Automating TL;DR: An Approach to Text Summarization

Haomiao Han

May 16, 2019

1. Introduction

Computers and technology have had a profound influence on how information spread in today's society – increasingly, people prefer to read news and acquire other types of new information online, rather than using traditional ways such as reading newspapers or books. With the enormous amount of information available on the internet, it would be quite time consuming for people to read everything they have found, and people began to avoid reading long articles. Therefore, developing a good automated text summarization algorithm, an important area in Natural Language Processing research, has become quite useful as it saves the amount of time and effort for those reading articles while still presents a reasonable amount of information to people. This paper proposes a text summarization algorithm for online news articles such that, given the text and an optional title of a news article, a summarized version of the article will be generated by the program.

2. Related Work

There are many existing algorithms on automatic text summarization, and the algorithm proposed by this paper is inspired by several of these previous researches.

The algorithm proposed by Hu et al. (2004) uses a two-step process, which includes clustering the paragraphs of the text and finding the "best" sentence in each paragraph. After preprocessing and transforming the paragraphs of the text into vectors, this algorithm first uses K-medoid method to split the paragraphs into different clusters based on the similarity of the paragraph vectors. Then, it selects one sentence in each paragraph cluster that best represents the cluster using cosine similarity. A summary of the original text is thus produced using these sentences. This algorithm provides a baseline for the algorithm proposed by this paper when summarizing articles without titles. This paper will continue to experiment with different varieties of this algorithm, such as changing the way how paragraphs are transformed into vectors and how paragraphs are clustered, in order to see which one produced the best outcome. This paper will also experiment with clustering sentences instead of paragraphs, as clustering sentences may be more effective in summarizing articles with relatively few paragraphs.

Similarly, Sankar and Sobha (2009) proposed an algorithm for automatic text summarization that also uses sentence clustering. However, instead of using machine learning models (such as K-medoid proposed by Hu et al.), Sankar and Sobha proposed four linguistic rules pertaining to connecting words, pronominals, NERs and lexical relationships that together determine how sentences in a text should be clustered. This algorithm also uses another metric – Levenshtein similarity weight – to select the best sentence(s) from each cluster. This algorithm is also worth experimenting as it uses lexical rules instead of pure math (i.e. machine learning models) to determine sentence clusters, which could potentially be more accurate. The similarity metric used by this algorithm is also interesting and may be experimented in this paper.

Ozsoy et al. (2010) proposed another text summarization algorithm that uses Latent Semantic Analysis (LSA) instead of clustering sentences and choosing the “best” sentence out of each cluster. LSA first transforms the text into a vector matrix, and then performs a Singular Value Decomposition (SVD) on the matrix. After that, sentences that are included in the summary are selected based on the result of the SVD matrices. Their proposed algorithm used a modified version of TD-IDF to transform the text into vectors, which could potentially be useful. This paper could also experiment with the LSA method instead of clustering and see if it produces better summarizations.

3. Algorithm and Implementation

The following procedure, based on the clustering algorithm, was used to generate text summarizations:

(1) Preprocess the title and the text (removing stop words, etc.)
(2) Transform each sentence of the text into a vector
(3) Cluster the sentences
(4) For each cluster, select one sentence that best represents the cluster
(5) Generate a summary using the sentences acquired in (4)

The below sections will detail each step of the procedure.

3.1. Preprocessing

After reading the input text, the program will first split the text into different sections, according to the structure of the original text. After some experiments on the sample articles, I decided to split the text into paragraphs instead of sentences. There are several reasons behind the decision:

(1) From a coding perspective, it could be difficult for programs to accurately split the text into sentences. I have attempted two Natural Language Processing packages for python – NLTK and spaCy (another python NLP package similar to NLTK) – and found that both had difficulty in accurately perform sentence segmentation on texts, especially those with more punctuations. Take the following sentence as an example:

    This September's Hugo, which ripped through the Virgin Islands and Puerto Rico before clobbering South Carolina, had sustained winds of 150 m.p.h. and an atmospheric pressure of 918 millibars (27.1 inches).

    Instead of recognizing it as one sentence, NLTK will segment the sentence at the last dot of “m.p.h.” and consider “and an atmospheric…” as another sentence, which is wrong. On the other hand, despite achieving an expected result when processing the above sentence, spaCy often have difficulty in dealing with sentences with quotation marks; it will sometimes split sentences before and after one quotation mark (“) and consider the quotation mark itself as one sentence.

(2) The objective of this program is to summarize news articles; given that a lot of online news articles today already have multiple shorter paragraphs with fewer sentences in each paragraph (instead of having a smaller number of paragraphs with more sentences in each paragraph), it seems that each paragraph has a relatively small amount of thematic focus, essentially making it similar to a longer sentence. Thus, the advantages of splitting texts into paragraphs should outweigh its disadvantages.

(3) Preliminary experiments with several articles show an improvement of summarization quality on most texts if the text is split into paragraphs instead of sentences in the preprocessing step. The results are demonstrated below (Figure 1).

	Average ROGUE-L F-Score*		Improvement for choosing paragraphs over sentences
	Segmenting text into paragraphs	Segmenting text into sentences	
Article 1 (Nature)	**0.329**	0.233	0.096 (41.2%)
Article 2 (Business)	**0.286**	0.198	0.088 (44.4%)
Article 3 (Politics)	**0.375**	0.372	0.003 (0.8%)

	Maximum ROGUE-L F-Score*		Improvement for choosing paragraphs over sentences
	Segmenting text into paragraphs	Segmenting text into sentences	
Article 1 (Nature)	**0.420**	0.297	0.123 (41.4%)
Article 2 (Business)	**0.384**	0.257	0.127 (49.4%)
Article 3 (Politics)	**0.453**	0.442	0.011 (2.5%)

Figure 1: A comparison of ROGUE-L performances of the summaries of three sample texts, using different text segmentation methods
(: For each article, multiple summaries are generated based on the different number of desired paragraphs/sentences in the summary.)*

Thus, the decision was made to segment the text into paragraphs. After that, the program will convert all words into lower case, and remove all punctuations and stop words from the text.

3.2. Vectorization

There are multiple existing methods for converting words into their vector representations, and perhaps the most well-known of which is TF-IDF, a score that is based on the Term Frequency and Inverse Document Frequency of the word. However, I decided to use spaCy's vectorization model, which is a dependency-based word embedding model created by Levy and Goldberg (2014). This model is based on the word2vec model, which used a neural network to model the linguistic context of words and produce the vectors for the words accordingly.

My experiments have shown that spaCy's word vectorization model (word2vec) yielded better results than using TF-IDF for vectorization. Below are the results of three sample texts (Figure 2).

	Average ROGUE-L F-Score		Improvement for choosing word2vec over TF-IDF
	Vectorization with word2vec	Vectorization with TF-IDF	
Article 1 (Nature)	**0.329**	0.263	0.066 (25.1%)
Article 3 (Politics)	**0.375**	0.283	0.092 (32.5%)
Article 4 (Politics)	**0.402**	0.262	0.140 (53.4%)

	Maximum ROGUE-L F-Score		Improvement for choosing word2vec over TF-IDF
	Vectorization with word2vec	Vectorization with TF-IDF	
Article 1 (Nature)	**0.420**	0.365	0.055 (15.1%)
Article 3 (Politics)	**0.453**	0.301	0.152 (50.5%)
Article 4 (Politics)	**0.602**	0.358	0.244 (68.2%)

Figure 2: A comparison of ROGUE-L performances of the summaries of three sample texts, using different text vectorization methods

A matrix that combines vector representations for each paragraph is then produced.

3.3. Clustering

After the matrix with word vectors are produced, the program then clusters each paragraph using K-Means clustering. The intuition behind clustering is that "similar" paragraphs are put into one cluster so that we could simply choose one paragraph in the cluster, thereby summarizing the text. In the current program, clustering is implemented using scikit-learn's K-Means clustering model.

For each text, multiple summaries are generated based on the different number of clusters per text. Currently, the minimum number of clusters is set to (number of paragraphs in the text / 5), and the maximum number of clusters is set to (number of paragraphs in the text / 2), which roughly means that the length summary will be around 20% to 50% of the original text. A new summary is generated for each number of clusters between the aforementioned minimum and maximum.

3.4. Sentence Selection

After the program performs K-means clustering on the paragraphs, one paragraph is selected from each cluster to be included in the summary. The selection process is detailed below:

(1) For each cluster, get the center point of the cluster;
(2) For each paragraph in this cluster, calculate the cosine similarity between the paragraph and the cluster center;
(3) The paragraph that is the most “similar” to the cluster center (i.e. has the highest cosine similarity value with the cluster center) is selected.

3.5. Summary Generation

The paragraphs are first sorted according to their position in the original text, so that the sequence of paragraphs in the summary will be the same as the sequence in the original text. A summary is then generated with these paragraphs.

4. Experiments and Results

4.1. Experiments

Experiments of the system will be conducted using multiple news articles found in the DUC 2001 dataset. The test articles include a variety of topics, such as nature, business, politics, science and finance, in order to fully evaluate the program’s performance. The texts in the DUC 2001 dataset each contains news article as well as a sample summary, which is used as the benchmark for evaluation.

4.2. Results and Evaluation

This paper plans to evaluate the summaries produced by the algorithm using ROGUE (Lin 2004), an evaluation model that compares a summary generated by the algorithm and a sample summary produced by humans. This model is commonly used by other researchers to evaluate the performance of their algorithms. Specifically, the F-Score of ROGUE-L metric (which measures the longest common subsequence between the system output and the reference) will be used.

Below are the results for the six articles that have been used for testing (Figure 3). A cluster visualization is also provided for Article 3 (Figure 4).

	Average ROGUE-L Scores		
	Precision	Recall	F-Measure
Article 1 (Nature)	0.254	0.483	**0.329**
Article 2 (Business)	0.246	0.352	**0.286**
Article 3 (Politics)	0.336	0.438	**0.375**
Article 4 (Politics)	0.364	0.458	**0.402**
Article 5 (Society)	0.166	0.308	**0.212**
Article 6 (Politics)	0.407	0.428	**0.413**

	Maximum ROGUE-L Scores		
	Precision	Recall	F-Measure
Article 1 (Nature)	0.352	0.583	**0.420**
Article 2 (Business)	0.292	0.560	**0.384**
Article 3 (Politics)	0.444	0.635	**0.453**
Article 4 (Politics)	0.596	0.607	**0.602**
Article 5 (Society)	0.255	0.356	**0.271**
Article 6 (Politics)	0.462	0.483	**0.429**

Figure 3: The ROGUE-L performances of the summaries of the six texts in the test dataset

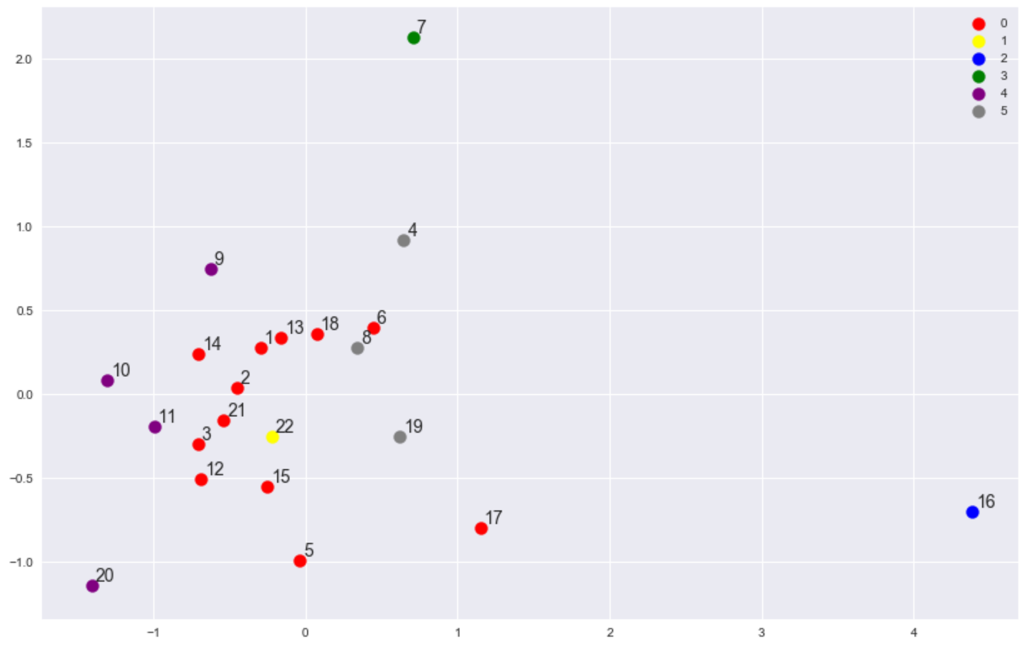

Figure 4: Cluster visualization of Article 3, which addresses the opening of the Channel Tunnel between the United Kingdom and France in the early 1990s. Each dot represents a paragraph in the text. Colors on the dots denote clusters (i.e. dots with the same color belong to the same cluster). ***Note that this graph is generated after performing dimension reduction on the original text vector matrix using Principal Component Analysis, and only preserves approx. 41.4% of the variance in the original data.***

As the paper by Hu et al. used a different scoring method, it would be difficult to compare the above results with the baseline. However, with an average F-Measure of 0.336 (based on the average F-Score over summaries generated for each text) or 0.427 (based on the summary with the highest F-Measure for each text) across all six texts, this program shows a clear improvement over cluster-based algorithms using TF-IDF for text vectorization, and outperforms the system suggested by Ozsoy et al.

4.3. Error Analysis

Analyzing the results from the test articles, I found that, because of the nature of clustering, the performance of this system may be negatively affected by outliers, which refers to paragraphs that cover a different theme than all other paragraphs in the text. When we choose to cluster the text into a relatively small number of clusters, the outliers will be included in one (or more) of those clusters and will affect the position of the cluster centroid, thereby influencing the paragraph selected in step 3.4. When there are a relatively large number of clusters, each outlier will be in its own cluster, and since each of these clusters only contain one paragraph, they will all be included in the output; this could negatively affect the summarization quality if the outliers are not related to the main topics of the text.

As an example, we can take a look at Article 3 again. Below is an excerpt of the article:

> Fagg and Cozette were chosen by draw from among the 3,000 workers to represent their countries in Saturday's meeting. After widening the passage from the size of a peephole to a window, they laid down their tools and shook hands.
>
> "Bonjour," boomed Fagg.
>
> "Hello," Cozette said, chuckling.
>
> Michel Delebarre and Malcolm Rifkind, the French and British transportation ministers, rode in small service trains down the maintenance tunnel to witness the handshake.

The second paragraph above ("Bonjour," boomed Fagg.) will almost always be included in the summary since it was an outlier – it is a short paragraph that contains a large portion of out of vocabulary words. However, as we can see, the mundane exchanges between the two workers here are not really related to the main theme of the article; including it in the summary will negatively affect the performance of the summarization.

Therefore, future improvements can be made to this system – which could be changes to the vectorization process or to the clustering process – in order to solve the issue of outlier paragraphs.

5. Conclusion

In this paper, I proposed an algorithm for automatically summarizing news articles based on K-Means clustering different sections of the text. My experiments have demonstrated that splitting the text into paragraphs instead of sentences produces more satisfactory results, and that using spaCy's word2vec instead of using TF-IDF for vectorization improves the quality of the summarizations.

Summaries are then evaluated using ROGUE-L metric. With an average F-Measure of 0.336 or 0.427, the performance of this system has been satisfactory and outperformed some baseline systems. In the future, there are some possible improvements that can be made to this system, such as incorporating the title of the article in the summary-generation process, or automatically deciding the number of clusters for each article.

References

Hu, P., He, T. and Ji, D. 2004. Chinese Text Summarization Based on Thematic Area Detection. *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 112-119.

Levy, O. and Goldberg, Y. 2014. Dependency-Based Word Embeddings. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Short Papers)*, pages 302–308.

Lin, C-Y. 2004. Chinese Text Summarization Based on Thematic Area Detection. *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74-81.

Sankar, K. and Sobha, L. 2009. An Approach to Text Summarization. *Proceedings of CLIAWS3, Third International Cross Lingual Information Access Workshop*, pages 53-60.

Ozsoy, M. G., Cicekli, I. and Alpaslan, F. N. 2010. Text summarization of Turkish texts using latent semantic analysis. *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 869-876.