GPT-2 assigning a loss in similar situations. Additionally, I will consider putting Newsela-Auto into the seed sentence bank to further improve substitution rewriting operations.

Table 41: Failures of GPT-2: Samples come from testing dataset.

	Sentences	GPT-2 Loss
Complex	Bone marrow is the flexible tissue found in the hollow interior of bones .	3.0145
Simplified	Bone marrow is the moveable tissue found in the empty insides of bones .	3.7665

Splitting analysis results indicate the number of sentence pairs contributing to the splitting operation is large enough that the model can learn how to split. I introduced WikiSplit into my seed sentence bank to simulate the splitting operation. As a result, SimSim contains 39.9 % sentence pairs, which I believe is enough to teach the model splitting. The high ratio of the split sentences is due to the WikiSplit dataset in the seed sentence bank. The WikiSplit dataset is constructed by checking edit history to learn how humans split a long sentence. Unlike the dataset constructed by back-translation, the dataset is written by Wikipedia users, reflecting a natural way to split a long sentence. However, the dataset only contributes to splitting and somehow reordering operations. It is necessary to explore other simulating rewriting operation approaches to maximize the contribution of rewriting operations for this dataset.

To simulate the dropping operation, I randomly remove certain functional phrases from the complex sentence, such as prepositional, adjective, or adverb phrases. These deletions are safe because removing these functional phrases will not change the sentence significantly, even though it may not be the preferred method if considerable modification is required. Another reason for a few large dropping in SimSim dataset is due to GPT-2. Table 42 suggests that GPT-2 may be likely to assign a low loss for a long sentence because it contains common words to lower the averaged GPT-2 loss. This is contradictory to the goal of sentence simplification. As I indicated before, fine-tuning of GPT-2 is needed. My initial approach was similar in that I simulated splitting operations by introducing a sentence compression dataset into the seed sentence bank. However, the performance was inferior because the model drops words aggressively. Future work will include combining two dropping approaches to enable different styles of dropping operations.

I did not determine an effective method for simulating reordering operations. The model learns to reorder words mostly from back-translation and the WikiSplit dataset. Unlike the other three rewriting operations, there is no consensus about which types of sentence structure are simpler. Some reordering examples in Section 6.3 are not widely accepted. I will explore better approaches to simulate and evaluate reordering operations.

Summarizing the simulation of rewriting operations is novel and important because while there are numerous NLP resources available online for other tasks, many only contribute to

Table 42: Failures of GPT-2: Samples come from testing dataset.

	Sentences	GPT-2 Loss
Complex	You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version.	1.6899
Simplified	In the Modified Version you may add 25 words to the front and back covers.	4.3446
Complex	Bankers from ShoreBank, a community development bank in Chicago, helped Yunus with the official incorporation of the bank under a grant from the Ford Foundation.	3.8089
Simplified	Yumus gained help to incorporate the bank from ShoreBank and the Ford Foundation.	5.7520

certain rewriting operations. As far as I know, there are no substantial written resources for this task. Therefore, using existing resources and simulate them to the four rewriting operations is helpful.

7.2.2 Model Design of `SimSim`

Due to the large vocabulary size for SIMSIM, it is necessary to use subword tokenization and name entities recognition pipeline. My initial experiment demonstrated an extremely slow optimization, and the model never fully converged without this pipeline. The reason is that the model takes time to learn the meaning of different named entities. However, I argue that this is unnecessary for the sentence simplification system because the model could copy the same name entities into a simplified sentence. Therefore, in the data pipeline, I combine the subword tokenization and named entities replacement. This step is helpful for optimization. Some recent works, such as [51], describe good performance after using subword tokenization with a small subword vocabulary and without a named entity recognizer. However, because my dataset SIMSIM was large and contained many named entities, the name entity recognizer was indispensable.

In my model design, I have named-entities guild sentence simplification and PPDB guild sentence simplification. The goal of such model design is to reduce the influence of fault from data construction. Such model design should be ignored if a better data construction approach is introduced in the future.

The experimental results show that the model trained by SIMSIM significantly outperforms other models regarding the SARI. The better performance is primarily due to adding factors in computing SARI. However, the deleting factor has an inferior performance. The good performance is also due to the nature of the testing dataset. The TURK dataset focuses more on substitution rewriting operations [1] and less on other rewriting operations, which is why there is less improvement for the ASSET dataset. Although I simulated dropping operations, the model simply averages the contributions of the entire training dataset, weakening the contributions to simplification operations. However, accurate measurement of contributions can be improved by tunable sentence simplification.

7.2.3 Tunable Sentence Simplification

Tunable sentence simplification provides users with flexibility. On the user side, it allows the user to provide input to tune the style of the simplified sentence. On the model side, it allows for simplified sentence generation from imperfect data through the full contribution of four rewriting operations. Therefore, the tunable sentence simplification is helpful to make model generation fully contribute to four rewriting operations.

Although the model can easily understand and tune most styles, there may be some deep features, such as sentence structure, that the model is not familiar with, leading to poor tunable sentence simplification performance. Thus, simply encoding this information into the model can resolve the failure. With the help of the syntax-aware tunable sentence simplification, all four rewriting operations are effective in tuning.

Conversely, over-tuning sentences could generate unreadable text. For example, if significant dropping operations are suggested, the model will drop the complex sentence entirely. My future work will involve reducing the possibility of generating unreadable sentences with unpredictable user inputs and exploring ways to automatically tune the styles based on a sample dataset.

Finally, while my styles are strongly correlated with four rewriting operations, additional styles may further improve sentence readability. For example, a relevance style score can be used to avoid generating unreadable sentences. ACCESS [51] explores different styles with different datasets to tune their sentence generation for a better SARI. In their work, they used normalized character-level Levenshtein similarity to filter out the less-correlated sentence pairs. My tunable sentence simplification model has a similar goal to the ACCESS [51] but achieves different results. Although we used different data and style definitions, resolving the conflicts is crucial. In the future, I will explore more simplification styles to improve sentence readability.

7.2.4 Summarizing SimSim and Model Design

Although SimSim is constructed in a way that results in fewer misalignments and simulates four simplification rewrite operations, it still has limitations.

The benefit of back translation is to provide large numbers of strongly aligned sentence pairs. Because SimSim is bigger than any of the existing publicly available training data sets and covers more text domains, I believe that a model trained by SimSim could generalize well to different situations. Additionally, SimSim contains far fewer misaligned sentence pairs so a model based on it is less likely to generate an unreadable sentence.

However, the changes made in the sentence pairs generated from back translation are limited. In my experience, back translation provides a sentence with a significant amount of substitution but performs poorly on other rewrite operations. With further simulation of the substitution rewrite operation, it is possible to merge substitution mapping from multiple back-translated sentences to maximize the potential of the substitution rewrite operation.

Due to the incorporation of the WikiSplit dataset, the process of splitting and rephrasing is also well simulated. Therefore, SimSim performs well in both substitution and split-and-rephrase operations. But it is less effective at simulating word dropping and reordering rewrite tasks.

Although I can simulate dropping by randomly removing certain functional words from the sentence, in most cases, the dropping is limited to deleting a few words. Also, I didn't determine an effective method for simulating reordering operations. Furthermore, GPT-2 fails to capture a simplified sentence with dropping and reordering rewrite operations due to the limitations in the way sentence pairs are validated.

In summary, SimSim contains strongly aligned sentence pairs and contributes significantly to substitution and split operations. However, it performs poorly on dropping and reordering rewriting operations. In my future work, I will focus more on dropping and reordering rewrite operations.

If high-quality sentence pairs are provided, the model design is, in a way, irrelevant. In this dissertation, my first goal for the model was to reduce the number of faults caused by the data construction. For example, the use of name entities to guide sentence simplification reduces the number of mismatched name entities in sentence pairs. Tunable sentence simplification aims to generate a sentence with a style that is different from that of the training data to maximize the simplification of the rewrite operations. My second goal was to make this model easier to optimize.

7.3 Future Work

In this section, I summarize my plans for future work on this model.

- Because GPT-2 fails to capture multiple substitution mapping and mistakenly favors longer sentences as shown in Table 41 and 42, future work will involve fine-tuning GPT-2 with human-provided, aligned sentence pairs (such as Wiki-Manual[33]) to encourage consideration of multiple factors in the sentence-simplification task. The fine-tuned GPT-2, as a high-quality, complex-simple sentence pairs validation tool, is a prerequisite of all the rest of my work.
- For split rewriting operation, WikiSplit, constructed by checking edit history to learn how humans split a long sentence, provides a good resource to teach the model how to split. As SimSim performs poorly on dropping and reordering rewrite operations, I need to explore more resources for improving its ability to handle these tasks as well.
- Back translation inhibits the generation of diverse paraphrased sentences. In my future research, I will explore multiple paraphrase systems to generate varied sentence pairs to include more rewrite operations in the data. In a related study, [48] researchers trained separate models for splitting and dropping rewrite operations to produce diverse sentences for multiple rewrite operations.
- Tunable sentence simplification is another issue that must be addressed. Overtuning sentences generates unreadable text. For example, if significant dropping operations are suggested, the model will drop complex sentences entirely. Future work will involve reducing the possibility of generating unreadable sentences when user inputs are unpredictable and exploring ways to automatically tune the styles based on a referenced dataset.
- In my tunable sentence-simplification simulation, the rewriting styles are strongly correlated with four rephrasing operations. Adding additional rewriting styles may further improve sentence readability. A related study, [51], explores different rewriting styles, used with different datasets, to tune sentence generation for a better SARI. Normalized character-level Levenshtein similarity was used to filter out the less-correlated sentence

pairs. In the future, I will explore simplification styles other than the four rewrite operations to improve sentence readability.

Bibliography

[1] 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.

[2] Colin Bannard and Chris Callison-Burch. Paraphrasing with bilingual parallel corpora. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 597–604. Association for Computational Linguistics, 2005.

[3] Regina Barzilay and Kathleen R McKeown. Extracting paraphrases from a parallel corpus. In *Proceedings of the 39th annual meeting of the Association for Computational Linguistics*, 2001.

[4] Jan A Botha, Manaal Faruqui, John Alex, Jason Baldridge, and Dipanjan Das. Learning to split and rephrase from wikipedia edit history. *arXiv preprint arXiv:1808.09468*, 2018.

[5] John A Carroll, Guido Minnen, Darren Pearce, Yvonne Canning, Siobhan Devlin, and John Tait. Simplifying text for language-impaired readers. In *EACL*, pages 269–270, 1999.

[6] Asli Celikyilmaz, Antoine Bosselut, Xiaodong He, and Yejin Choi. Deep communicating agents for abstractive summarization. *arXiv preprint arXiv:1803.10357*, 2018.

[7] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*, 2018.

[8] Raman Chandrasekar, Christine Doran, and Bangalore Srinivas. Motivations and methods for text simplification. In *Proceedings of the 16th conference on Computational linguistics-Volume 2*, pages 1041–1044. Association for Computational Linguistics, 1996.

[9] Wenhu Chen, Evgeny Matusov, Shahram Khadivi, and Jan-Thorsten Peter. Guided alignment training for topic-aware neural machine translation. *arXiv preprint arXiv:1607.01628*, 2016.

[10] James Clarke and Mirella Lapata. Models for sentence compression: A comparison across domains, training requirements and evaluation measures. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 377–384. Association for Computational Linguistics, 2006.

[11] James Clarke and Mirella Lapata. Global inference for sentence compression: An integer linear programming approach. *Journal of Artificial Intelligence Research*, 31:399–429, 2008.

[12] 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: short papers-Volume 2*, pages 665–669. Association for Computational Linguistics, 2011.

[13] Zihang Dai, Zhilin Yang, Yiming Yang, William W Cohen, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

[14] 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.

[15] Bill Dolan, Chris Quirk, and Chris Brockett. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of the 20th international conference on Computational Linguistics*, page 350. Association for Computational Linguistics, 2004.

[16] William B Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, 2005.

[17] Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation. *arXiv preprint arXiv:1805.04833*, 2018.

[18] Lijun Feng. Text simplification: A survey. *The City University of New York, Tech. Rep*, 2008.

[19] Jessica Ficler and Yoav Goldberg. Controlling linguistic style aspects in neural language generation. *arXiv preprint arXiv:1707.02633*, 2017.

[20] Katja Filippova, Enrique Alfonseca, Carlos A Colmenares, Lukasz Kaiser, and Oriol Vinyals. Sentence compression by deletion with lstms. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 360–368, 2015.

[21] Katja Filippova and Yasemin Altun. Overcoming the lack of parallel data in sentence compression. 2013.

[22] Katja Filippova and Michael Strube. Sentence fusion via dependency graph compression. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 177–185. Association for Computational Linguistics, 2008.

[23] Philip Gage. A new algorithm for data compression. *C Users Journal*, 12(2):23–38, 1994.

[24] Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. Ppdb: The paraphrase database. In *HLT-NAACL*, pages 758–764, 2013.

[25] Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. Creating training corpora for nlg micro-planning. In *55th annual meeting of the Association for Computational Linguistics (ACL)*, 2017.

[26] Caglar Gulcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loic Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. On using monolingual corpora in neural machine translation. *arXiv preprint arXiv:1503.03535*, 2015.

[27] Han Guo, Ramakanth Pasunuru, and Mohit Bansal. Dynamic multi-level multi-task learning for sentence simplification. *arXiv preprint arXiv:1806.07304*, 2018.

[28] Wei He, Zhongjun He, Hua Wu, and Haifeng Wang. Improved neural machine translation with smt features. In *Thirtieth AAAI conference on artificial intelligence*, 2016.

[29] Michael Heilman and Noah A Smith. Question generation via overgenerating transformations and ranking. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA LANGUAGE TECHNOLOGIES INST, 2009.

[30] M Honnibal. Introducing spacy, 2003.

[31] J Edward Hu, Rachel Rudinger, Matt Post, and Benjamin Van Durme. Parabank: Monolingual bitext generation and sentential paraphrasing via lexically-constrained neural machine translation. *arXiv preprint arXiv:1901.03644*, 2019.

[32] 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*, pages 9–16. Association for Computational Linguistics, 2003.

[33] Chao Jiang, Mounica Maddela, Wuwei Lan, Yang Zhong, and Wei Xu. Neural crf model for sentence alignment in text simplification. In *Proceedings of the Association for Computational Linguistics (ACL)*, 2020.

[34] Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351, 2017.

[35] Siddhartha Jonnalagadda and Graciela Gonzalez. Sentence simplification aids protein-protein interaction extraction. *arXiv preprint arXiv:1001.4273*, 2010.

[36] Dan Jurafsky and James H Martin. *Speech and language processing*, volume 3. Pearson London, 2014.

[37] Nobuhiro Kaji, Daisuke Kawahara, Sadao Kurohash, and Satoshi Sato. Verb paraphrase based on case frame alignment. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 215–222. Association for Computational Linguistics, 2002.

[38] Yuta Kikuchi, Graham Neubig, Ryohei Sasano, Hiroya Takamura, and Manabu Okumura. Controlling output length in neural encoder-decoders. *arXiv preprint arXiv:1609.09552*, 2016.

[39] J Peter Kincaid, Robert P Fishburne Jr, 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, Naval Technical Training Command Millington TN Research Branch, 1975.

[40] Kevin Knight and Daniel Marcu. Statistics-based summarization-step one: Sentence compression. *AAAI/IAAI*, 2000:703–710, 2000.

[41] Kevin Knight and Daniel Marcu. Summarization beyond sentence extraction: A probabilistic approach to sentence compression. *Artificial Intelligence*, 139(1):91–107, 2002.

[42] Catherine Kobus, Josep Crego, and Jean Senellart. Domain control for neural machine translation. *arXiv preprint arXiv:1612.06140*, 2016.

[43] Jiwei Li, Michel Galley, Chris Brockett, Georgios P Spithourakis, Jianfeng Gao, and Bill Dolan. A persona-based neural conversation model. *arXiv preprint arXiv:1603.06155*, 2016.

[44] Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, and Dan Jurafsky. Adversarial learning for neural dialogue generation. *arXiv preprint arXiv:1701.06547*, 2017.

[45] Dekang Lin and Patrick Pantel. Discovery of inference rules for question-answering. *Natural Language Engineering*, 7(4):343–360, 2001.

[46] Yizhu Liu, Zhiyi Luo, and Kenny Zhu. Controlling length in abstractive summarization using a convolutional neural network. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4110–4119, 2018.

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

[48] Mounica Maddela, Fernando Alva-Manchego, and Wei Xu. Controllable text simplification with explicit paraphrasing. *arXiv preprint arXiv:2010.11004*, 2020.

[49] Mounica Maddela and Wei Xu. A word-complexity lexicon and a neural readability ranking model for lexical simplification. *arXiv preprint arXiv:1810.05754*, 2018.

[50] Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60, 2014.

[51] Louis Martin, Benoît Sagot, Eric de la Clergerie, and Antoine Bordes. Controllable sentence simplification. *arXiv preprint arXiv:1910.02677*, 2019.

[52] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*, 2010.

[53] Shashi Narayan and Claire Gardent. Hybrid simplification using deep semantics and machine translation. In *the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 435–445, 2014.

[54] Shashi Narayan, Claire Gardent, Shay B Cohen, and Anastasia Shimorina. Split and rephrase. *arXiv preprint arXiv:1707.06971*, 2017.

[55] Sergiu Nisioi, Sanja Štajner, Simone Paolo Ponzetto, and Liviu P Dinu. Exploring neural text simplification models. In *Proceedings of the 55th annual meeting of the association for computational linguistics (volume 2: Short papers)*, pages 85–91, 2017.

[56] Tong Niu and Mohit Bansal. Polite dialogue generation without parallel data. *Transactions of the Association of Computational Linguistics*, 6:373–389, 2018.

[57] Joakim Nivre, Marie-Catherine De Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan T McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. Universal dependencies v1: A multilingual treebank collection. In *LREC*, 2016.

[58] Ellie Pavlick and Chris Callison-Burch. Simple ppdb: A paraphrase database for simplification. In *The 54th Annual Meeting of the Association for Computational Linguistics*, page 143, 2016.

[59] Ellie Pavlick, Pushpendre Rastogi, Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. Ppdb 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 425–430, 2015.

[60] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

[61] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazon-*

aws. com/openai-assets/research-covers/languageunsupervised/language understanding paper. pdf, 2018.

[62] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *URL https://openai. com/blog/better-language-models*, 2019.

[63] Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*, 2015.

[64] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

[65] Luz Rello, Clara Bayarri, Azuki Gòrriz, Ricardo Baeza-Yates, Saurabh Gupta, Gaurang Kanvinde, Horacio Saggion, Stefan Bott, Roberto Carlini, and Vasile Topac. Dyswebxia 2.0!: more accessible text for people with dyslexia. In *Proceedings of the 10th International Cross-Disciplinary Conference on Web Accessibility*, page 25. ACM, 2013.

[66] Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel. Self-critical sequence training for image captioning. In *CVPR*, volume 1, page 3, 2017.

[67] Holger Schwenk. Investigations on large-scale lightly-supervised training for statistical machine translation. In *International Workshop on Spoken Language Translation (IWSLT) 2008*, 2008.

[68] Abigail See, Peter J Liu, and Christopher D Manning. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*, 2017.

[69] Rico Sennrich, Barry Haddow, and Alexandra Birch. Improving neural machine translation models with monolingual data. *arXiv preprint arXiv:1511.06709*, 2015.

[70] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

[71] Rico Sennrich, Barry Haddow, and Alexandra Birch. Controlling politeness in neural machine translation via side constraints. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 35–40, 2016.

[72] Advaith Siddharthan. A survey of research on text simplification. *ITL-International Journal of Applied Linguistics*, 165(2):259–298, 2014.

[73] Anuroop Sriram, Heewoo Jun, Sanjeev Satheesh, and Adam Coates. Cold fusion: Training seq2seq models together with language models. *arXiv preprint arXiv:1708.06426*, 2017.

[74] Yui Suzuki, Tomoyuki Kajiwara, and Mamoru Komachi. Building a non-trivial paraphrase corpus using multiple machine translation systems. In *Proceedings of ACL 2017, Student Research Workshop*, pages 36–42, 2017.

[75] Shunsuke Takeno, Masaaki Nagata, and Kazuhide Yamamoto. Controlling target features in neural machine translation via prefix constraints. In *Proceedings of the 4th Workshop on Asian Translation (WAT2017)*, pages 55–63, 2017.

[76] Kristina Toutanova, Chris Brockett, Ke M Tran, and Saleema Amershi. A dataset and evaluation metrics for abstractive compression of sentences and short paragraphs. 2016.

[77] Zhaopeng Tu, Yang Liu, Zhengdong Lu, Xiaohua Liu, and Hang Li. Context gates for neural machine translation. *Transactions of the Association for Computational Linguistics*, 5:87–99, 2017.

[78] Zhaopeng Tu, Yang Liu, Lifeng Shang, Xiaohua Liu, and Hang Li. Neural machine translation with reconstruction. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[79] Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. Modeling coverage for neural machine translation. *arXiv preprint arXiv:1601.04811*, 2016.

[80] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

[81] David Vickrey and Daphne Koller. Sentence simplification for semantic role labeling. *Proceedings of ACL-08: HLT*, pages 344–352, 2008.

[82] Tu Vu, Baotian Hu, Tsendsuren Munkhdalai, and Hong Yu. Sentence simplification with memory-augmented neural networks. *arXiv preprint arXiv:1804.07445*, 2018.

[83] Willian Massami Watanabe, Arnaldo Candido Junior, Vinícius Rodriguez Uzêda, Renata Pontin de Mattos Fortes, Thiago Alexandre Salgueiro Pardo, and Sandra Maria Aluísio. Facilita: reading assistance for low-literacy readers. In *Proceedings of the 27th ACM international conference on Design of communication*, pages 29–36. ACM, 2009.

[84] Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei-Hao Su, David Vandyke, and Steve Young. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. *arXiv preprint arXiv:1508.01745*, 2015.

[85] John Wieting and Kevin Gimpel. Paranmt-50m: Pushing the limits of paraphrastic sentence embeddings with millions of machine translations. *arXiv preprint arXiv:1711.05732*, 2017.

[86] John Wieting, Jonathan Mallinson, and Kevin Gimpel. Learning paraphrastic sentence embeddings from back-translated bitext. *arXiv preprint arXiv:1706.01847*, 2017.

[87] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[88] Kristian Woodsend and Mirella Lapata. Learning to simplify sentences with quasi-synchronous grammar and integer programming. In *Proceedings of the conference on empirical methods in natural language processing*, pages 409–420. Association for Computational Linguistics, 2011.

[89] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[90] Sander Wubben, Antal Van Den Bosch, and Emiel Krahmer. Sentence simplification by monolingual machine translation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 1015–1024. Association for Computational Linguistics, 2012.

[91] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057, 2015.

[92] Wei Xu, Chris Callison-Burch, and Bill Dolan. Semeval-2015 task 1: Paraphrase and semantic similarity in twitter (pit). In *Proceedings of the 9th international workshop on semantic evaluation (SemEval 2015)*, pages 1–11, 2015.

[93] Wei Xu, Chris Callison-Burch, and Courtney Napoles. Problems in current text simplification research: New data can help. *Transactions of the Association of Computational Linguistics*, 3(1):283–297, 2015.

[94] 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.

[95] Wei Xu, Alan Ritter, Chris Callison-Burch, William B Dolan, and Yangfeng Ji. Extracting lexically divergent paraphrases from twitter. *Transactions of the Association for Computational Linguistics*, 2:435–448, 2014.

[96] Quanzeng You, Hailin Jin, Zhaowen Wang, Chen Fang, and Jiebo Luo. Image captioning with semantic attention. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4651–4659, 2016.

[97] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*, 2019.

[98] Xingxing Zhang and Mirella Lapata. Sentence simplification with deep reinforcement learning. *arXiv preprint arXiv:1703.10931*, 2017.

[99] Sanqiang Zhao, Rui Meng, Daqing He, Saptono Andi, and Parmanto Bambang. Integrating transformer and paraphrase rules for sentence simplification. *arXiv preprint arXiv:1810.11193*, 2018.

[100] Sanqiang Zhao, Piyush Sharma, Tomer Levinboim, and Radu Soricut. Informative image captioning with external sources of information. *arXiv preprint arXiv:1906.08876*, 2019.

[101] Shiqi Zhao, Xiang Lan, Ting Liu, and Sheng Li. Application-driven statistical paraphrase generation. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 834–842. Association for Computational Linguistics, 2009.

[102] 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*, pages 1353–1361. Association for Computational Linguistics, 2010.