TSS-EA: Text Summarization and Simplification Model
An Extractive and Abstractive Approach

Yuexiang Liao, Yi Song, Yilin Shan, Dinan Zhao
{yl5767, ys3234, ys3719, dz1371}@nyu.edu
New York University

Abstract

This document is the written report for the final project for the NLP (natural language processing) course. We used a combination of extractive and abstractive approach to summarize and simplify the paragraph. The TextRank algorithm is used as the extractive approach to summarize the top 50% related sentences by running iteration until the probability of importance converges. Then a rule-based approach is used to break a complex sentence into multiple simpler ones. Lexical simplification is performed to replace complex words identified in each sentence with substitutes generated from Bidirectional Encoder Representations from Transformers model (BERT) that has the highest occurrence probability.

***Key Words*: Natural Language Processing, Text Summarization, Text Simplification, TF-IDF, TextRank, BERT, Complex Word Identification, Keras, TensorFlow

1 Introduction

Text Summarization and Simplification is an increasingly popular tasks in NLP because it provides benefits for a large portion of population – people with relatively low literacy or reading disability, or non-native English speakers and introductory English language students, or even normal people who only need to know the key ideas and do not have time to conduct full-length reading. Because text simplification append readability to a complex text while keeping its central ideas, it allows anyone to understand text with significantly less time spent. Many researchers have conducted research using different approaches and have provided important insights. People have recently viewed the simplification holistically as a monolingual text-to-text generation task utilizing statistical machine learning (Zhang et al, 2017), while preceding people focused more on individual aspect of the process, for instances, they performed syntactic simplification (Carroll et al, 1999). A recent breakthrough is obtained by Delvin et al, who proposed a pretrained deep Bidirectional Encoder Representations from Transformers (BERT) from unlabeled text by jointly conditioning both left and right context in all layers, and this model is widely used as language understanding and next word prediction. These methodologies are generally considered as *abstractive approach.* Abstractive simplification approach is generally considered as performing generation of novel sentences by either rephrasing or using new words (Gupta, 2018). Another track of sentence simplification is called *extractive approach*, which is about finding out the most significant sentences and re-ordering then to perform summarization (Gupta, 2018).

We therefore propose our TSS-EA (Text Summarization and Simplification with Combination of Extractive and Abstractive Approach) with main goal of obtaining a shorted version of a paragraph and an both structurally and lexically simplified version of each salient sentence. With satisfying results, we demonstrate TSS-EA as an accurate and easy-to-implement model.

2. Problem Motivation

As college students, we are confronted with tedious amounts of text and information, often with redundant or trivial sentences that do not provide useful information to comprehend while appending painfulness to read. Much of this is online, and while some texts need to be read in full, we may benefit and save precious time by condensing and simplifying what we read to the information that is most relevant and the main idea of the text.

We want to create a system that allows the input of any text, and compress it to become much shorter and more readable, and present a concise and sharp version of the input text. This will allow us to quickly learn what is useful and spend the time saved elsewhere, on some other productive work that would be a more preferable way to spend the time instead of reading tedious blocks of text.

This summarization system can become of use to many groups of people, not only college students. This will not only benefit those who, for whatever reason, want to save time reading large chunks of text, but also will benefit those who have reading or sight disabilities that prevent extended periods of staring at a screen. Even with screen readers or visual assistance, the amount of time spent listening or absorbing longer text would be extremely time-consuming and tiring.

Furthermore, the average attention span of the general public has decreased significantly as technology advances and more people spend their time online, and thus this system will prove useful and friendly to those with shortened attention spans.

3. Methodology

3.1 Extractive: Identifying Important Sentences With A Graph Based Approach

The key concept of the extractive approach – important sentence identification – is that the most important sentences (that should not be discarded and tentatively reduced) are the ones that are the most correlated. We made this assumption that in each paragraph, if we can find the top 50% correlated sentences, then we are able to identify them as the top 50% most important sentences within the context of the text. Based on the assumption, we have devised the following processes in the extractive approach, as shown in *figure 1*.

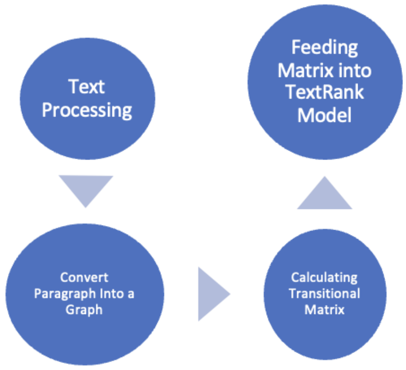

Figure 1

3.1.1 Initial Step: Initializing Transitional Matrix

We first process the paragraph by removing all stop words (for instance, the, this, in, and, etc.) given a manually specified vocabulary of such stop words, words that have no effect on sentence meaning, and all the special characters that do not count as English words which are not needed in the training sequence. Then the input paragraph is split into a list of sentences to be individually analyzed. Then we transform a sentence into a list of weighted undirected graph using adjacency matrix; each vertex denotes a sentence in the paragraph and the weight of the undirected edge between two vertices is their transitional probability from one vertex to another, computed by their similarity score, and is initialized to zero for all the entry in the adjacency matrix.

3.1.2 Calculating the Transition Matrix Using TF-IDF scores and Cosine Similarity

For each of the sentence in the paragraph, we obtained a vector representation of its semantic meaning. Term Frequency – Inverse Document Frequency (TF-IDF), calculated as TF(term, sentence) * IDF(term) is thereby utilized to construct the vector representation. Suppose term *t* appears *n* time in the sentence *s* having in total *x* different terms, and there are in total *y* different sentences and *z* of them contain the term t; then we can get the formula:

$$TF(t,s) = n/x$$, and

$$IDF(t) = log(y/z)$$, where

$$TF - IDF(t,s) = TF(t,s) * IDF(t)$$

Then suppose sentence *s* has term $t_{1....k}$, then the vector representation of *s* can be constructed as:

$$Vector(s) = < TF-IDF(t_1, s), TF-IDF(t_2, s) ... TF-IDF(t_k, s) >$$

Assume that sentences $s_{1...k}$ correspond to vertices $V_{1...k}$, a transitional matrix between two vectors *x* and *y* of dimension *n* is calculated using *cosine similarity* with the following formula:

$$Similarity(x, y) = \frac{\sum_i^n x_i \times y_i}{\sqrt{\sum_i^n x_i^2} \times \sqrt{\sum_i^n y_i^2}}$$

3.1.3 Finding the Most Important Sentences

Mihalcea et al (2004) proposed an *TextRank* algorithm that extend the PageRank algorithm proposed by Page et al (1998) which first used graph representation directed unweighted of web pages and calculate the probability of a user ending up on each page through random clicking based on directed edge connecting each page, using an iteration-based method.

PageRank utilized the iteration-based approach on weighted graph. Formally, let $G = (V, E)$ be a directed graph with the set of vertices *V* and the set of edges *E*, where *E* is the subset of $V \times V$. For a given vertex *V*, let *In(V)* denote the set vertices that has edges pointing to *V* (predecessor of *V*), and let *Out(V)* denote the set of vertices that *V* is pointing to (successor of *V*). *TextRank* extend this idea by incorporating a weighted score W_{ij} indicating the "strength" of the connection between vertex V_i and V_j (Rada Mihalcea et al, 2004).

PageRank and TextRank algorithm assume that the probability score converges regardless of the initial probability score for each vertex. Let $WS(V_i)$ denote the score for probability of node V_i in each iteration. By feeding the similarity matrix into the *TextRank* function, which will generate the probability for each of the sentences in the paragraph according to the formula:

$$WS(V_i) = (1-d) + d * \sum_{V_j \in In(V_i)} \frac{w_{ji}}{\sum_{V_k \in Out(V_j)} w_{jk}} WS(V_j)$$

, where d denotes a damping factor for each iteration. Pseudocode can be found in *figure 2*.

We can choose the top n% sentences with the highest probabilities. By default, we implement the algorithm with n = 50.

```
def textrank(matrix: dict, damping = 0.85, epslone = 0.0001) -> list:
    prob = dict()
    for i in matrix:
        prob[i] = 1 / len(matrix)
    smallenough = False
    while not smallenough:
        count = 0
        newprob = dict()
        for i in matrix.keys():
            sumoutside : float = 0
            for j in matrix[i].keys():
                visited = list()
                suminside : float = 0
                for k in matrix[j].keys():
                    if j != k:
                        if set([j, k]) not in visited:
                            suminside += matrix[j][k]
                            visited.append(set([j, k]))
                if suminside == 0:
                    sumoutside += 0
                else:
                    sumoutside += (matrix[i][j] / suminside) * prob[j]
                newscore = (1 - damping) + damping * sumoutside
            newprob[i] = newscore
        for index, score in prob.items():
            if abs(score - newprob[index]) < epslone:
                count += 1
        if count == len(matrix.keys()):
            smallenough = True
        prob = newprob
    return newprob
```

Figure 2

3.2 Abstractive: Rule-Based Sentence Splitting

Rule-based sentence splitting is applied to conjunction sentences, already identified as the most salient sentences in the input paragraph, to obtain several simplified versions of the original sentences, while keeping its original meaning. POS tagging and sentence parsing are used to determine sentence structures and the role of each token plays in the sentence.

Conjunction sentences usually contain Coordinating Conjunction (CC) (i.e., He ate a sandwich and a slice of pizza.), or belong to compound sentences. Compound sentences are composed of two or more clauses (i.e., He ate a sandwich because he is hungry), which often involves in SBAR. We mainly implement rule-based approach on the two categories.

3.2.1 Splitting Sentences Based on Coordinating Conjunction (CC)

For compound sentences with the Coordinating Conjunction (CC), we first identify the position of the POS "CC", which pinpoints the location of conjunction. This is the point where the phrase to the left and right involve different things and can thus be separated.

Then we create the number of deep copies of the tree the same as the number of non-special-character siblings "CC" has. For example, in the sentence "He eats a slice of pizza and an apple", the word “and” is POS tagged as “CC”, and we deep copied the tree two times because the sub-tree with the POS tag “CC” has two non-special-character siblings “a slice of pizza” and “an apple.” Then for each of the deep copied trees, we delete the sub-tree with the POS tag “CC" and all other siblings. Therefore, the original tree is broken into “He eats a slice of pizza.” and “He eats an apple.”

3.2.2 Parsing a complex sentence with a subclause

A clause is the sentence with the POS tag “SBAR”, such as because, while, when…. We can further categorize clauses into two subcategories: the one with NP, i.e. “**when John is eating a slice of pizza**, he is also playing with his phone”, and the one without NP, i.e. “**when eating a slice of pizza**, John is also playing with his phone”. For the first subcategory, we can simply break it into 2 sentences by finding the SBAR and break the sentence with the SBAR as a reference point: break off the SBAR into an independent sentence, then trim the remaining main clause to also become an independent sentence. For the subcategory without the NP, we can simply find the NP as child of S, SBAR’s sibling, and copy the node under the SBAR, and remove the connecting clause. So the sentence “When eating a slice of pizza, John is also playing with his phone” becomes “John is eating a slice of pizza.” and “John is also playing with his phone.” Here, we copy “John is” to the front of “eating”, and deleted the conjunction of “while”.

3.3 Lexical simplification using BERT

Lexical complexity has been one of the main aspects contributing to text complexity (Dubay, 2004). Therefore, in order to minimize the lexical complexity, we constructed a model to identify and replace lexical complex terms with its simpler substitutes. Lexical simplification in our model can be further split into two separate tasks: Complex Word Identification (CWI) and simplified word substitution. The workflow in shown in *figure 3*.

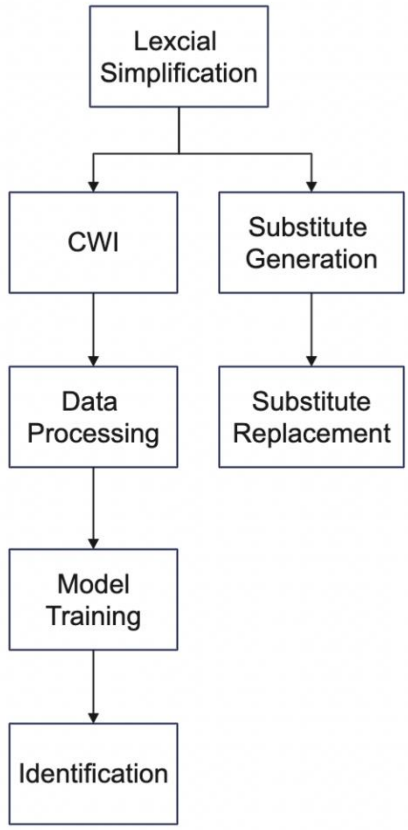

Figure 3

3.3.1 Data

The data we used comes from Complex Words Identification Shared Task 2018 data sets. Sentences from the data set were collected (sequences of words, up to maximum 50 characters) using the Amazon Mechanical Turk (MTurk) crowdsourcing platform, from native and non-native English, German, and Spanish speakers. For the purpose of English literature simplification, we used English only. The English data set consists of three genres:

- Professionally written news
- News written by amateurs (WikiNews)
- Wikipedia articles

Then we split the data set into training set and testing set for model training.

3.3.2 Identifying difficult words

Recurrent Neural Network (RNN) has long been recognized as suitable for dealing with time series analysis. However, RNN suffers from the absence of sufficiency of memory that grasps the preceding terms; therefore, a *Long Short Term*

Memory (LSTM) model is suitable to deal with text processing systems such as ours.

Moreover, *Bi-directional Long Short Term Memory* (BiLSTM) unit model exhibits higher performance because it trains two instead of one LSTM on input sequences, the first input being the original one and the second input being the reversed version of the original input. Therefore, BiLSTM provides us with additional context to the network and results in faster learning. We let the first hidden layer have 150 memory units and same dense layer to be applied at every timestep during LTSM unrolling. Then we fit the model based on padded training and testing complex words in the data set.

3.3.3 Complex words substitution

For each of the sentences, we first use the trained CWI model to identify complex words, and then we mask the complex words. Then we can construct a pair of sentences, the original sentence and the sentence after masking the complex words. The *BERT* will generate its predictions, the probability distribution of its candidate words, by feeding the sentence pair into the model by using BertForMaskesLM function. Then we can choose the most common words by choosing the candidate word with the highest *zipf* value. The higher the *zipf* value, the more common the given word appears in the English language.

4. Evaluation Method

We deployed two common metrics system for evaluating our Text Simplification result: BLUE (Papineni et al., 2002) and SARI (Xu et al., 2016b).

BLEU (bilingual evaluation understudy) is a traditional method to evaluate the quality of machine-translated text from one language to another by calculating the proportion of all matching Ngrams (words) between candidate sentence and referenced sentence. BLUE neglects the lexical and structural aspects of simplified result.

SARI (System output Against References and against the normal sentence): It explicitly measures the goodness of words that are added, deleted and kept by the systems. SARI metric evaluates whether the model has reduced the complexity of the original text by comparing the original sentence, set of reference sentences and the output sentence. We calculated the average overall score of adding, deleting, and keeping score as the final score.

For example #1, comparing the original text and reference sentence with the output result as the candidate:

Original (Input): August is the eighth month of the year in the Gregorian Calendar and one of seven Gregorian months with the length of 31 days.

Reference (Input): August is the eighth month of the year. It has 31 days.

Candidate (Output): August is the eighth month of one of seven annual months with the time of 31 days .August is the eighth month of the year in the indian Calendar .

Testing #	**BLEU Score**	**SARI Score**
1	0.59	0.51
2	0.39	0.35
3	0.45	0.30
4	0.50	0.28
5	0.71	0.30

Table1. Sample Resulting Score

5. Related work

5.1 Text simplification

To alleviate people’s pressure from obscure and verbose sentences, text editing processes such as text simplification and summarization have remained a topic of substantial interest for scholars in the field of natural language. Numerous related developments of the system and corresponding papers have been published ahead of our implementation. Kumar et al (2020) have proposed an iterative based unsupervised learning method that provides metrics in deciding whether to perform “Removal”, “Extraction”, “Reordering”, and “Substitution” and decides the best operation to perform in each iteration until the scoring metric does not improve. Their method provides novel methodologies and insights into text simplification.

5.2 Text rank algorithm

When developing the architecture for important sentence identification, we utilized the TextRank algorithm envisioned by Mihacea et al. (2004) proposed in their paper “TextRank: Bringing Order into Texts”, which is a graph-based algorithm derived from the PageRank algorithm which is proposed by Page et al (1998). Basically, we apply the algorithm to extract a graph from each paragraph and use a text rank model to model the transitional matrices calculated from the extracted graphs to identify the most essential sentences.

5.3 bidirectional Encoder Representations from Transformers model (BERT)

Delvin et al's proposal on the new language representation model, which is called Representations from Transformers (BERT) contributed to one of the core models we used to abstract sentence and develop our TSS-EA system. As suggested in Delvin et al's work, BERT is intended for pre-training deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. In our development. We implemented the BertForMaskesLM function to generate distribution of the most possible candidates of simplified words.

5.4 Evaluation

The related work studied by Omelianchuk et al. (20210), trained a model to perform TST, which is the Text Simplification system based on sequence Tagging. Despite the slight difference in our goal of the systems, we used the same automated evaluation as they used to report the system's result: SARI (System output Against References and against the normal sentence). Same to the process of our measurement with the SARI metric, Omelianchuk et al. (2010) report the scores of operations on adding, deleting and keeping.

Another related work by Knight et al. (2000) designed an algorithm based on approaches such as noisy channel and decision-tree to achieve sentence compression that would preserve the key information of each sentence and restructure each to remain grammatically correct. The evaluation method they used, BLEU((bilingual evaluation understudy) is also applied in our own study, which is a traditional evaluation method on machine translation by calculating the proportion of words matched between candidate sentence and referenced sentence.

6. Future/extending work

While our system is rather simple and is a general-purpose text simplifier that does not cater to any specific type/genre of text (e.g. academic articles, news articles, instruction texts etc.). In the future, it is possible to train reference data based on texts from a specific text type, and alter this system to specialize in the simplification of said text type. This will likely improve the accuracy and simplicity of specific text types, and further help achieve our original goal of making texts easier to read.

In addition, it is also possible to incorporate machine learning (deep learning) into our system for it to become more linguistically realistic and similar to a manual summarization. It is possible to use an existing open-source deep learning algorithm with an appropriate set of data. We can create a set of standards to note the changes from the original sentence to the summarized sentence, such as the POS or BIO tags of each word, as well as word complexity using a trained CWI model, and feed these into a deep learning algorithm (such as the maximum entropy) for it to discern the characteristics of the words discarded and the words kept. It can then generate keep/discard tags for a test set based on the trained evaluation method of given sentences.

7. Conclusion

This paper introduced TSS-EA, an integration of extractive and abstractive approach to text simplification. TSS-EA selects the top 50% important sentenced by using TextRank and PageRank algorithm and splits sentence by POS Tagging. We considered the Coordinating Conjunction as the splitting point and parsed a complex sentence when we tagged a subordinate clause. TSS-EA is able to identify complex word with CWI models and substitute it with simpler word by using BERT model. The result sentence generated by TSS-EA is simple structured and easier to read. As same as our motivation that is generating a shorter text with common vocabularies for people to read. The generated result from TSS-EA gets higher BLUE score but relatively lower SARI score, and the overall score did not reach our expectation. The lack of lexical simplification may be resolved by employing and feeding larger data sets that contain more and a wider variety of words, which may provide a higher possibility for the system to identify matching words.

Acknowledgments

We acknowledge our Professor Dr. Adam Meyers and mentor Mr. Tosin Adesina in guiding us throughout the course of research and in providing us with valuable suggestions.

References

Carroll, John A., et al. "Simplifying text for language-impaired readers." *Ninth Conference of the European Chapter of the Association for Computational Linguistics*. 1999.

Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." *arXiv preprint arXiv:1810.04805* (2018).

Dornescu, Iustin, Richard Evans, and Constantin Orăsan. "A tagging approach to identify complex constituents for text simplification." Proceedings of the International Conference Recent Advances in Natural Language Processing RANLP 2013. 2013

DuBay, William H. "The Principles of Readability." *Online Submission* (2004).

Knight, Kevin, and Daniel Marcu. "Statistics-based summarization-step one: Sentence compression." AAAI/IAAI 2000 (2000): 703-710.

Kumar, Dhruv, et al. "Iterative edit-based unsupervised sentence simplification." *arXiv preprint arXiv:2006.09639* (2020).

Mihalcea, Rada, and Paul Tarau. "Textrank: Bringing order into text." *Proceedings of the 2004 conference on empirical methods in natural language processing*. 2004.

Omelianchuk, Kostiantyn, Vipul Raheja, and Oleksandr Skurzhanskyi. "Text Simplification by Tagging." arXiv preprint arXiv:2103.05070 (2021).

Page, Lawrence, et al. *The PageRank citation ranking: Bringing order to the web*. Stanford InfoLab, 1999.

Siddharthan, Advaith. "An architecture for a text simplification system." Language Engineering Conference, 2002. Proceedings. IEEE, 2002.

Som Gupta, S. K Gupta, Abstractive summarization: An overview of the state of the art, Expert Systems with Applications, Volume 121, 2019, Pages 49-65, ISSN 0957-4174

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

Zhu, Zhemin et al. “A Monolingual Tree-based Translation Model for Sentence Simplification.” COLING (2010).

Papineni, Kishore, et al. "Bleu: a method for automatic evaluation of machine translation." Proceedings of the 40th annual meeting of the Association for Computational Linguistics. 2002.

Xu, Wei, et al. "Optimizing statistical machine translation for text simplification." Transactions of the Association for Computational Linguistics 4 (2016): 401-415.