Bridging the Gap Between Text Simplification and Summarization

Course: 2021-201300074-2B, Group: 13, Submission Date: July 6, 2022

Manish Kumar Mishra
University of Twente
The Netherlands
m.k.mishra@student.utwente.nl

Peshmerge Morad
University of Twente
The Netherlands
p.morad@student.utwente.nl

ABSTRACT

Text summarization and simplification are related NLP tasks that have always been done individually. In our project, we aim to bridge the gap between text simplification and summarization. We observe the effects of the order that each step has on a document by proposing two pipelines for generating a simplified summary by changing the order of simplification and summarization. We use a pre-trained BERT based simplification model and a PageRank based extractive summarization model for our project. We evaluate the outputs of our pipelines by comparing the ROUGE and WMD scores to determine the effects of the order. Finally, we conclude that the order of simplification and summarization does not matter.

KEYWORDS

Text summarization, Text simplification, NLP, BERT, PageRank, ROUGE, WMD

1 INTRODUCTION

Massive amounts of textual data are generated with the exponential increase in internet use, be it accessing websites, search engines, hybrid web applications, or social networks. This growth has also resulted in exhaustive research in natural language processing (NLP) and information retrieval (IR) applications. Due to such large amounts of data, the need to simplify and make concise information accessible becomes essential. Text simplification and text summarization are two of the various NLP tasks that, as the names suggest, aim to make textual information more accessible to everyone on the internet.

Text simplification makes any given text easier to read and understand for children, non-native speakers, and people with cognitive disabilities such as aphasia, dyslexia, or autism by modifying the content and simplifying the structure and the grammar of the text. It reduces the linguistic complexity of the text while retaining the original text's meaning in a simplified manner [3, 19, 34, 37, 42]. Traditionally, the task of text simplification has been performed by language experts and has been costly and time-consuming. Therefore, automated data-driven approaches to text simplification have been researched for as long as 20 years [12]. Text simplification can be done on a sentence level [3] and on a document level [37] as well. In another work, Zong et al. [43] train a classification model to analyze and predict sentence deletion for text simplification using manually annotated English data.

Text summarization, on the other hand, aims to reduce the content and the length of any given text by removing redundant information and filler text while conveying the key information from the original text. In general, as a simple and non-rigid definition, the summary of a text would be no longer than half of the input text and many times much less than that [7].

For the task of text summarization, there are two major approaches: 1) ***extractive summarization***, where salient information of the original text is identified, extracted and concatenated together, and 2) ***abstractive summarization***, where novel sentences are being generated from the extracted information from the original text using internal semantic representation. Both approaches can be performed using supervised learning methods. Automated abstractive summarization requires advanced NLP techniques such as Natural Language Generation (NLG) [1, 11, 18]. However, extractive summarization is simpler than abstractive summarization because the former can be performed using unsupervised learning, eliminating the need for external knowledge or training. Graph-based models [21] including LexRank [8] and PageRank [28], and Latent Semantic Analysis Method (LSA) [20, 26] are used for unsupervised extractive summarization.

Another important aspect of automated text simplification and summarization approaches is evaluating the results. The evaluation process is based on document similarity measures. We have two main types of document similarity: lexical and semantic similarity. Lexical similarity depends on the words and the word overlap between vocabularies. Semantic similarity focuses on the semantic meaning rather than the documents' structure or vocabulary. Recall-Oriented Understudy for Gisting Evaluation (ROUGE) [17], BiLingual Evaluation Understudy (BLEU) [29], System output Against References and Input sentence (SARI) [40] are lexical based similarity metrics. On the other hand, Word Movers Distance (WMD) [16] is a semantic similarity metric. In this project, we use ROUGE and WMD to evaluate the output of our pipelines.

The research community has often approached automated text simplification and summarization as separate tasks without much exploration in the potential of combining the two. In addition, the current approaches for automated text simplification consider the problem at the sentence level. Such an approach has its own benefits: 1) easier to collect suitable data, 2) utilizing existing methods from the field of machine translation, and 3) developing proper and robust evaluation methods. However, generating simplified summaries cannot be achieved by simplifying one sentence at a time. The task must be performed at the document-level [2, 44].

Merging the two processes can prove extremely useful in generating more straightforward text for all audiences. Sometimes,

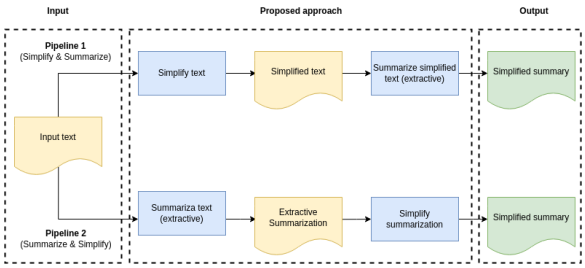

Figure 1: Our proposed approach for generating simplified summaries

summarized texts might still be complex to understand, and simplified text might not be concise. In this project, we aim to merge the processes of text simplification and summarization to aid in both the tasks and create simplified summaries of a given input text.

To narrow down the scope of the project, we have formulated the following research question:

"***What effect does the order of simplification and summarization have on generating a simplified summary of a text?***"

To be able to answer this question, we have specified the following two sub-questions:

- *which order provides better results?*
- *how can the simplified summaries be evaluated?*

Fig. 1 demonstrates our approach for generating simplified summaries:

(1) **Pipeline 1** *(P1)*: applying text simplification first and then summarizing the simplified text.
(2) **Pipeline 2** *(P2)*: applying text summarization first and then simplifying the summarized text.

We use the Wiki-auto aligned data generated by Jiang et al. [13] for our project. For text simplification, we use BERT base model that is pre-trained on Wikipedia [13]. For summarization, we use an unsupervised approach using the PageRank algorithm to select the most important sentences in the text. Finally, we evaluate the outputs from both pipelines using ROUGE and WMD and compare the scores to conclude which pipeline delivers the highest score.

This paper is organized as follows. In Section 2, we give an overview of the research that has already been done for automated text simplification and summarization, individually and jointly. In Section 3 we describe the dataset we use for this project and present a simple analysis. In Section 4, we give a detailed description of the task, data preparation and model implementation and evaluation. We present and discuss the project's results in Section 5. Finally, we provide a conclusion and discuss potential future work recommendations in Section 6.

2 RELATED WORK

Text simplification and text summarization are major ways to simplify a given text. They are related yet have different objectives; the former aims to reduce the linguistic complexity of the text, while the latter focuses on reducing the number of sentences in any given text. Both tasks are well-researched topics in the field of NLP [7, 12].

The first text simplification systems were rule-based, focusing significantly on syntactic simplification. The early systems were often a preprocessing step in various NLP applications like machine translation, parsing, and information retrieval [35]. With the latest developments in the field of machine learning (ML) and deep learning (DL), the task of text simplification encompassed the process of simplification of higher levels, such as lexical and syntactic simplification and explanation generation [3, 27]. However, the early text simplification models were based on machine-translation approaches as researched by [23, 39]. The most advances in simplification are achieved using DL and deep reinforcement learning approaches such as using Recurrent Neural Networks (RNNs) as shown by [25, 38, 42].

Within the text summarization task, we have two types of summarization: extractive and abstractive. For both summarization types, many approaches have been proposed. Some researchers have used standard Naive Bayes classifiers or Support Vector Machine [4, 24]. Other researchers have used an unsupervised approach [9, 10]. The more recent techniques for text summarization utilize the power of DL and RNN, namely Long-Short Term Memory (LSTM), to achieve the best results as shown in [18, 22, 31, 33, 36].

However, up and until now, the research community has been mainly focusing on text simplification and summarization individually. To our knowledge, only a handful of researchers researched the tasks of text simplification and summarization jointly. Researchers in [41] have combined both tasks using an end-to-end approach. They have proposed a composite loss function for simplification and summarization by extending the pointer generator architecture proposed by [32] to perform simplification simultaneously as text summarization. In addition, they present a new metric called CSS, which combines the SARI and Rouge metrics.

Nevertheless, our approach for generating simplified summaries is unique and has not been previously researched.

3 DATASET AND MATERIALS

3.1 Dataset

The dataset we use for this project is the Wiki-Auto dataset [13]. This dataset is based on the WikiLarge dataset [42]. The Wikilarge dataset consists of original Wikipedia items and their corresponding simplified versions, whereas, the Wiki-Auto dataset is created by utilizing a fine-tuned pre-trained BERT model [6] for the task of sentence alignment.

	All Data		**Selected subset**	
No. of documents	138095		500	
No. of aligned documents	122810		500	
No. of sentences	11386147		69019	
	Normal	**Simple**	**Normal**	**Simple**
No. of sentences	10144476	1241671	39093	29926
Avg. length of documents	74	9	79	60

Table 1: Wiki-auto dataset statistics on whole data and the selected subset of 500 documents

In Table 1 we list general information of the Wiki-auto dataset along with the subset of 500 documents used for the project. We see

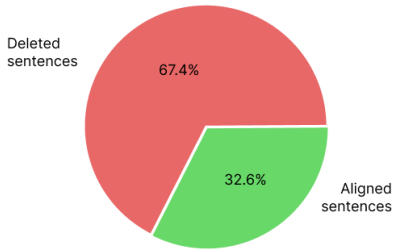

Figure 2: Percentage of deleted (not aligned sentences) in the Wiki-auto dataset

that the dataset has 138095 Wikipedia documents. Each document consists of the normal and the simplified version. In addition, we see that the dataset has 10144476 normal sentences and 1241671 simplified sentences (12.24%). The sentences from the normal document are aligned with sentences in the simplified document.
However, not all normal documents have sentences aligned to sentences in the simplified documents. In Fig. 2 we see that 67.4% of all normal sentences are deleted. In the context of our project, *deleted* sentence means that the sentence is not aligned with a simple sentence.

For our project, a subset of the dataset was used. The subset was selected based on two criteria:

$$|S_n| > 5 \text{ and } |S_s| > (|S_n|/2) + 1$$

where $|S_n|$ and $|S_s|$ are the number of normal and simple sentences in any given document. Both criteria are based on the fact that a summarized text should be no longer than half of the original text [7]. Based on those criteria, we skipped the total number of 110985 documents, leaving us with 14069 documents. Considering the time constraints we had during the project, we have decided to pick up the first 500 documents out of 14069.

3.2 Materials

For the purpose of the reproducibility, we list all materials used during this project. We have hosted our code on Github [1]. The `requirements.txt` file in the main branch of the repository contains the list of Python packages used along with their versions and can be installed using the command:

`pip install -r requirements.txt`.

Besides the code, the dataset we have created based on the two criteria mentioned in Section 3.1 is also available on Github. The dataset is organized in folders where the folder name is the document id, and each folder contains the normal (`source.txt`) and simplified (`destination.txt`) documents.

4 METHODOLOGY & IMPLEMENTATION

As described in Fig. 1, we have a two pipelines setup as follows:

P1 In this flow, the original text is first simplified using the simplification model. The simplified text is then passed to the extractive summarization model to generate the final simplified summary.

P2 Here, first, the summary of the original text is extracted. The summary is then passed to the simplification model to generate the final simplified summary.

The core models used for simplification and summarization in both the pipelines remain the same. The models are described in the following subsections (Sections 4.1 and 4.2). There is no difference in the model architectures between the two pipelines; only the input at each step differs. We have used two-step pipelines instead of a combined model for simplicity of implementation and modularity. Along with the pipelines for generating simplified summaries, we use ROUGE scores and WMD values to evaluate the outputs. For illustration purposes, an original sample text[2] containing six sentences from the dataset is shown below:

> *A crew is a body or a class of people who work at a common activity, generally in a structured or hierarchical organization. A location in which a crew works is called a crewyard or a workyard. The word has nautical resonances: the tasks involved in operating a ship, particularly a sailing ship, providing numerous specialities within a ship's crew, often organized with a chain of command. Traditional nautical usage strongly distinguishes officers from crew, though the two groups combined form the ship's company. Members of a crew are often referred to by the title "Crewman". "Crew" also refers to the sport of rowing, where teams row competitively in racing shells.*

4.1 Text Simplification

For the simplification model, we have used the huggingface implementation of **BERT**$_{base}$ with parameter checkpoints and experiment samples provided by Jiang et al [13]. Specifically, we have used the checkpoints for the model pre-trained on Wikipedia articles.

The model requires pre-processing of text to generate simplified text from complex text. We achieved this by running the preprocessing scripts provided[3], which converts the text into embeddings and builds a vocabulary to be used by the pre-trained model for simplification. The model performs a sentence-level simplification and generates a simple text from the input complex text.

For the first pipeline, where simplification is performed first, the input for this model is the original complex text which is simplified and passed to the summarization model. The second pipeline's input to the simplification step is a summarized text. This text is shorter than the original complex text and would contain a subset of the vocabulary because we are using an extractive summarization model. The simplified text from the sample original text is shown in Fig. 3

4.2 Text Summarization

As mentioned in Section 1, the two major text summarization approaches are extractive and abstractive. Zaman et al. have used a combination of abstractive and extractive summarization models in their HTSS solution, where the abstractive model is employed as the simplification model [41]. Abstractive summarization models result in simplified texts where the structure of the text is changed,

[1] https://github.com/peshmerge/REDI-PROJECT

[2] Article #6801 from wiki-auto aligned dataset [13]

[3] simplification package in https://github.com/chaojiang06/wiki-auto/

A crew is a bodygroup or a class of people who work at a common activity,. generally in a structuredplace orwhere hierarchical organization. A location in which a crew works is called a "crewyard" or a "workyard.". The word has nautical resonances: the tasks involved in operating a ship, particularly a sailing ship, providing numerous specialities within a ship's crew, often organised with a chain of command. Traditional nautical usage strongly distinguishes officers from crew, though the two groups combined form the ship's company. Members of a crew are often referredcalled to by the title "Crewmancrewman". "Crew" also refers to the sport of rowing, where teams row competitively in racing shells.

Figure 3: The result of performing text simplification using the pre-trained BERT Model on the original text. Red indicates deleted words, green indicates added words

A crew is a body or a class of people who work at a common activity, generally in a structured or hierarchical organization. A location in which a crew works is called a crewyard or a workyard. The word has nautical resonances: the tasks involved in operating a ship, particularly a sailing ship, providing numerous specialities within a ship's crew, often organised with a chain of command. Traditional nautical usage strongly distinguishes officers from crew, though the two groups combined form the ship's company. Members of a crew are often referred to by the title "Crewman". "Crew" also refers to the sport of rowing, where teams row competitively in racing shells.

Figure 4: The result of performing text summarization using unsupervised extractive text summarization on the original text. Red indicates deleted words

and new and simple words replace the complex lexical structures. This behaviour makes abstractive models akin to automated text simplification [3]. Due to these reasons, we have used an extractive summarization model. Moreover, we have chosen an unsupervised model, eliminating the need to train the model. This model also removes the vocabulary gap between complex and simple texts.

Our summarization model is a simple ranking-based model. First, we generate the sentence-level embeddings. These are then used to generate a sentence similarity matrix which uses cosine distances to generate similarity scores between every sentence pair from the input text. We then utilize the PageRank algorithm [28] to rank the sentences. More specifically, PageRank does not require any previous training. It is a general-purpose graph-based ranking algorithm which works very well for NLP tasks such as text or sentence ranking.

The model then picks the top **n/2 + 1** sentences from the ranked list and arranges them in the same order as the sentences found in the input text to generate the summary. We choose $n/2 + 1$ as the length of the summary based on the definition that a summary would be at most half the length of the original article [7]. The length of the output summary is an adjustable parameter to the model.

In the first pipeline, the summarization model's input is the simplified text and produces the final simplified summary. In the second pipeline, the input to this model is the original text which is shortened for the simplification model to generate the final simplified summary. The summarized text from the original text in the previous subsection is shown in Fig. 4.

4.3 Evaluation

One of the objectives of this project is to evaluate the quality of a generated simplified summary. In Sections 4.1 and 4.2, we have explained how we generated the simplified summary. We have two simplified summaries for each input text according to the specific pipelines P1 and P2.

The quality of the simplified summaries was evaluated automatically using ROUGE [17], and an embedding-based similarity measure was calculated using WMD [16]. We calculated these metrics for each document in the test dataset (described in Section 3.1) and then took the average scores across the dataset. In the following text, we refer to the generated simplified summary as GSS and the reference simplified summary as RSS.

4.3.1 ***ROUGE.*** The ROUGE metric measures the similarity between GSS and RSS. ROUGE is a set of metrics, more than just one metric. Each metric has a recall, precision, and F1 score. Rouge-1 and Rouge-2 measure the match rate of unigrams and bigrams between the GSS and RSS. The recall measure counts the number of overlapping unigrams and bigrams. These scores are suitable for measuring how well the model is capturing information. In other words, we calculate the percentage of the matching n-grams from the GSS that can be found in the RSS; the higher the number, the more matches we have. The recall tells how much of the generated summary is relevant, whereas, the precision measures how much the simplified summary is relevant. The precision and recall are calculated as follows:

$$R_{rouge-n} = \frac{\text{count(matched n-grams)}}{\text{count(n-grams in RSS)}}; n \in \{1, 2\}$$

$$P_{rouge-n} = \frac{\text{count(matched n-grams)}}{\text{count(n-grams in the GSS)}}; n \in \{1, 2\}$$

The F1 score is based on the precision and the recall. It is a reliable measure because it combines the model's performance of capturing as much information as possible (recall) and with focusing only on the relevant words (precision).

Rouge-L measures the longest common subsequence (LCS) between the GSS and RSS, ignoring the newlines in both texts. The Rouge-LSum is similar to the Rouge-L, however, it requires the sentences in the reference text to be separated by the new line delimiter (\n). rouge-LSum is the LCS score of the union LCS between reference sentence S_{r_i} and generated simplified summary.
For example, if we have the following sentence $S_{r_i} = w_1w_2w_3w_4w_5$ in the RSS, and the following two sentences $S_{g_1} = w_1w_2w_6w_7w_8$ and $S_{g_2} = w_1w_3w_8w_9w_5$ in the GSS. Then the LCS of S_{r_i} and S_{g_1} is $\{w_1, w_2\}$ and the LCS of S_{r_i} and S_{g_2} is $\{w_1, w_3, w_5\}$. The union LCS of S_{r_i}, S_{g_1}, and S_{g_2} is $\{w_1, w_2, w_3, w_5\}$ and union_LCS = 4/5 [17].

For rouge-l the recall and precision are calculated as follows:

$$R_{rouge-l} = \frac{\text{LCS(matched words)}}{\text{count(words in RSS)}}$$

$$P_{rouge-l} = \frac{\text{LCS(matched words)}}{\text{count(words in GSS)}}$$

For rouge-lSum:

$$R_{rouge-LSum} = \frac{\sum_{i=1}^{n} LCS \cup (S_{r_i}, GSS)}{\text{count(words in RSS)}}$$

$$P_{rouge-LSum} = \frac{\sum_{i=1}^{n} LCS \cup (S_{r_i}, GSS)}{\text{count(words in GSS)}}$$

where n is the number of sentences in the RSS.
The $F1$-score for all the ROUGE metrics is calculated as

$$F1 = 2 * \frac{R * P}{R + P}$$

4.3.2 ***WMD***. The Word Movers Distance [16] is a similarity metric which is evaluated by comparing word embeddings of two documents and computing the shortest distance between the two. As we evaluate texts that do not necessarily contain the exact words, using a word embedding-based evaluation model can provide better insights into our pipeline's results. Word Movers distance is built on the concept of Earth Movers Distance where the scores convey the minimum work required to convert one distribution to another [30]. In WMD, this distribution is the GSS and RSS document embeddings. It provides a numerical value which signifies the minimum travel distance between the word embeddings of GSS and RSS.

Given a word embedding space $X \in \mathbb{R}^{d \times n}$ for a vocabulary of n words and the GSS and RSS documents being represented as normalized bag-of-words (nBOW) vectors $d \in \mathcal{R}^n$, where $d_i = \frac{c_i}{\sum_{j=i}^{n} c_j}$ for a word i appearing c_i times in the document d_i (d_i & d_i' are GSS and RSS respectively). The distance between word x_i and x_j is taken as $c(i, j)$. Every word from the GSS is allowed to be transformed into every word in the RSS and the distances are stored in a sparse flow matrix $T \in \mathcal{R}^{n \times n}$. Finally, the minimum cost is calculated as follows:

$$WMD = \arg\min_{T \geq 0} \sum_{i,j=1}^{n} T_{ij} c(i, j)$$

subject to:

$$\sum_{j=1}^{n} T_{ij} = d_i, \ \& \ \sum_{i=1}^{n} T_{ij} = d_i'; \ \ \forall i, j \in \{1, \cdot, n\}$$

As WMD is an algorithm which calculates this distance in a vector space, we can use any pre-trained vector space to get the distances. We used the FastText word representation model for our project, which contains one million word vectors trained on Wikipedia 2017 [5]. This representation model aligns with our dataset as mentioned in Section 3.1. Moreover, the FastTest model uses sub-word information to create these word embeddings and thus, we would not face any out-of-vocabulary words [5, 14, 15].

5 RESULTS

Fig. 5 illustrates the outputs at each step of our pipelines. In P1, the outcome of the simplification model is as shown in Section 4.1. The final simplified summary obtained from the P1 is as shown in Fig. 5a. The output from the summarization model of P2 is as shown in Section 4.2. The final simplified summary from P2 is shown in Fig. 5b. The differences between the pipeline outputs are highlighted in Fig. 5b in green.

	P1			**P2**		
	F1	**P**	**R**	**F1**	**P**	**R**
Rouge-1	0.513	0.583	0.495	0.509	0.594	0.478
Rouge-2	0.23	0.267	0.217	0.228	0.274	0.21
Rouge-L	0.271	0.317	0.256	0.272	0.32	0.251
Rouge-LSum	0.498	0.566	0.48	0.494	0.576	0.464

Table 2: Average ROUGE scores - F1, Precision (P), and Recall (R) for the test dataset

WMD	**Average**	**Least**	**Highest**
P1	0.589	1.028	0.27
P2	0.592	0.993	0.242
Difference	0.027	0	0.247

Table 3: WMD scores for our outputs. Lower is better

By examining the Rouge metrics, as shown in Table 2, we see that our models perform better in Rouge-1 and Rouge-LSum scores compared to Rouge-2 and Rouge-l. In addition, we observe minuscule differences in the Rouge scores between P1 and P2.
Table 3 shows the WMD scores for our outputs. The "*Average*" score columns for P1 and P2 indicate the distance between GSS and RSS in each pipeline. As these values represent the distance - the lower the value, the better the score. The "*Least*" column shows the least scores for P1, P2 and the least difference among all the articles. The zero score signifies that there is at least one article where the outputs from both the pipelines are exactly the same. The "*Highest*" column shows the best scores attained for the respective pipelines and the highest difference in the scores between P1 and P2. However, we observe that the difference in scores of P1 and P2 are minimal, as is the case with the ROUGE scores.

The visual inspection of the pipeline outputs shown in Fig. 5 verifies the insignificant differences in our evaluation scores.

6 CONCLUSION & FUTURE WORK

In this project, we have investigated how the order of simplification and summarization affects generating a simplified summary of a given text. We have utilized a supervised pre-trained BERT model for the task of text simplification and an unsupervised extractive summarization model based on the PageRank algorithm. We observe that neither of these pipelines seems to have significant advantages over the other. Considering our setup and that we have only used a subset of 500 documents, along with the minuscule difference in ROUGE and WMD scores, we conclude that the order of simplification and summarization on a complex text does not matter.

However, by carefully analyzing our results, we realize a scope for improvement in the future. Firstly, we would like to experiment

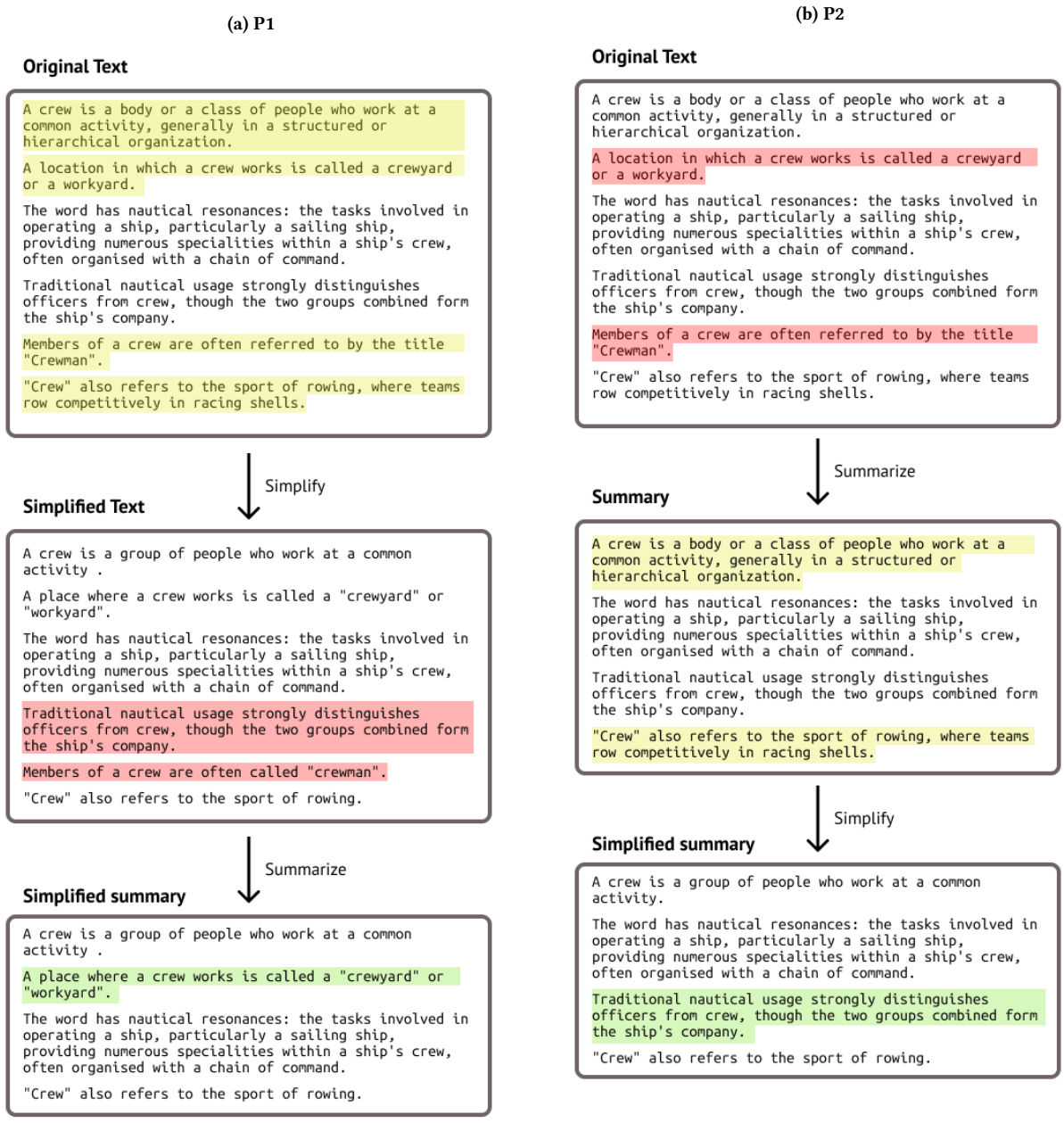

Figure 5: The outputs from both pipelines. Red indicates deleted sentences, yellow indicates modified sentences and green indicates differences in the final outputs between P1 and P2

with different sentence deletion thresholds for the unsupervised summarization model. We have set the threshold to **n/2 +1**, where *n* is the number of sentences in the input text. This value could be based on the target audience. In addition to this threshold, we have also limited our input dataset by restraining the number of sentences in the normal text, as mentioned in Section 3.1. Secondly, we could run our pipelines for only 500 documents due to time and computational constraints. We would also like to extend our work to cover more documents from the subset based on the threshold and, later, the entire dataset.

Moreover, we have used ROUGE metrics and WMD scores to assess the quality of our pipelines' generated simplified summaries. We believe it would be valuable to explore other text evaluation metrics for the task of simplified summarization.

Finally, we would also like to investigate supervised summarization

models for extractive summarization and test various simplification models.

REFERENCES

[1] Mehdi Allahyari, Seyedamin Pouriyeh, Mehdi Assefi, Saeid Safaei, Elizabeth D Trippe, Juan B Gutierrez, and Krys Kochut. 2017. Text summarization techniques: a brief survey. *arXiv preprint arXiv:1707.02268* (2017).

[2] Fernando Alva-Manchego, Carolina Scarton, and Lucia Specia. 2019. Cross-Sentence Transformations in Text Simplification.. In *WNLP@ ACL*. 181–184.

[3] Fernando Alva-Manchego, Carolina Scarton, and Lucia Specia. 2020. Data-driven sentence simplification: Survey and benchmark. *Computational Linguistics* 46, 1 (2020), 135–187.

[4] MEHRNOOSH BAZRFKAN and MUOSA RADMANESH. 2014. Using machine learning methods to summarize persian texts. *Indian J. Sci. Res* 7, 1 (2014), 1325–1333.

[5] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching Word Vectors with Subword Information. *arXiv preprint arXiv:1607.04606* (2016).

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

[7] Wafaa S El-Kassas, Cherif R Salama, Ahmed A Rafea, and Hoda K Mohamed. 2021. Automatic text summarization: A comprehensive survey. *Expert Systems with Applications* 165 (2021), 113679.

[8] Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of artificial intelligence research* 22 (2004), 457–479.

[9] Muhammad Fachrurrozi, Novi Yusliani, and Rizky Utami Yoanita. 2013. Frequent term based text summarization for bahasa indonesia. (2013).

[10] René Arnulfo García-Hernández, Romyna Montiel, Yulia Ledeneva, Eréndira Rendón, Alexander Gelbukh, and Rafael Cruz. 2008. Text summarization by sentence extraction using unsupervised learning. In *Mexican International Conference on Artificial Intelligence*. Springer, 133–143.

[11] Sebastian Gehrmann, Yuntian Deng, and Alexander M Rush. 2018. Bottom-up abstractive summarization. *arXiv preprint arXiv:1808.10792* (2018).

[12] Horacio. 2017. Automatic Text Simplification. *Morgan & Claypool Publishers* (April 2017). https://doi.org/10.2200/S00700ED1V01Y201602HLT032

[13] Chao Jiang, Mounica Maddela, Wuwei Lan, Yang Zhong, and Wei Xu. 2020. Neural CRF model for sentence alignment in text simplification. *arXiv preprint arXiv:2005.02324* (2020).

[14] Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. 2016. FastText.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651* (2016).

[15] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of Tricks for Efficient Text Classification. *arXiv preprint arXiv:1607.01759* (2016).

[16] Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. 2015. From Word Embeddings To Document Distances. In *Proceedings of the 32nd International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Francis Bach and David Blei (Eds.), Vol. 37. PMLR, Lille, France, 957–966. https://proceedings.mlr.press/v37/kusnerb15.html

[17] Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*. 74–81.

[18] Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. *arXiv preprint arXiv:1908.08345* (2019).

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

[20] Igor V Mashechkin, MI Petrovskiy, DS Popov, and Dmitry V Tsarev. 2011. Automatic text summarization using latent semantic analysis. *Programming and Computer Software* 37, 6 (2011), 299–305.

[21] N Moratanch and S Chitrakala. 2017. A survey on extractive text summarization. In *2017 international conference on computer, communication and signal processing (ICCCSP)*. IEEE, 1–6.

[22] Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *Thirty-first AAAI conference on artificial intelligence*.

[23] Shashi Narayan and Claire Gardent. 2014. Hybrid simplification using deep semantics and machine translation. In *The 52nd annual meeting of the association for computational linguistics*. 435–445.

[24] Joel Larocca Neto, Alex A Freitas, and Celso AA Kaestner. 2002. Automatic text summarization using a machine learning approach. In *Brazilian symposium on artificial intelligence*. Springer, 205–215.

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

[26] Makbule Gulcin Ozsoy, Ferda Nur Alpaslan, and Ilyas Cicekli. 2011. Text summarization using latent semantic analysis. *Journal of Information Science* 37, 4 (2011), 405–417.

[27] Gustavo Paetzold and Lucia Specia. 2017. Lexical simplification with neural ranking. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. 34–40.

[28] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The PageRank Citation Ranking: Bringing Order to the Web.* Technical Report 1999-66. Stanford InfoLab. http://ilpubs.stanford.edu:8090/422/ Previous number = SIDL-WP-1999-0120.

[29] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*. 311–318.

[30] Y. Rubner, C. Tomasi, and L. J. Guibas. [n. d.]. A metric for distributions with applications to image databases. In *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*. IEEE, India, 07. https://doi.org/10.1109/ICCV.1998.710701

[31] Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685* (2015).

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

[33] R. Subha Shini and V.D. Ambeth Kumar. 2021. Recurrent Neural Network based Text Summarization Techniques by Word Sequence Generation. In *2021 6th International Conference on Inventive Computation Technologies (ICICT)*. 1224–1229. https://doi.org/10.1109/ICICT50816.2021.9358764

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

[35] Sanja Štajner. 2021. Automatic text simplification for social good: Progress and challenges. *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021* (2021), 2637–2652.

[36] Dima Suleiman and Arafat A. Awajan. 2019. Deep Learning Based Extractive Text Summarization: Approaches, Datasets and Evaluation Measures. In *2019 Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS)*. 204–210. https://doi.org/10.1109/SNAMS.2019.8931813

[37] Renliang Sun, Hanqi Jin, and Xiaojun Wan. 2021. Document-Level Text Simplification: Dataset, Criteria and Baseline. *arXiv preprint arXiv:2110.05071* (2021).

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

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

[40] Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. 2016. Optimizing statistical machine translation for text simplification. *Transactions of the Association for Computational Linguistics* 4 (2016), 401–415.

[41] Farooq Zaman, Matthew Shardlow, Saeed-Ul Hassan, Naif Radi Aljohani, and Raheel Nawaz. 2020. HTSS: A novel hybrid text summarisation and simplification architecture. *Information Processing & Management* 57, 6 (2020), 102351.

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

[43] Yang Zhong, Chao Jiang, Wei Xu, and Junyi Jessy Li. 2020. Discourse Level Factors for Sentence Deletion in Text Simplification. *Proceedings of the AAAI Conference on Artificial Intelligence* 34, 05 (Apr. 2020), 9709–9716. https://doi.org/10.1609/aaai.v34i05.6520

[44] Yang Zhong, Chao Jiang, Wei Xu, and Junyi Jessy Li. 2020. Discourse level factors for sentence deletion in text simplification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 9709–9716.