I AM SAM: An Automatic Text Summarization System using different Extractive Techniques

John Ray Martinez
Drexel University
Philadelphia, PA
jbm332@drexel.edu

Jonathan Musni
Drexel University
Philadelphia, USA
jem472@drexel.edu

Juan Miguel Trinidad
Drexel University
Philadelphia, USA
jbt46@drexel.edu

ABSTRACT

Three NLP (Natural Language Processing) automated summarization techniques were tested on business news articles from BBC news website. The automated summaries were generated after feeding the news articles text files into each algorithm. Using the standard ROUGE Recall scoring technique, the LSA extractive summarizer technique had the best summarization score. The LSA extractive technique yielded an average Rouge-1, Rouge-2, Rouge-l Recall score of 0.867, 0.617, and 0.841 respectively which outperformed the LexRank and Luhn algorithms. This article covers the implementation of an automatic test summarization system called I AM SAM through Heroku, the system architecture, and features of the system including its capability to summarize news article from a URL.

CCS CONCEPTS

• **Applied computing** → **Information Retrieval**.

KEYWORDS

Automatic Text Summarization, Extract, SUMY.

ACM Reference Format:
John Ray Martinez, Jonathan Musni, and Juan Miguel Trinidad. 2020. I AM SAM: An Automatic Text Summarization System using different Extractive Techniques. In *ACM BCB '20: ACM Conference on Bioinformatics, Computational Biology, and Health Informatics, Aug 30–Sep 02, 2020, Virtual Conference.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

In recent years, there has been a growth of large volume of text data from a variety of sources. This explosion of amount of text data led to the problem of information overload. Online news surfing, for example, provides readers with many articles since it involves multiple news sources.

ACM BCB '20, Aug 30–Sep 02, 2020, Virtual Conference

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

The generation today called 'Net generation' learns through multitasking, performing activities simultaneously, and has short attention span. 'Net generation' can perform more tasks simultaneously and shift their attentions quickly from one to another, but would probably be overwhelmed or get frustrated if they are asked to read a long report for a while . Thus, more educators motivate them to engage in the learning content by supplying shorter contents in the curricula [17].

To alleviate information overload and considering the characteristic of the 'Net generation', the need for automatic text summarization is even deemed necessary. Automatic text summarization is formally defined as the process of distilling the most important information from a source (or sources) to produce an abridged version for a particular user (or users) and task (or tasks) [15]. Automatically producing summaries from text documents is a challenging but critical task for information retrieval. It aims to condense the original text into its essential content and to assist in filtering and selection of necessary information [3].

Extractive and abstractive are the two types of summaries that can be inferred from a text. Extractive summaries [6, 16] are created by reusing words and sentences of the input text while abstractive summaries [5, 10, 13, 14] are created by re-generating the extracted content. Studies on abstractive summarization often rely on an extractive preprocessing component to produce the abstract of the text [9, 18]. In general, abstractive summarization methods are relatively harder than extractive ones since it encounters problems such as semantic representation, inference and natural language generation.

Studies on text summarization often focus on extractive since it gives better results [7] due to its data-driven approaches such as sentence extraction. One tool for text summarization is Python package sumy [1]. It has three most notably used models namely LSA (latent semantic analysis), LexRank and Luhn. LSA is an unsupervised method of summarization that combines term frequency techniques with singular value decomposition to summarize texts. Also an unsupervised approach, LexRank is a graphical based text summarizer inspired by algorithms PageRank. Meanwhile, Luhn is a naive approach based on TF-IDF. It scores sentences based on frequency of the most important words and also assigns higher weights to sentences occurring near the beginning of a document [2].

In this study, we investigate and evaluate the application of sumy models on the extractive summarization task using news articles and show that the results obtained with

LSA are competitive with other two algorithms developed. Furthermore, utilizing the sumy extractive summarization techniques, we build and implement a web application on Heroku that mainly functions as text summarizer. The objectives of this study are to: i) to study whether LSA extractive summarizer outperform other sumy methods such as Luhn and LexRank using news articles summarization tasks; and ii) using sumy methods, implement a system that summarizes news articles from plain text or URL and gives the ROUGES scores instantaneously.

This paper is organized as follows. In the next section, the experiment conducted is described. Then, the implemented system adopted to summarize texts is discussed. The paper concludes with derived insights and avenue for future research.

2 EXPERIMENTS

In this section we describe the summarization dataset, present our implementation and evaluation methods, and analyze our results.

2.1 BBC Summarization Dataset

This dataset is approximately 2225 documents from the BBC news website and represented into five topical areas such as business, entertainment, politics, sport, and technology [8]. This dataset for extractive text summarization has 510 business news articles of BBC from 2004 to 2005. For each article, one summary are provided in the Summaries folder.

In this study, the first 100 pairs of business news articles and its correponding reference summaries were manually selected and used. The extractive summary articles will be used as reference summaries (gold standard) for evaluating the system summaries using ROUGE. .

2.2 Methodology

In order to answer the research question of this study, we applied the three sumy methods in the sampled business news articles. All these algorithms extract six sentences from each article in order to compose the summary. We performed an experimental comparison with three extractive summarization techniques.

The performance of each summarization technique was evaluated by using variants of the ROUGE measure [11]. This performance metrics is a method based on Ngram statistics and found to be highly correlated with human evaluations [12]. Concretely, Rouge-N with unigrams and bigrams (Rouge-1 and Rouge-2) and Rouge-L. Each Rouge has corresponding F1, precision, recall scores. First, the value of the evaluation measure was calculated for each of the article. Next, we took average of those scores to arrive at a consolidated Recall and F1 scores for each Rouge. Algorithm 1 shows the pesudo-code of the implementation of the method in this study.

Algorithm 1: Average Rouge F1 and Recall scores calculation from BBC business news articles data

input :
- sample_size := # of articles to be considered
- news data set := BBC data composed of n-sampled business news articles
- reference data set := BBC data composed of corresponding n-sampled business news reference summaries
- language := user-defined language to be used by tokenizer, stemmer and stop-words removal methods
- sent_count:= user-defined number of sentence to be extracted by the model from original news article

output : complete list of Rouge metrics mean scores of different models

```
begin
    agg_r1f_table = []; agg_r1r_table = [];
    agg_r2f_table = []; agg_r2r_table = [];
    agg_rlf_table = []; agg_rlr_table = [];
    for iter = 1 to sample_size do
        parser = parser(news article, tokenizer);
        model = summarizer(stemmer);
        summary = model(parsed doc, sent_count);
        scores = rouge.get_scores(summary, reference
         summary);
    end
    use agg_table to create the list of Rouge scores,
     and compute the mean of their F1 and Recall
     scores;
end
```

2.3 Experimental Results and Discussion

We evaluate the four summarization techniques on a single-document summarization task using 100 news articles from business section of BBC dataset. For this task to have a meaningful evaluation, we report ROUGE Recall as standard evaluation and take output length into account [4]. For each article, each summarizer generate a six-sentences summary. The corresponding 100 human-created reference summaries are provided by BBC and used in the evaluation process.

We compare the performance of the three different summarizing techniques with each other. Table 1 shows the results obtained on this data set of 100 news articles, including the results for LSA (shown in bold), and the results of the other two sumy summarizers in the single document summarization task.

LSA summarization technique succeeds in summarization task on news articles followed by sumy-LexRank then sumy-Luhn. Figure 1 Figure 1 visualizes the comparison of models using Rouge Recall as performance metrics. As shown, LSA has the best performance in extractive summarization task on business news articles.

Table 1: The average Recall (F1) of test set results on the BBC business news articles dataset using granularity of text metrics ROUGE-1, ROUGE-2 and ROUGE-L.

Model	ROUGE-1	ROUGE-2	ROUGE-L
LSA	**0.867 (0.059)**	**0.617 (0.041)**	**0.841 (0.082)**
Luhn	0.794 (0.052)	0.407 (0.031)	0.612 (0.045)
LexRank	0.844 (0.072)	0.576 (0.049)	0.807 (0.096)

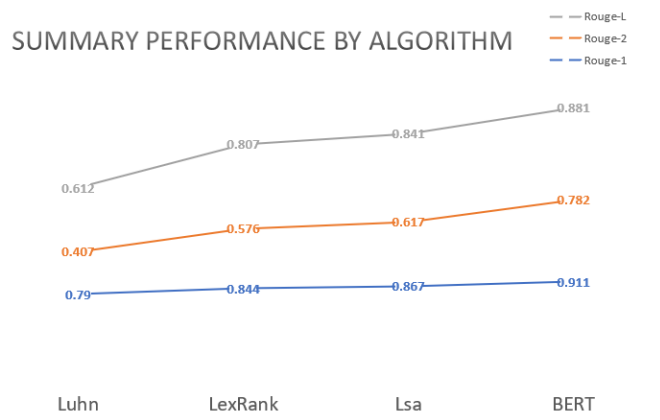

Figure 1: ROUGE performance of algorithms.

3 IMPLEMENTED SYSTEM

In this section, we present the software/hardware requirement and the overall architecture to implement the proposed system, and discussed the major system features.

3.1 Software/Hardware Requirements

Currently the web application is deployed and hosted on Heroku, a a cloud platform as a service (PaaS) solution. This solution allows a quick deployment, building and managing the application while bypassing infrastructure requirement to host a web application. Heroku supports applications written in several programming languages such as Python. We utilized the version control system Git along with Github as a development platform. The Github repository with the uploaded codes is linked with Heroku. The server machine should have internet connectivity if it is hosted on an independent server so that the I AM SAM web app can be invoked. Currently the application is hosted on Heroku's cloud server which has internet connectivity and can be accessible through internet. In addition, the application works in all browsers and has mobile support.

3.2 System Architecture

In this section, we present the overall architecture of our proposed web application for single document summarization based on news components using sumy models LSA, Luhn and LexRank. As highlighted in Figure 2, there are three main phases which include the back-end, front-end, and deployment. To create a web application, we utilized Flask which is a micro web framework written in Python. For layout to look good, we styled it with Boostrap. Finally, we deployed the models on Heroku. Figure 3 visually presented the detailed diagram of system architecture. For the detailed diagram of system architecture, please refer to the Appendix.

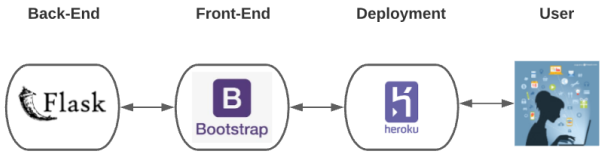

Figure 2: System architecture overview.

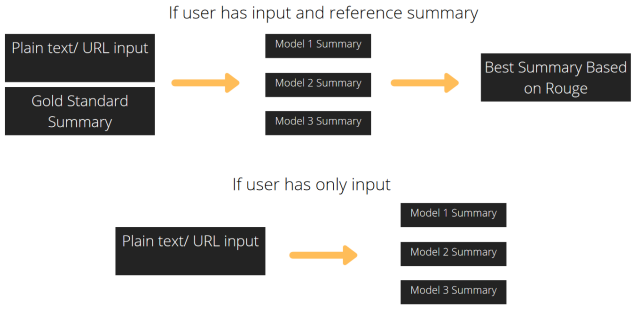

Figure 3: System Features.

3.3 System Features

I AM SAM web app alleviates information overload by distilling important information using machine learning algorithms. It is an assistant that helps the users manage time by providing text summary in seconds. In this project, we focus on plain text and URL as inputs. More specifically, we consider the following features:

3.3.1 Target Length. Target length is the number of sentences in the text summary. This feature lets the user input the prefer length of summary in terms of number of sentences in the output.

3.3.2 Inputs. There are two possible inputs: plain text articles or URL that contains the text articles. Figure 3 visualizes the diagam flow for these two options.

3.3.3 Check Mode. Feature 'check mode' gives the user a choice to put up a reference summary. There is a ON and OFF toggle for this feature. It provides the user to check how good the provided summaries with respect to the reference summaries. This also shows the calculated Rouges (F1, precision, recall) scores for each model text summary.

3.3.4 Best Summary Generator. Once the check mode is ON, the system compares the summary output of each algorithm and output the best model based on Rouge Recall metric.

3.3.5 Instant Text Summary. The system provides the user a summary text of any plain text or text online in just a few seconds. It helps user to manage time and money.

4 CONCLUSION AND FUTURE WORKS

We presented sumy python package implementation of LSA (latent semantic analysis) outperforming the other models such as LexRank and Luhn in extractive summarization task using BBC business news articles. Furthermore, we implemented an automatic text summarization system called I AM SAM through Heroku that has capability to summarize news article from a URL or plain text utilizing the three sumy extractive summarization techniques.

As future work, we plan to extend the averaging algorithm to all articles in business news articles folder which has a total of 510 articles. In addition, we will explore other news articles such as entertainment, politics, sport, and technology.

5 APPENDIX

For the presentation of the detailed architecture of the application, please refer to Figure 4.

6 ACKNOWLEDGMENTS

This work was supported in part by the Drexel University College of Computer and Informatics under the course INFO-624-003. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Drexel University College of Computer and Informatics.

REFERENCES

[1] Mišo Belica. 2020. Module for automatic summarization of text documents and HTML pages. https://github.com/miso-belica/sumy

[2] Mišo Belica. 2020. Summarization methods. https://github.com/miso-belica/sumy/blob/master/docs/summarizators.md

[3] F. L. Wang C. C. Yang. 2008. Hierarchical summarization of large documents. *Journal of the American Society for Infor-mation Science and Technology* 59, 6 (2008), 887 – 902.

[4] Benjamin Van Durme Courtney Napoles and ChrisCallison-Burch. 2011. Evaluating sentence com-pression: Pitfalls and suggested remedies.. In *In Proceedings of the Workshop on Monolingual Text-To-Text Generation*. Association for Computational Linguistics: Portland, Oregon, 91–97.

[5] H.P. Edmundson. 1969. New methods in automatic abstracting. *Journal of the Association for Computing Machinery* 16, 2 (1969), 264 – 285.

[6] H.P. Edmundson. 1969. New Methods in Automatic Extraction. *J. ACM* 16, 2 (1969), 264 – 285.

[7] Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *J. Artif. Intell. Res.(JAIR)* 22, 1 (2004), 457 – 479.

[8] Derek Greene and Pádraig Cunningham. 2006. Practical Solutions to the Problem of Diagonal Dominance in Kernel Document Clustering. In *Proc. 23rd International Conference on Machine learning (ICML'06)*. ACM Press, 377–384.

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

[10] M.P. Marcus L.A. Ramshaw. 1995. Text chunking using transformation-based learning. In *Proceedings of the Third ACL Workshop on Very Large Corpora, Cambridge MA, USA*.

[11] C.Y. Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *In Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*. Association for Computational Linguistics: Barcelona, Spain, 74–81.

[12] C.Y Lin and E.H. Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *In Proceedings of Human Language Technology Conference (HLT-NAACL 2003)*. Association for Computational Linguistics: Edmonton, Canada.

[13] H.P. Luhn. 1958. The Automatic Creation of Literature Abstracts. *IRE National Convention,* (1958), 60 – 68.

[14] H.P. Luhn. 1959. The Automatic Creation of Literature Abstracts. *IBM Journal of Research and Development* (1959), 159 – 165.

[15] Maybury M.T Mani I. 1999. In *Advances in Automatic Text Summarizat*. The MIT Press.

[16] R. Mihalcea. 2004. Graph-based ranking algorithms for sentence extraction, applied to text summarization. In *In: Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*.

[17] D. G. Oblinger and J. L. Oblinger. 2005. In *Educating the net generation*. Educause. Retrieved August, 19, 2020 from https://www.educause.edu/ir/library/PDF/pub7101.PDF.

[18] Dan Klein Taylor Berg-Kirkpatrick, Dan Gillick. 2011. Jointly learning to extract and compress. In *In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, 481–490.

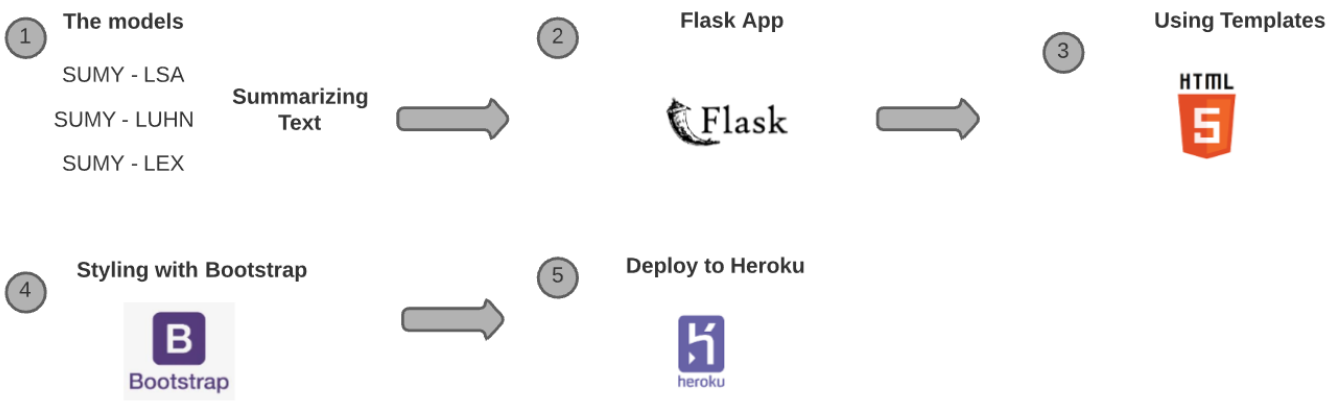

Figure 4: System architecture detail.