2021

Topic Identification & Text Summarization for Computer Science Journals

Group 4

Nestor Molina

Ngoc Phan

William Baker

GitHub Repos

https://github.com/nphan20181/nlp_project

Introduction

When working on a research paper or article, most researchers often have difficulty in writing an effective abstract and in identifying appropriate keywords that reflect the main ideas. According to the Writing Center at the University of Wisconsin-Madison, an abstract is a short summary of a research paper that includes one paragraph of 6-7 sentences or 150-250 words [1]. The Writing Center at the University of Wisconsin-Madison also indicates that "search engines and bibliographic databases use abstracts to identify key terms for indexing research articles" [1]. Therefore, an effective abstract would enable identification of relevant key terms for the article and potentially increase the article's access rate as it increases the chance of the article showing up on the search page when the users search for related topics. This research applies text summarization techniques to generate a short summary of Computer Science research articles, which could further serve as an abstract of the article. Relevant topics would also be extracted from the article's texts to enable researchers to quickly identify relevant keywords for their paper. This could further enable researchers to come up with an effective title for their paper and attract more readers to access their article.

The objectives of this project can be split into several distinct parts.

- Data collection
- Data pre-processing
- Topic extraction
- Article summarization
- Model evaluation
- Web app development using Plotly and Dash

Using the article's full texts as the training data, we will develop two different models. The first model would be used to identify the topic of the relevant text and the second model will be used to summarize that same text. Additionally, a pre-trained text summarization model would serve as a baseline model for comparison across models. Finally, after the models are trained and evaluated, the best text summarization model would be deployed in a web application to summarize the texts provided by the user. Furthermore, the same topic identification algorithms would also be used to extract topics from the input text provided by the user.

Background

Neural Extractive Text Summarization with Syntactive Compression [7]

Written by Jiacheng XU and Greg Durrett, this article explores a neural architecture for single-document summarization using the CNN, Daily Mail, and New York Times datasets. Their proposed system begins by encoding a given text. It then selects groups of sentences to further compress while attempting to keep intact the grammar and meaning of those sentences. The neural net decides which compressions to apply to the groups of sentences based on the context of the text, syntactic constituency parser, and the decoder models recurrent state. They found that their system matches or exceeds the state of the art on the given datasets. In this context, compression refers to methods that reduce the size of a text by removing unnecessary or redundant information. This specific model uses, and expands upon, rules derived by other computer scientists in previous papers. The model is composed of a bidirectional LSTM that is used for encoding, a sequential LSTM that is used for decoding, and a text compression module that helps to select certain words and sentences.

This model performs very well and is a great candidate for our neural summarization model. However, it is quite complex and difficult to implement.

LDA for Text Summarization and Topic Detection [4]

This article, written by Rosaria Silipp in early 2019, talks about using Latent Dirichlet Allocation for topic detection. Latent Dirichlet Allocation, or LDA, is an algorithm used for unsupervised dataset. The general and oversimplified idea of LDA is a probabilistic approach to extracting the K number of topics from a set of documents. Each word from a document is assigned to one of the topics. Then through some probabilities, each word is reassigned to a different topic. This process is repeated until each word can no longer be reassigned based on some probability. This technique is used for topic detection. Documents are clustered according to their apparent topic which is extracted using an LDA. Most of the limitations come from the probability based algorithm and choosing the number of topics to extract. Overcoming the limitation of the algorithm is difficult considering it would take high-level math knowledge. Choosing a ‘good’ number of topics to extract seems more realistic as a task to improve the algorithm. Testing several numbers of topics to see what kind of results each number gives is a simple yet effective way to see how much better the model runs. Too many and the model may take too long to create. Too few and the topics extracted may not be enough to encapsulate the dataset.

Text Summarization Techniques and Applications [5]

This article, written by Dehru, Virender, et al in 2021, discusses two text summarization methods: extractive summarization and abstractive summarization [5]. The extractive method splits the text into sentences and assigns a weight to each sentence. The higher weight a sentence has, the more important it is. After the sentences have been assigned a weight, the top *k* sentences with the highest weight would be selected to be included in the summarized text. The order of a sentence in the summarized text is based on its arrangement in the original text. There are three common methods for determining the weight of a sentence: 1) word weighted frequency, 2) word probability, and 3) TextRank. In contrast to the extractive method, the abstractive method learns the grammatical structure and semantic meaning of the text and then generates new sentences as a summarized version of the given text. Compared to the extractive method, the abstractive method is more complex as it trains deep learning models to predict the sequences of words.

The article then further discusses the results after applying two aforementioned methods on News Summary dataset and Food Reviews Amazon dataset. For the extractive method, the model that uses TextRank for sentence's weight assignment performs slightly faster and better than the models that use word weighted frequency or word probability for sentence's weight assignment. For the abstractive method, a Long Short Term Memory (LSTM) model has been trained to produce the summarized text. However, the result of the LSTM model has not been discussed.

Work Process Flow

Our project contains four main processes: 1) data preparation, 2) model training, 3) model evaluation, and 4) web application development. During the data preparation process, we perform web scraping to collect pdf articles of Computer Science journals. Then, we extract the full texts from pdf files and save the data into a csv file. Next, we perform exploratory data analysis on the unprocessed data. Finally, we preprocess the raw data and export the cleaned data for model training. During the model training phase, we build a text summarization model for text summarization and a topic identification model for topic extraction. After the models have been trained, we evaluate the model results and go back to the model training process to retrain the models until the models meet the desired performance level. Lastly, we build a web-based application that enables the user to paste the texts into a multiline textbox and then provides the summarized text and the identified keywords for the input texts. Figure 1 displays the work process flow diagram of our project.

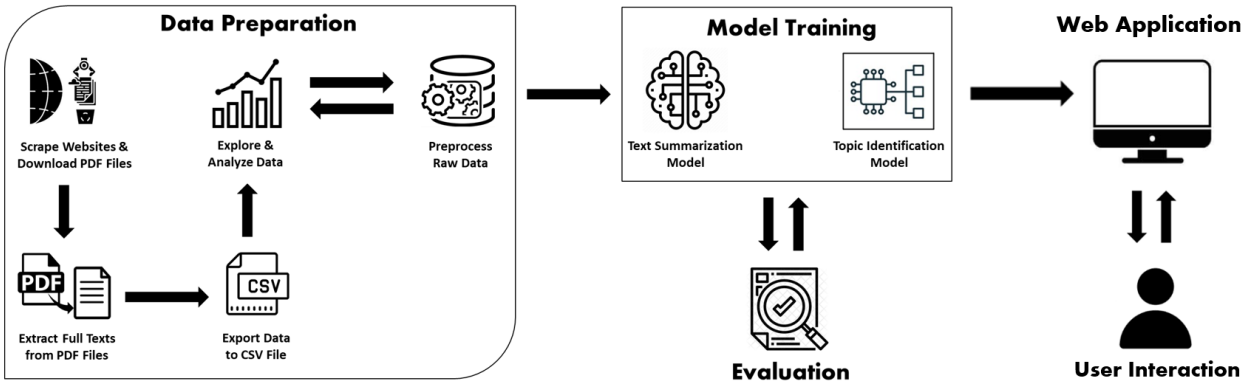

Figure 1: Work Process Flow Diagram

Dataset

Data Collection

A collection of published articles from 2005 to 2021 in the Journal of Computer Science has been retrieved from the Science Publications website [2]. Web scraping has been performed on the following web pages to retrieve article's information and download the article's PDF file.

- Archive Page of Journal of Computer Science [2]
  This page contains the URLs for all journal issues from 2005 to current. On this page, web scraping has been performed to retrieve the URLs of journal issues along with year of publication, volume number, and issue number.

Figure 2: Screenshot of archive page of Journal of Computer Science

- Issue Page of Journal of Computer Science

  This page shows all articles belonging to the selected journal issue. The web page can be accessed by clicking on the hyperlink for a specific journal issue on the Archive Page of Journal of Computer Science. Web scraping has been performed on this page to retrieve the article's title, URL of the article's PDF file, and URL of the article's abstract page.

Figure 3: Screenshot of Issue Page of Journal of Computer Science

- Article's Abstract Page

This page contains general information for one specific article and can be accessed from Issue Page of Journal of Computer Science. Web scraping has been performed on this page to retrieve the abstract and the keywords of the selected article.

Figure 4: Screenshot of article's Abstract Page

After retrieving the URL of all published articles and relevant information, a Python code has been written to download the articles and extract the article's texts from PDF files. Once the texts have been extracted, the data is saved to a csv file. The result dataset contains 2,691 examples and 7 columns. Figure 5 displays the first three rows of the data.

Date	Title	Abstract	Keywords	File Name	URL	Text
Published: 8 January 2021	A Systematic Literature Review on English and ...	Due to the enormous growth of information and ...	English Bangla Comparison, Latent Dirichlet Al...	2021_17_1_jcssp.2021.1.18.pdf	https://thescipub.com/pdf/jcssp.2021.1.18.pdf	Because of the rapid development of Informatio...
Published: 21 January 2021	DAD: A Detailed Arabic Dataset for Online Text...	This paper presents a novel Arabic dataset tha...	Arabic Dataset, Arabic Benchmark, Arabic Recog...	2021_17_1_jcssp.2021.19.32.pdf	https://thescipub.com/pdf/jcssp.2021.19.32.pdf	In the literature, many papers that focus on A...
Published: 20 January 2021	Collision Avoidance Modelling in Airline Traff...	An Air Traffic Controller (ATC) system aims to...	Air Traffic Control, Collision Avoidance, Conf...	2021_17_1_jcssp.2021.33.43.pdf	https://thescipub.com/pdf/jcssp.2021.33.43.pdf	Collision avoidance on air traffic becomes ver...

Figure 5: First three rows of dataset

Data Features

As shown in figure 5, the raw dataset contains seven columns: 1) the date of the publication, 2) the title of the article, 3) the article's abstract, 4) the article's keywords, 5) the name of the article's pdf file stored in local machine 6) the hyperlink to the article's pdf file, and 7) the article's full text. Additional features have also been extracted from the data for further data analysis. These features include the year of publication, the length of the article's full text, the title's length, the abstract's length, and the number of keywords. Figure 6 displays the summary statistics of the data features of unprocessed dataset. As shown in figure 6, the minimum length of the full text is three which indicates that some examples in the dataset may contain invalid text. Therefore, further analysis of the full text is needed when performing data preprocessing.

	mean	median	std	min	max
Title Length	83	81	26	14	260
Abstract Length	1437	1270	717	59	7295
Number of Keywords	5	5	2	1	16
Text Length	17686	16546	11257	3	107028

Figure 6: Summary Statistics of Unprocessed Data

Exploratory Data Analysis

Figure 7 shows the number of published articles per year. Year 2014 has the greatest number of published articles while year 2016 has the least number of published articles. There seems to be an increasing trend in the number of published articles from 2005 to 2014. After 2014, the trend seems to be decreasing.

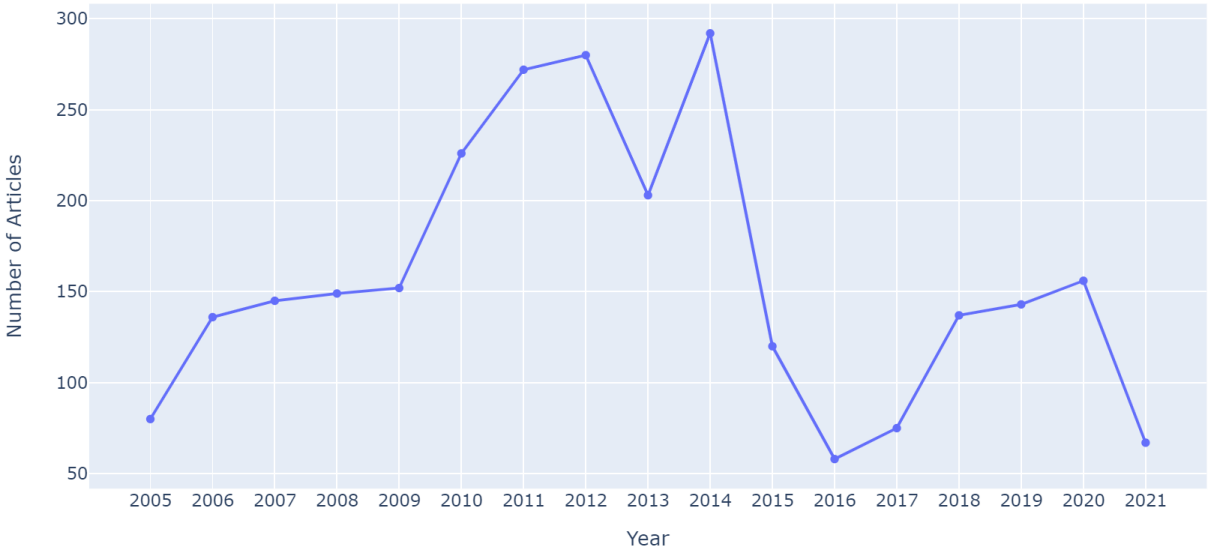

Figure 7: Number of published articles per year (raw data).

As shown in Figure 8, the article's length did not change much before 2014. However, most articles published between 2105 and 2018 seem to have shorter length compared to that of articles published before 2015 and after 2018. Furthermore, most articles published after 2019 seem to have longer text compared to articles published in other time periods.

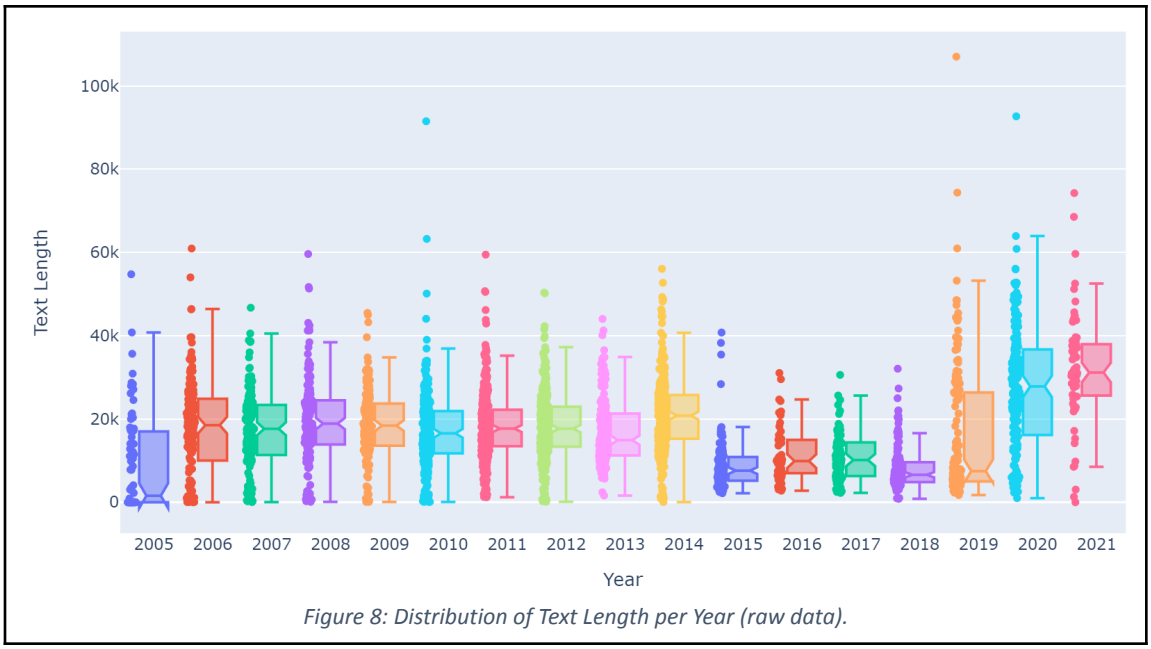

Figure 8: Distribution of Text Length per Year (raw data).

Data Preprocessing

Missing Values & Invalid Data Handling

Before performing data preprocessing, we look at the dataset's information and find that there are 40 examples that have missing values for the article's full text and there 5 examples that have missing keywords. Furthermore, we find two articles that have been flagged as plagiarism and two articles that have been noted as "retracted due to the request of the authors". Since the aforementioned examples contain either missing values or irrelevant data, we remove those examples from the dataset.

After studying the full texts of the dataset, we find a good number of examples that contain invalid data. These include irrelevant texts or texts that contain invalid characters or numerous symbol characters. Therefore, we remove all examples with the full texts that do not start with a word that has the first letter capitalized. This further reduces the size of the dataset to 1,811 articles.

Text Preprocessing

The following text preprocessing techniques have been applied to the abstracts, keywords, and article full texts:

- Remove hyperlinks, footnote texts, and figure captions.
- Convert texts to lowercase.
- Remove stop words.

For the abstracts and article full texts, lemmatization has been applied on the texts to retain meaning of the word in its original context. For the keywords, stemming has been performed on the text to transform the word to its simpler form.

Preprocessed Dataset

After performing data preprocessing, we save the cleaned data into a csv file. A table of summary statistics of the preprocessed data is shown in figure 9. As indicated in the table, the full text has 2,033 to 107,028 words and the preprocessed data includes the published articles between 2005 and 2021.

	Min	Max	Mean	Median	Standard Deviation
Number of Keywords	1	16	5	5	2
Title Length	14	208	82	80	26
Abstract Length	128	5149	1411	1266	666
Text Length	2033	107028	22214	20505	10104
Year	2005	2021	2012	2011	4

Figure 9: Summary statistics of preprocessed data.

Figure 10 shows the distribution of text length in the preprocessed dataset. Most articles have a text length that is close to 20,505 words. There are few articles that have a text length greater than 40,000 words.

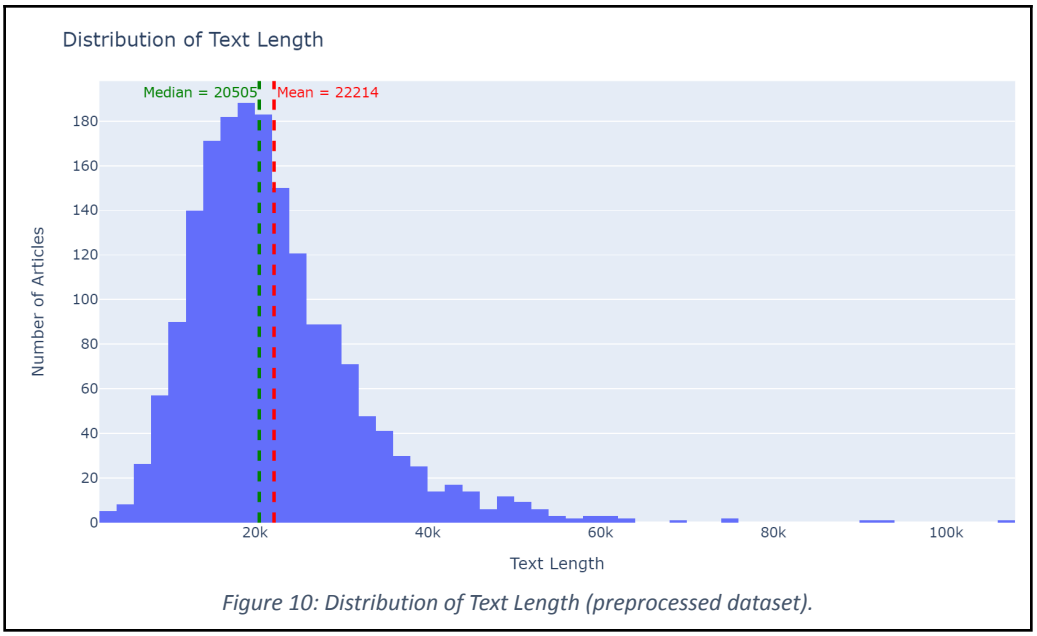

Figure 10: Distribution of Text Length (preprocessed dataset).

Figure 11 displays the distribution of articles per year in the preprocessed data. As shown in the figure, after performing data preprocessing, we lose three years of data. These are the articles that were published between 2016 and 2018.

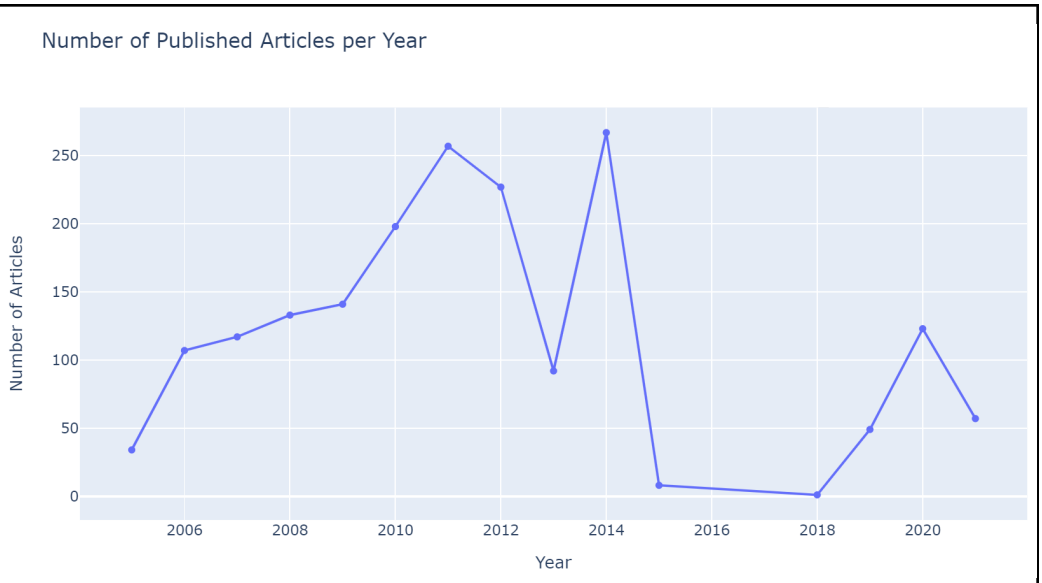

Figure 11: Distribution of number of articles per year (preprocessed data).

Models

Topic Identification Model

Model Development

The article, Topic Modeling with Gensim (Python) [3], was used for this portion. The current technology, Latent Dirichlet allocation (LDA), is used to extract multiple topics from multiple documents. This type of modeling is called “Topic Modeling”. This means that after creating the model from the dataset, the model creates a list of likely topics extracted from all of the documents.

Topic modeling is done using the Gensim Python library. Using the preprocessed data, the text is then made into bigram models using a Gensim function, gensim.models.Phrases(). Then a dictionary is created from the dataset which is used to create a bag-of-words representation of the bigrams. Once this is done, the LDA model is created. Using the model and some handy pyLDAvis functions, the topics extracted are visualized into an inter-topic distance map via multidimensional scaling. The graph is interactive. One can click on a mapped topic and see the saliency and relevance each token had on the topic. A rudimentary form of topic extraction will be used in the next iteration using term frequency just to see how it fares in comparison.

For the second increment, topic modeling was redone with updated modules. Also, some more text preparation and cleaning was also done to the already preprocessed text. This was done to ensure

the text was cleaned properly for the model. This extra cleaning did seem to help in creating a better model. Everything else was the same up until the modeling. The model was scored based on perplexity and coherence. After the creation of the visualization of the model results, an LDA mallet model was created with the our corpus to find an optimal number of topics for the LDA. One of the hyperparameters of the model is the number of topics to be extracted from the documents.

Some extra code was added that did not affect the model but had interesting results. For example, finding the dominant topic in each sentence was also done. This can be seen in the python notebook code. Finding the most representative document for each topic and topic distribution across documents were also done. As stated, these had no bearing on the model, but are just interesting things that can be done using the model.

Model Results

Considering the LDA model does not provide a topic for each article it is difficult to evaluate the results. One of the parameters for creating the model is the number of topics. Picking a number of topics to extract from the corpus is a whole subtopic of research. In the future the plan includes looking at how changing the number affects the results. The following image is the graphs created from the model. On the left is a mapping of each topic extracted. Each circle is a topic with size corresponding to the topic distribution found through the entire dataset. On the right is the “Top-30 Most Salient Terms”. Each topic has its own list of “Top-30 Most Relevant Terms” corresponding to the topic chosen on the left. Viewing the graphs on a Python Notebook environment allows for interactive viewing of each topic and the list of words associated with each topic.

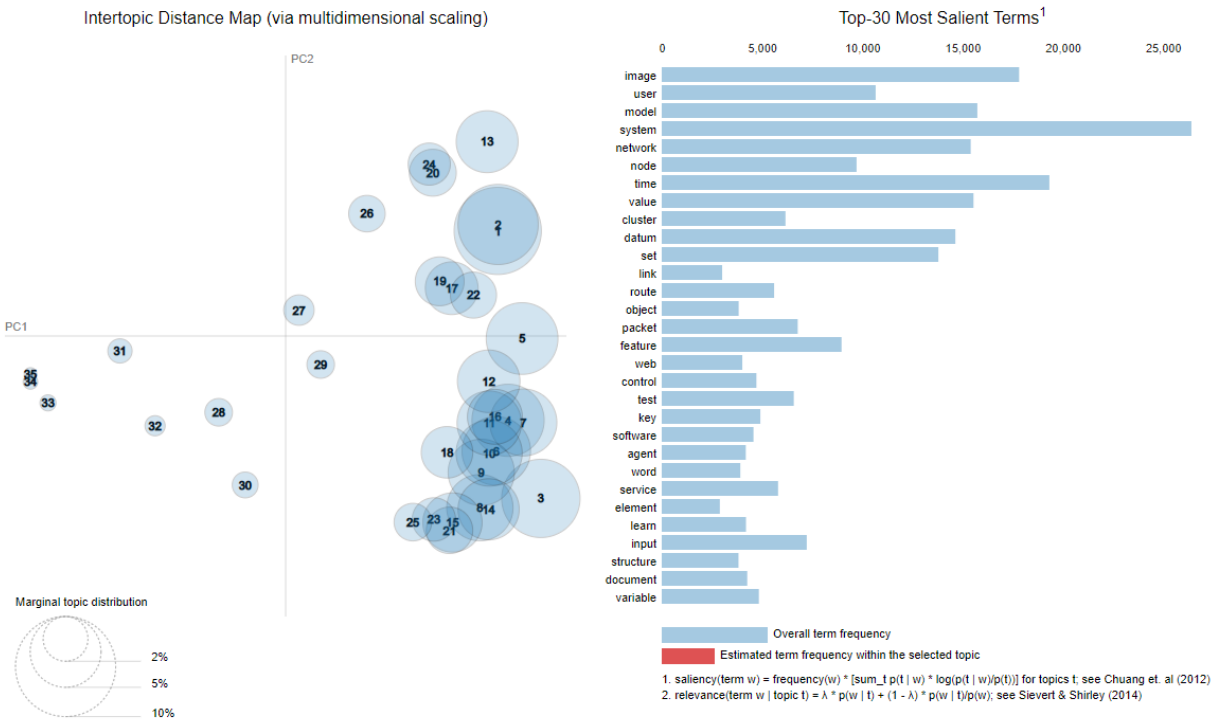

Figure 12: Intertopic distance map (left) and the most 30 salient terms (right).

As stated before, the coherence and perplexity scores were computed in this iteration. The model with a hyperparameter of thirty-five topics had a perplexity score of -7.2397 and a coherence score of 0.45696. The model wrapped with the mallet model had a coherence score of 0.48185. A line plot of coherence score per number of topics, denoted as k value, has been used to determine which k value is optimal for our documents (figure 13).

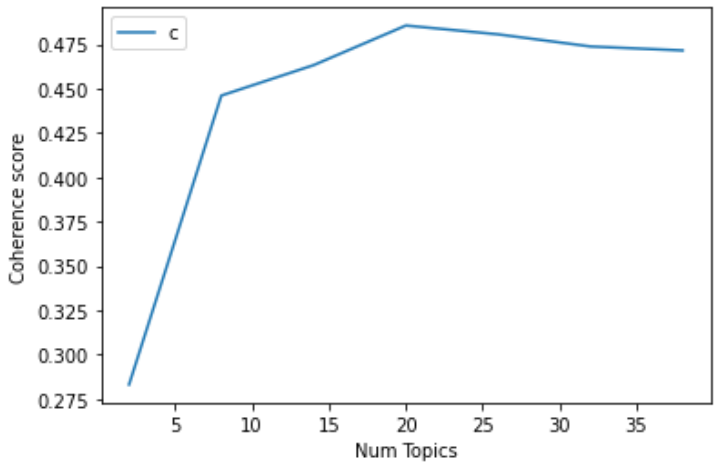

Figure 13: A line plot of coherence score per number of topics.

As shown in figure 13, the coherence score goes up starting from 2 topics and starts to level out a small amount at 10 topics. The coherence score starts to go down after 20 topics. From the graph above, one can determine that the optimal number of topics to choose for the corpus is 20 with a coherence value of 0.4856.

Text Summarization Model

Model Development

For our initial summarization model, we created a statistical method that uses unigrams to create extractive summaries. We begin by counting the frequency of each word in the given document. We then split the text into sentences, and then scored each sentence according to its term frequency. Term frequency is simply the sum of the frequency of every non-stop word in a sentence. To normalize these values and ensure that longer sentences aren't given priority over shorter ones, we divide the sum by the number of words in the sentence. Next, we determined a threshold for the term frequencies to determine which sentences should be included in our summary. Our base model simply uses the average of the term frequency scores. Finally, if a sentence is scored higher than the threshold, it's sequentially included in our summary. This method generates summaries but has several issues. Even on cleaned text, the model prioritizes words that occur frequently. Initially we thought this would be beneficial, but because of the nature of our data it was actually a hindrance. The summaries this method generates tend to be too long, and aren't particularly good. Specifically, the summaries tend to include sentences with dates and names that are referenced frequently, which is common in research papers.

Next, we implemented a version of Google's T5 Transformer model using Hugging Face. T5 is a transfer learning transformer model similar to BERT, GPT, and XLNET. T5 was trained on the C4 corpus, and can be used for many tasks, including classification, translation, question answering, and of course document summarization. Since the model is pretrained, we simply need to tokenize and encode the input, and allow the model to generate the output sentences using its previous knowledge.

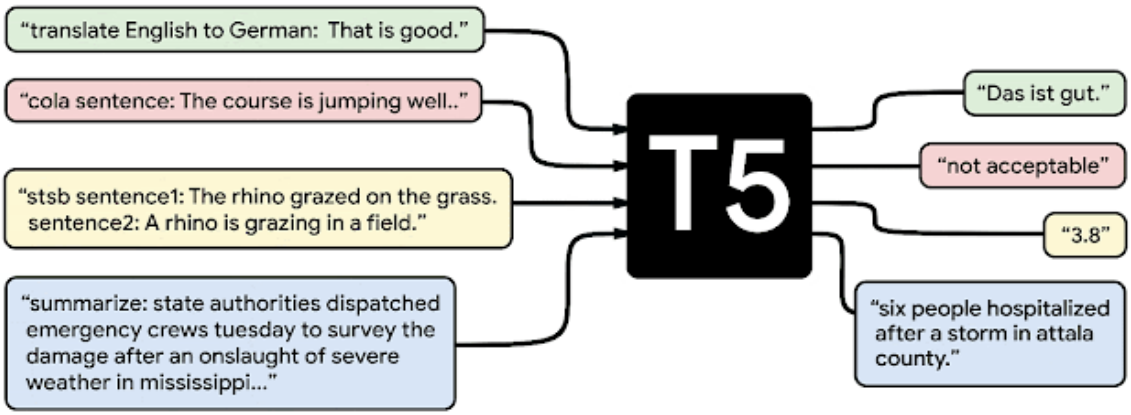

Image from https://ai.googleblog.com/2020/02/exploring-transfer-learning-with-t5.html

Model Results

Image 1: Output of statistical Model

```
Pallottino et al. (2002)  Yudhi Purwananto et al. Alonso-Ayuso et al. Alonso-Ayuso et al. (2011)
created groups of airplanes based on altitude levels with 1,000 ft distance between the groups.
Alonso-Ayuso et al. CPLEX is also a Mixed-Integer Programming (MIP) solver. Yudhi Purwananto et al. The
safe distance is measured in NM (Pallottino et al., 2002). 1a. 2a). 1a) is avoided. 1b). 1c). 1d). 2b).
3a). 3a). (a) (b)  Fig. 4a). 4b). 1d).
```

Image 2: Output of Google's T5 model

```
<pad> air traffic control system aims to increase the safety of the airplane passengers. the aim of the
CDR is to create a standard procedure to help the airplane controller. the proposed solution is called
the Velocity and Altitude Change (VAC) model. it uses a mixed-integer linear optimization (MILO)
approach to avoid conflicts.</s>
```

Image 3: Abstract of summarized article

An Air Traffic Controller (ATC) system aims to manage airline traffic to prevent collision of the airplane, called the Collision Avoidance (CA). The study on CA, called Conflict Detection and Resolution (CDR), becomes more critical as the airline traffic has grown each year significantly. Previous studies used optimization algorithms for CDR and did not involve the presence of cumulonimbus clouds. Many such clouds can be found in tropical regions like in Indonesia. Therefore, involving such clouds in the CDR optimization algorithms will be significant in Indonesia. We developed a CDR-based CA modelling that involves the Cumulonimbus (CB) clouds by considering three airplane maneuvers, i.e., Velocity, angle Turn and Altitude level Change (VTAC). Our optimization algorithm is developed based on a Mixed-Integer Programming (MIP) solver due to its efficiency. This proposed algorithm requires two input data, namely the initial airplane and cloud states input and the flight parameter such as velocity, angle and altitude levels. The outputs of our VTAC optimization algorithm are the optimum speed, altitude and angle turn of an airplane that is determined based on the currently calculated variables. Extensive experiments have been conducted to validate the proposed approach and the experiment results show that collisions between airplanes and clouds can be avoided with minimum change of the initial airplane velocity, angle and altitude levels. The VTAC algorithm produced longer distance to avoid collision between airplanes by at least 1 Nautical Mile (NM) compared to the VAC algorithm. The addition of angle in the VTAC algorithm has improved the result significantly. An Air Traffic Controller (ATC) system aims to manage airline traffic to prevent collision of the airplane, called the Collision Avoidance (CA). The study on CA, called Conflict Detection and Resolution (CDR), becomes more critical as the airline traffic has grown each year significantly. Previous studies used optimization algorithms for CDR and did not involve the presence of cumulonimbus clouds. Many such clouds can be found in tropical regions like in Indonesia. Therefore, involving such clouds in the CDR optimization algorithms will be significant in Indonesia. We developed a CDR-based CA modelling that involves the Cumulonimbus (CB) clouds by considering three airplane maneuvers, i.e., Velocity, angle Turn and Altitude level Change (VTAC). Our optimization algorithm is developed based on a Mixed-Integer Programming (MIP) solver due to its efficiency. This proposed algorithm requires two input data, namely the initial airplane and cloud states input and the flight parameter such as velocity, angle and altitude levels. The outputs of our VTAC optimization algorithm are the optimum speed, altitude and angle turn of an airplane that is determined based on the currently calculated variables. Extensive experiments have been conducted to validate the proposed approach and the experiment results show that collisions between airplanes and clouds can be avoided with minimum change of the initial airplane velocity, angle and altitude levels. The VTAC algorithm produced longer distance to avoid collision between airplanes by at least 1 Nautical Mile (NM) compared to the VAC algorithm. The addition of angle in the VTAC algorithm has improved the result significantly.

Evaluating summarization models is difficult. Two methods have been developed recently, Rouge and Bleu, but these models essentially evaluate the precision and recall of a model. Since we are comparing generated summaries to abstracts of articles, these methods won't be particularly useful. Further, there are no established libraries offering implementations of these metrics. Instead, I suggest we simply evaluate the performance of these models using our own knowledge of what a good summary might look like and include.

As can be seen from the images above, it is clear that Google's T5 model outperformed the statistical model. Despite having similar lengths, the statistical model does not include very much relevant information. It tends to focus on sentences in the original text that include dates and names that are commonly repeated. In contrast, T5 has actually generated unique sentences that summarize content in the article despite being given as input and encoding that counts occurrences of tokens,

similar to the statistical model. This really demonstrates the power of deep learning, transfer learning, and transformer models. In the end, we decided to include only the T5 model in our Web App.

Web Application Development

For web application development, the following technologies have been used: Dash, Plotly, and WordCloud. Firstly, Dash is used to build a web-based application in Python. Secondly, Plotly is used to create visualizations. Thirdly, WordCloud is used to enable analysis of high frequency terms in the input text. Figure 13 displays a screenshot of the application.

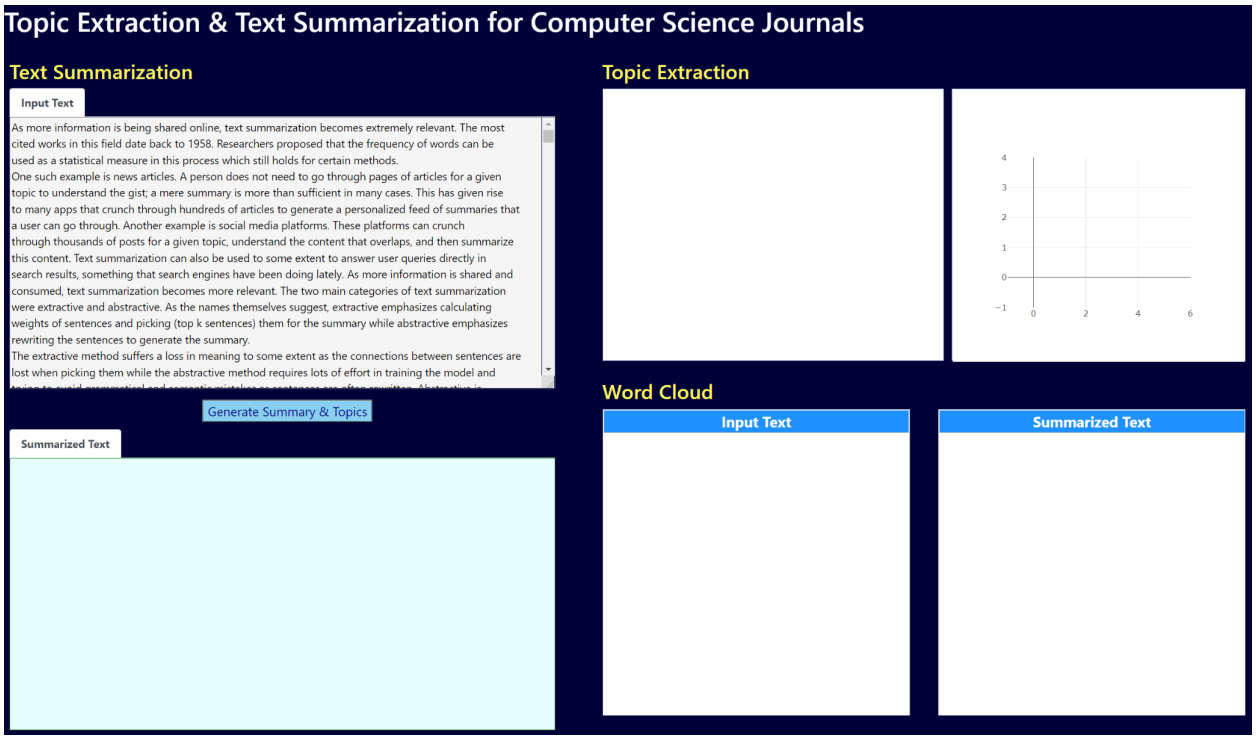

Figure 13: A screenshot of a web application for text summarization and topic extraction. The full text of an article related to text summarization has been pre-loaded into the input textbox for summary generation and topics extraction.

As shown in figure 13, the application has three main sections: 1) Text Summarization, 2) Topic Extraction, and 3) Word Cloud. Text summarization section contains an input textbox which enables the user to enter or paste text for summary generation and topics extraction. Once the user provides the input text and clicks on the button “Generate Summary & Topics”, the application would display the summarized text, generated by the T5 model, in the light green box beneath. When the summarized text is available, the application would show a word cloud of input text and a word cloud of summarized text in the Input Text subsection and the Summarized Text subsection of the Word Cloud section,

respectively. In the meantime, five LDA models would be trained on a different number of topics ranging from 1 to 5. An LDA model that has a highest coherence score would be selected for performing topic extractions. Once the topics have been identified, a topic table that includes the terms associated with each topic would be displayed in the first box under the Topic Extraction section. At the same time, a figure that displays the line plots of coherence scores and perplexity scores would also be shown in the second box under the Topic Extraction section. Figure 14 shows a screenshot of the web application after the user clicks on the "Generate Summary & Topics" button.

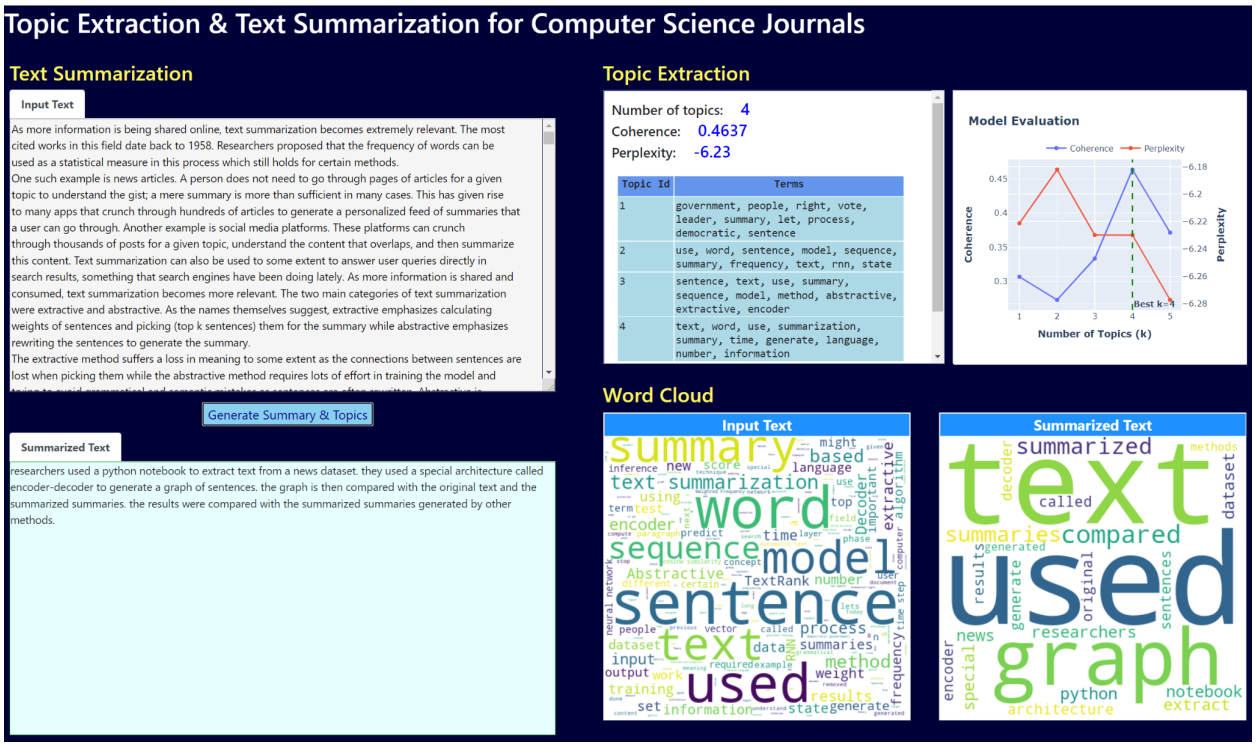

Figure 14: A screenshot of the web application after the user clicks on the button "Generate Summary & Topics".

Project Management

Work Completed

Team Member	Responsibility	Issues or Concerns	Contribution Percentage
Nestor Molina	• Perform Topic Extraction/Modeling using LDA • Record relevant portion of project • Contribute to related works • Contribute to Increment report		33%

Ngoc Phan	• Perform data collection, data preprocessing, and data analysis. • Build web application • Format and proofread the document for submission. • Compile the recording videos for submission.		34%
William Baker	• Text summarization of both clean and original text with two different models • Algorithm research • Recorded relevant portion of project update • Relevant portion of project increment report		33%

References

[1] *Writing an Abstract for Your Research Paper*. (n.d.). The Writing Center | University of Wisconsin – Madison. Retrieved September 20, 2021 from https://writing.wisc.edu/handbook/assignments/writing-an-abstract-for-your-research-paper/

[2] *Journal of Computer Science*. (n.d.). Science Publications. Retrieved September 20, 2021 from https://thescipub.com/jcs/archive

[3] Prabhakaran, S. (2018). *Topic Modeling with Gensim (Python).* Machine Learning Plus. Retrieved October 30, 2021 from https://www.machinelearningplus.com/nlp/topic-modeling-gensim-python/.

[4] Silipo, R. (2019, February 18). *LDA for Text Summarization and Topic Detection*. DZone AI. Retrieved October 30, 2021 from https://dzone.com/articles/lda-for-text-summarization-and-topic-detection.

[5] Dehru, Virender, et al. (2021). Text Summarization Techniques and Applications. *IOP Conf. Series: Materials Science and Engineering*, vol. 1099, 2021, pp. 3-8. https://doi.org/10.1088/1757-899X/1099/1/012042

[6] Panchal, A. (2021, February 8). *Text summarization in 5 steps using NLTK*. Medium. Retrieved October 31, 2021, from https://becominghuman.ai/text-summarization-in-5-steps-using-nltk-65b21e352b65

[7] Xu, J., & Durrett, G. (2019). Neural extractive text summarization with syntactic compression. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. https://doi.org/10.18653/v1/d19-1324