Extractive vs. Abstractive Text Summarization Methods

Michelle Zhao
CS141

Abstract

Text summarization solves the problem of condensing information into a more compact form, while maintaining the important information in the text. The methods of automatic text summarization fall into two primary categories: extractive and abstractive. A common approach of extractive summarization involves selecting the most representative sentences that best cover the information expressed by the original text based on a ranking of sentences by relevance. A popular method of abstractive text summarization is using an encoder-decoder structure, which generates a latent factor representation of the data, and decodes it to generate a summary. The goal of the project was to analyze and compare the effectiveness of both methods when applied specifically to scientific texts.

1 Introduction

Automatic text summarization is the process of shortening a text documentation using a system for prioritizing information. Technologies that generate summaries take into account variables such as length, style, and syntax. Text summarization from the perspective of humans is taking a chunk of information and extracting what one deems most important. Automatic text summarization is based on the logical quantification of features of the text including, weighting keywords, and sentence ranking.

My motivation for this project came from personal experience. As a student in college, I'm often faced with a large number of scientific papers and research articles that pertain to my interests, yet I don't have the time to read them all. I wanted a way to be able to get summaries of the main ideas for the papers, without significant loss of important content. Text summarization is a widely implemented algorithm, but I wanted to explore different text summarization methods applied to scientific writing in particular.

1.1 Extractive Text Summarization

Extractive text summarization does not use words aside from the ones already in the text, and selects some combination of the existing words most relevant to the meaning of the source. Techniques of extractive summarization include ranking sentences and phrases in order of importance and selecting the most important components of the document to construct the summary. These methods tend to more robust because they use existing phrases, but lack flexibility since they cannot use new words or paraphrase.

1.2 Abstractive Text Summarization

Abstractive text summarization involves generating entirely new phrases and sentences to capture the meaning of the text. Abstractive methods tend to be more complex, because the machine must read over the text and deem certain concepts to be important, and then learn to construct some cohesive phrasing of the relevant concepts. Abstractive summarization is most similar to how humans summarize, as humans often summarize by paraphrasing.

2 Materials and Methods

Although the primary goal of my project was to be able to summarize entire scientific papers, and essentially create abstracts given papers, a paper was too long of an input text to start with. I decided to first work with generating summaries given abstracts, which are much shorter than entire papers. Essentially, my project can be thought of as generating paper titles, given abstracts. First, I needed a dataset of abstract texts with their corresponding titles.

I used the NSF Research Award Abstracts 1990-2003 Data Set from the UCI machine learning repository. The dataset consisted of abstracts that had won the NSF research awards from 1990 to 2003, along with the title of the paper. For my abstractive learning, the training input X was the abstract and the title was the training input Y.

2.1 Extractive Summarization

For extractive summarization, I used the TextRank algorithm, which is based on Google's PageRank algorithm. TextRanks works by transforming the text into a graph. It regards words as vertices and the relation between words in phrases or sentences as edges. Each edge also has different weight. When one vertex links to another one, it is basically casting a vote of importance for that vertex. The importance of the vertex also dictates how heavily weighted its votes are. TextRank uses the structure of the text and the known parts of speech for words to assign a score to words that are keywords for the text.

Algorithm 1 TextRank Algorithm

```
1: procedure TEXTRANK ALGORITHM
2:     Identify filtered text units most representative of the text and add them as vertices to the graph.
3:     Identify relations that connect such text units, and use these relations to draw edges between vertices in the graph.
4:     Iterate the graph-based ranking algorithm until convergence.
5:     Sort vertices based on their final score. Use the values attached to each vertex for ranking/selection decisions.
```

First, we take the input text and split the entire text down to individual words. Using a list of stop words, words are filtered so that only nouns and adjectives are considered. Then a graph of words is created where the words are the nodes/vertices. Each vertex's edges are defined by connections of a word to other words that are close to it in the text. The TextRank algorithm is then run on the graph. Each node is given a weight of 1. Then, we go through the list of nodes and collect the number of edges and connections the word has, which is essentially the influence of the connected vertex. The scores are computed and normalized for every node, and the algorithm takes the top-scoring words that have been identified as important keywords. The algorithm sums up the scores for each of the keywords in all of the sentences, and ranks the sentences in order of score and significance. Finally, the top K sentences are returned to become the TextRank generated summary.

2.2 Abstractive Summarization

First, we need to preprocess the data by constructing an embedding of the text. Embedding the input converts the text into numbers, a more interpretable numerical representation of the data for the encoder-decoder network to work with. I experimented with two different embedding methods: Word2Vec and Global-Vectors (GloVe). Word2Vec is algorithm that combines continuous bag of words and the Skip-gram model to generate word vector representations. GloVe is an unsupervised learning algorithm for obtaining vector representations for words, training from a dictionary of common words.

The encoder-decoder model is composed of multiple recurrent neural networks, one of which works as an encoder, and one as a decoder. The encoder converts an input document into a latent representation (a vector), and the decoder reads the latent input, generating a summary as it decodes. With encoder decoder structures, issues to consider include determining how to set the focus on the import sentences and keywords, how to handle novel or rare words in the document, how to handle incredibly long documents, and how to make summaries readable and flexible with a large vocabulary.

The encoder-decoder recurrent neural network architecture has been shown to be effective when applied to text summarization. The architecture involves two components: an encoder and a decoder. The encoder reads the entire input sequence and encodes it into an internal representation, often a fixed-length vector. The decoder reads the encoded input sequence from the decoder and generates the output sequence, which is the summary. Both the encoder and decoder sub-models are trained jointly, meaning their output feed into the other as input.

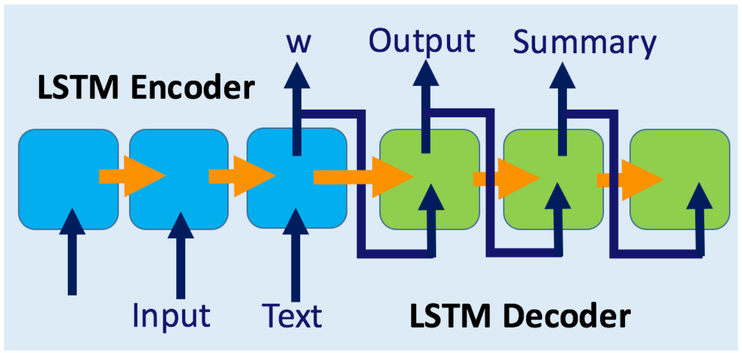

Figure 1. Diagram of LSTM Encoder-Decoder network

The encoder is a bidirectional LSTM recurrent neural network (RNN). RNNs can use their internal state (memory) to process sequences of inputs. LSTMs are capable of learning long term dependencies by storing long-term states and inputs in gated cell memory. The tokenized words of the text are fed one-by-one into the encoder, a single-layer bidirectional LSTM, producing a sequence of hidden states, which is a latent representation of the input. The decoder is a single-layer unidirectional LSTM, which receives the word embedding of the previous word, and the embedding is transformed into a word representation, which is part of the summary.

I used the one-shot encoder-decoder model, where the entire output sequence is generated in a one-shot manner, meaning the decoder uses the latent context vector alone to generate the output summary.

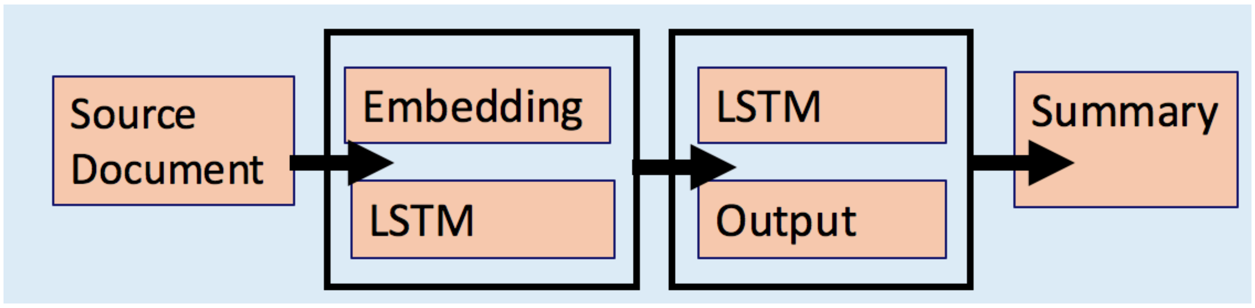

Figure 2. Diagram of one-shot encoder-decoder model.

Abstractive methods like the encoder-decoder network are capable of generating entirely new phrases and sentences to capture the meaning of the text. They tend to be more complex than extractive methods, since they learn to construct some cohesive phrasing of the relevant concepts. However, this also means they are more susceptible to error.

3 Results

3.1 Extractive Summarization Results

The TextRank algorithm generated the following summary. I specified how many sentences to reduce, and generated a 70% reduction summary and a 90% reduction summary which contained the top 3 most important sentences and the top 1 most important sentence, respectively.

Text Input	TextRank Summary
Commercial exploitation over the past two hundred years drove the great Mysticete whales to near extinction. Variation in the sizes of populations prior to exploitation, minimal population size during exploitation and current population sizes permit analyses of the effects of differing levels of exploitation on species with different biogeographical distributions and life-history characteristics. Dr. Stephen Palumbi at the University of Hawaii will study the genetic population structure of three whale species in this context, the Humpback Whale, the Gray Whale and the Bowhead Whale. The effect of demographic history will be determined by comparing the genetic structure of the three species. Additional studies will be carried out on the Humpback Whale. The humpback has a world-wide distribution, but the Atlantic and Pacific populations of the northern hemisphere appear to be discrete populations, as is the population of the southern hemispheric oceans. Each of these oceanic populations may be further subdivided into smaller isolates, each with its own migratory pattern and somewhat distinct gene pool. This study will provide information on the level of genetic isolation among populations and the levels of gene flow and genealogical relationships among populations. This detailed genetic information will facilitate international policy decisions regarding the conservation and management of these magnificent mammals.	**70% Reduction:** Variation in the sizes of populations prior to exploitation, minimal population size during exploitation and current population sizes permit analyses of the effects of differing levels of exploitation on species with different biogeographical distributions and life-history characteristics. Stephen Palumbi at the University of Hawaii will study the genetic population structure of three whale species in this context, the Humpback Whale, the Gray Whale and the Bowhead Whale. This study will provide information on the level of genetic isolation among populations and the levels of gene flow and genealogical relationships among populations. **90% Reduction:** Stephen Palumbi at the University of Hawaii will study the genetic population structure of three whale species in this context, the Humpback Whale, the Gray Whale and the Bowhead Whale.

Table 1. Two summaries were generated from the input text using two different proportions of reduction: 70% and 90%.

3. 2 Abstractive Summarization Results

As part of the pre-processing analysis, ranking the words in order of number of appearances, we saw this distribution of keywords and their frequencies in the training data.

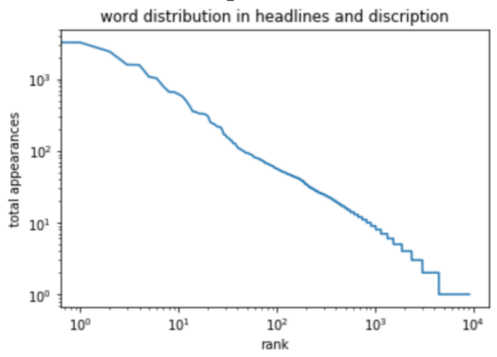

Figure 1. Indexing words by frequency.

We saw that the distribution of set of text input words is much larger and wider than that of words in the summaries.

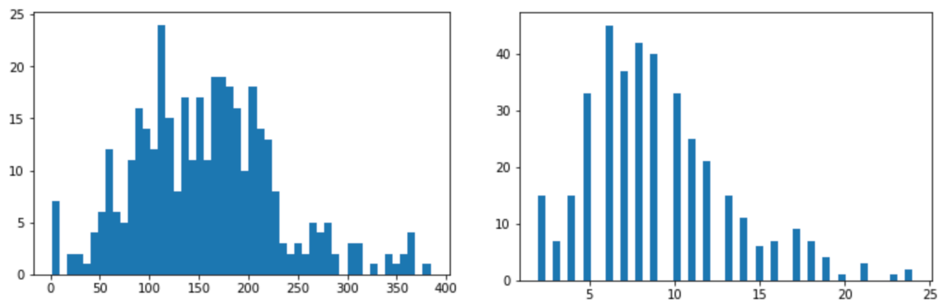

Figure 2. Histogram of distribution of text input words (Left). Histogram of distribution of summary words (Right).

The encoder decoder network generated the following two summaries on the testing input.

Text Input	Summary
Proposal seeks to demonstrate a technique for observing ocean currents by electric field measurements using a towed instrument of recent design measurements will be made in conjunction with a cruise across the in which several additional observational techniques will be employed several data types will be to improve the accuracy of the methods	**Summary #1** Drum frame multidisciplinary **Summary #2** Extension solver bearing.

Table 2. Two summaries were generated with the encoder-decoder network. Both summaries were three words, and used words not within the text.

Experimenting with generated longer summaries to discover whether a longer generated summary would be a better summary for the information, I reran the encoder-decoder network to get a 4-word summary:

Summary #3
Exceptional geology goal visited

This summary is much better than the 3-word summaries, which poses an interesting question: what is the relationship between length of summary and quality of summary generated by encoder-decoder structures?

4 Discussion

TextRank selected the two most significant sentences in the text. E-D generated two different three-word summaries, using words not present in the text, but the summaries generated were not representative of the text and did not make logical sense.

By analyzing the two summaries generated by the encoder-decoder network, I found qualitatively that the extractive summarizer worked better than the abstractive text summarizer. The summaries generated by the extractive method were more understandable and representatively of the text than the abstractive summaries. The extractive summaries were generated much quicker than the abstractive ones. The TextRank algorithm took about 2 seconds to generate a summary, while the encoder-decoder network took about 15 minutes to train.

The encoder decoder network performed rather poorly in comparison to the extractive method. This may have been because the encoder-decoder network didn't have enough training. If the encoder-decoder network perhaps have had more epochs of training, it would have performed better. The training input may have also been too small.

5 Conclusion and Outlook

In conclusion, the TextRank summarization method was very effective in choosing important sentences. As a further extension to the TextRank algorithm, it would be worthwhile to experiment with more ways of choosing "connections" from words to other words. Instead of using the proximity of words to other words in sentences, there may be other ways to measure connections between words, such as using proximity to other words with high connection to the word in question.

The encoder-decoder network was found to be less effective than TextRank, likely because abstractive methods in general are less flexible and more susceptible to error than extractive methods, especially since words not in the text can be used. The next steps in improving the encoder-decoder network are to train on a larger training set, experiment with model hyper-parameters, use beam search, and explore different preprocessing methods.

6 References

1. Jing, Hongyan. “Sentence Reduction for Automatic Text Summarization.” Proceedings of the Sixth Conference on Applied Natural Language Processing -, 2000, doi:10.3115/974147.974190.
2. Garg, Sneh, and Sunil Chhillar. “Review of Text Reduction Algorithms and Text Reduction Using Sentence Vectorization.” International Journal of Computer Applications, vol. 107, no. 12, 2014, pp. 39–42., doi:10.5120/18806-0380.
3. JRC1995. “JRC1995/Abstractive-Summarization.” GitHub, github.com/JRC1995/Abstractive-Summarization/blob/master/Summarization_model.ipynb.
4. “A Gentle Introduction to Text Summarization.” Machine Learning Mastery, 21 Nov. 2017, machinelearningmastery.com/gentle-introduction-text-summarization/.
5. “A Survey of Relevant Text Content Summarization Techniques.” International Journal of Science and Research (IJSR), vol. 5, no. 1, 2016, pp. 129–132., doi:10.21275/v5i1.nov152644.
6. “Text Summarization in Python: Extractive vs. Abstractive Techniques Revisited.” Pragmatic Machine Learning, rare-technologies.com/text-summarization-in-python-extractive-vs-abstractive-techniques-revisited/.
7. “Neural Machine Translation (seq2seq) Tutorial | TensorFlow.” TensorFlow, www.tensorflow.org/tutorials/seq2seq.
8. “Encoder-Decoder Long Short-Term Memory Networks.” Machine Learning Mastery, 20 July 2017, machinelearningmastery.com/encoder-decoder-long-short-term-memory-networks/.
9. Dalalkrish. “Dalalkrish/Text-Summarization-Keras.” GitHub, github.com/dalalkrish/text-summarization-keras/blob/master/Text_Summarization.ipynb.
10. “Text Summarization with TensorFlow.” Google AI Blog, 24 Aug. 2016, ai.googleblog.com/2016/08/text-summarization-with-tensorflow.html.
11. llSourcell. “LlSourcell/How_to_make_a_text_summarizer.” GitHub, github.com/llSourcell/How_to_make_a_text_summarizer/blob/master/train.ipynb.