Fall 2022

Abstractive Text Summarization for Tweets

Siyu Chen

Abstractive Text Summarization for Tweets

A Project Report

Presented to

The Faculty of the Department of Computer Science

San Jose State University

In Partial Fulfillment

of the Requirements for the Degree of

Master of Science

By

Siyu Chen

Fall 2022

The Designated Committee Approves the Master's Project Titled

ABSTRACTIVE TEXT SUMMARIZATION FOR TWEETS

by

Siyu Chen

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSÉ STATE UNIVERSITY

December 2022

Dr. Chris Tseng	Department of Computer Science
Dr. Genya Ishigaki	Department of Computer Science
Dr. Nada Attar	Department of Computer Science

ACKNOWLEDGMENTS

My sincere gratitude goes out to Dr. Chris Tseng, who is my project adviser, for his knowledge, patience, direction, and inspiration. I deeply appreciate all of his support and supervision, as well as his willingness to answer emails even on weekends and holidays. I would like to express my gratitude to Dr. Genya Ishigaki and Dr. Nada Attar for their contributions and suggestions. Finally, I would like to thank my family and friends for their thoughtful support.

ABSTRACT

In the high-tech age, we can access a vast number of articles, information, news, and opinion online. The wealth of information allows us to learn about the topics we are interested in more easily and cheaply, but it also requires us to spend an enormous amount of time reading online. Text summarization can help us save a lot of reading time so that we can know more information in a shorter period. The primary goal of text summarization is to shorten the text while including as much vital information as possible in the original text so fewer people use this strategy on tweets since tweets are commonly shorter than articles or news. However, as social networking software becomes more widespread, Text summarization can assist us in swiftly reviewing a large number of comments and discussions. In this project, we applied fuzzy logic and a neural network to extract essential sentences, followed by an abstraction model to provide a summary. Summaries generated by our model contain more vital content and obtain a better ROUGE score than classic abstraction models since we extract the crucial information first; summaries generated by our model are more similar to human-written summaries than traditional extraction models because we are using an abstract model. In the end, we provided a web-based application to display our model more interactively.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

In the discipline of Natural Language Processing, text summarization is a formidable challenge, because producing a quality summary necessitates accurate text analysis such as semantic and lexical analysis [1]. It can benefit corporations, celebrities, and consumers in efficiently summarizing articles, news, feedback, comments, and postings related to the information they care about, resulting in a major improvement in user experience and quality of life.

On the other hand, social media has grown increasingly significant in our everyday lives, to the point that it has replaced traditional forms of communication. This has become even more apparent as a result of the Covid-19 pandemic and the sudden cessation of in-person interactions. Online communication and the usage of social media in changing times have expedited technological adaptation and online socializing. The shift in communication channels has also influenced feedback gathering. For example, if someone wanted feedback on a thought or an idea, they could ask their close friends, family, or coworkers for input but would be severely limited from data sources outside of their social circle without massive input of both time and other resources. One of the advantages of social media includes the ability to receive and search for input from both within and outside of a person's social circle for all-encompassing and broad feedback on a subject matter. Not only does this broaden the regular scope of feedback from a person's social circle to the entire userbase of a popular social media platform but would also drastically speed up the process of data collection. However massive data collection also has its drawbacks. While a single user or a team of data analysts can process and summarize all the data if given enough time, it would never be able to match the processing speed of a computer if there was a model that can summarize and give an abstract on a specific matter.

This project will be focusing on using fuzzy logic with neural networks to improve the performance of an abstractive model that can process not just articles but also tweets, and comments from a singular post into an abstract or summary that encompasses all the data provided. Unlike summarizing a book or an article that often has one singular opinion, a tweet would often have conflicting feedback, multiple stances, and variation in the severity of opinions, thus making the need for an AI model even more apparent due to the increase in difficulty. We also compared the performance with extractive models using different types of datasets using the ROUGE score. We also created a web-based application for our model using Drupal and Django.

2. BACKGROUND AND PRIOR WORK

In this section, we will provide an overview of text summarization, natural language processing, and fuzzy logic as well as discuss and compare some popular models for two main methods used for automated summarization: abstraction and extraction. In addition, we will explore related natural language processing techniques and how fuzzy logic can be applied to improve the model's performance.

2.1 Automatic Text Summarization

Text summarization (TS) is one of the most challenging topics in Natural Language Processing (NLP) [1]. The basic idea is to find the most informative phrase or sentence in a document. It can be applied to different types of text like articles, news, books, novels, emails, and Tweets to reduce reading time and analyze a large set of text. According to El-Kassas, the earliest research on ATS was from Luhn in 1958. Luhn's algorithm exploits abstracts of

magazine articles and technical publications automatically [1]. Since Lun's Publication in 1958, text summarization has been developed into many different types and directions, as illustrated below in Fig 1.

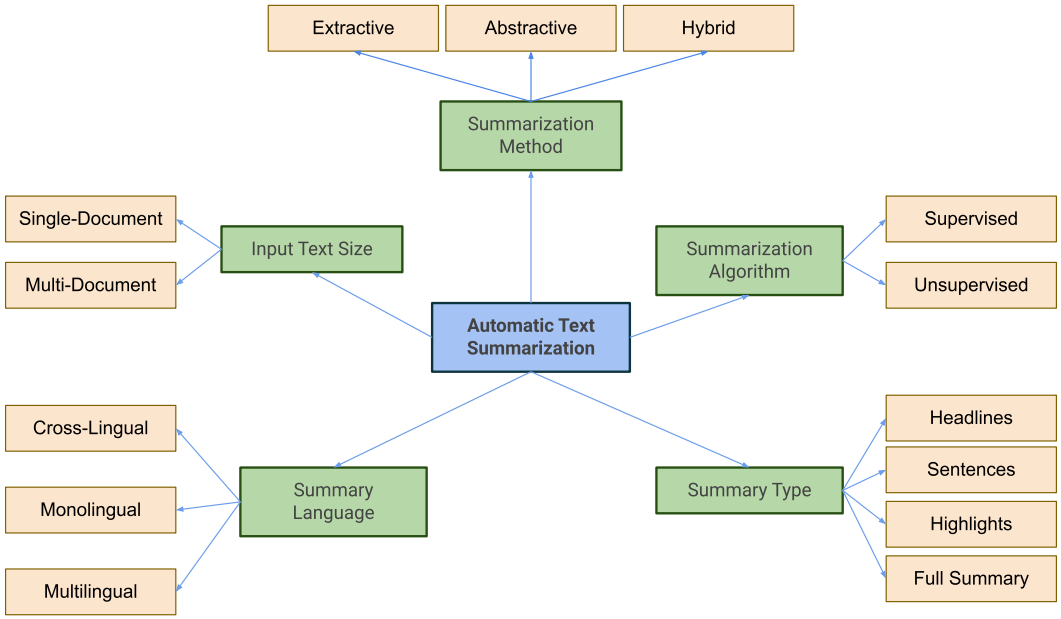

Fig.1. Automatic Text Summarization Types

Based on the summarization method, we have three major approaches: extractive, abstractive, and hybrid. Extraction-based summarization generates a summary from a subset of existing sentences or phrases in the given text. A simple model architecture is shown in Fig.2, sentences' scores will be calculated based on the analysis from the pre-processing stage, and the length of the summary can be adjusted by providing a maximum value as the cutoff. Extractive summarizing is simple and resilient, it is used in the majority of popular summarizing systems[2]. On the other hand, summaries generated using the extractive method may be quite different from human write summaries especially when the given text is short. The result may contain redundant sentences, temporal expressions disagreeing, and conflicting information as well as a lack of semantics and cohesiveness in summary sentences as a result of poor sentence

linking and unresolved co-reference connections [1]. There are many available models online extractive, like TextRank, Sumy, LexRank, and SpaCy.

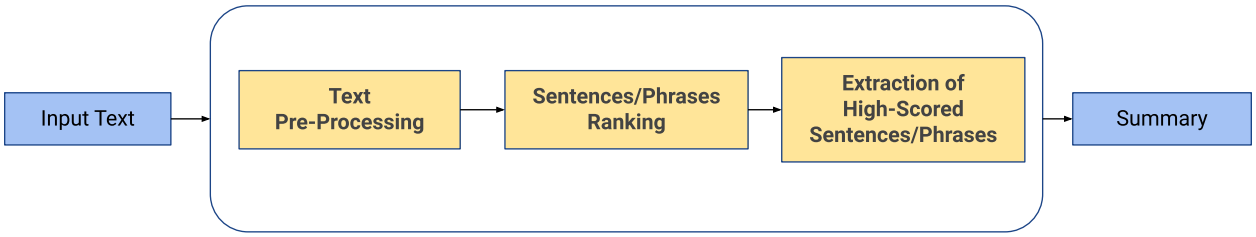

Fig.2. Simple architecture of extractive summarization model

Abstraction-based summarization will generate the sentences of the summary by itself instead of extracting it from the input. To be able to create new phrases and sentences, a deeper understanding of the source's fundamental idea is necessary. To construct the summary, we would need to develop an internal semantic representation, as illustrated in Fig.3; some common methods include graph-based, tree-based, rule-based, and deep-learning-based. Unlike extractive models, it can include words not found in the given source based on fusion, paraphrasing, and compression [1] which can help reduce redundancy and summary length. One of the major drawbacks is the lack of sophistication therefore high-quality models would often default to other methods, and require more data to train compared to the extractive method [1]. Recent abstractive summarizing attempts have mostly focused on the use of deep learning models, particularly in short text summarization [3]. Etemad compared fifteen models using deep learning methods including CNN, Seq2Seq RNN, Convolutional Seq2Seq, BERT, Transformer, LSTM, etc; Zhang's Convolution Seq2Seq model and Raffel's T5 model outperform in multiple datasets. He mentioned most of the researchers are focused on solving the two major difficulties of text summarization: syntactic and semantic [4].

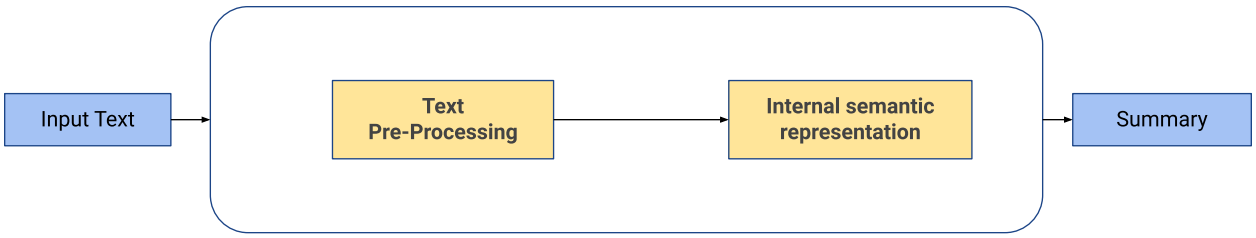

Fig.3. Simple architecture of abstractive summarization model

Lastly, the hybrid model is the combination of both previous methods by extracting the important sentence using extractive methods and summarizing it using abstractive methods. It contains the advantage of both abstractive and execrative models but produces a lower quality abstractive summary than the pure abstractive technique since the resulting summary is based on extracts rather than the source text [1].

Based on the input size, we will have two types of summarizers. A single document summarizing will be much easier since the input is shorter and usually has a primary theme. Because multi-document summarization provides a summary based on more than one input, it may include contradictory information, and will be difficult to incorporate all information in summary.

Text summarization is not limited to English, so models can be classified based on input and output languages. When the source text and output summaries are both in the same language, we call it monolingual; when we want to generate summaries in different languages than the input, we call it cross-lingual; and when the input source included more than one language, we call it multilingual. Monolingual would be the easiest to build because it does not require the model to handle more than one language or translation procedure.

Next, we will talk about systems depending on the summary type. Some of the more common types are headlines, highlights, sentences, and full summaries. Headlines and highlights are commonly used to offer high-level overviews of the sources. The abstractive approach is

typically used by sentence-level summarizers, and complete summary summarizers are most commonly led by the needed compression ratio or summary length [1].

Last but not least, unsupervised algorithms do not require training data and phase, unlike supervised algorithms. The latter requires texts with human-generated summaries to identify which features are more essential to the sentence. Term frequency, the number of proper nouns in the sentences, the position or length of the sentences, and so on are examples of features.

2.2 Text Pre-Processing

As noted previously, text pre-processing is an important part of TS. It transfers sentences and paragraphs into structured data based on the needs of the model. According to Widyassari's research about the pre-processing methods used in TS over the last ten years, the most frequently used are stop word removal, stemming, tokenizing, sentence segmentation, sentence selection, lemmatize, team weight, word frequency, sentence by term matric, word segmentation, normalize, paragraph segmentation, pos tagging, proper noun set, and a bag of words [5].

Stop word removal is a method to remove meaningless and common words from inputs. Usually includes conjunctions (like "yet", "for", "and"), pronouns (like "he", "her", "you"), prepositions (like "in", "by", "about"), and articles (like "a", "the", "an"). By filtering stop words, we can reduce the dataset size and the model will focus more on relevant information. To be able to use the method, we will need to come up with a list of stop words. Many users on Github give free stopword lists; Python libraries such as NLTK, spaCy, Gensim, and Scikit-learn also include stopword filtering functions.

The bag of words technique would convert sentences into a collection of word tokens with how many times the word appears in the text. It is useful when calculating the team weight,

and word frequency. However, the order of the word will be ignored. If we would like to keep the relationship between words, N-gram will be a better fit. N-grams return a collection of sequences of n-words.

Stemming and Lemmatize are also frequently used for TS. Both approaches are used to normalize words so we can translate them to their basic format. The difference is that stemming is a rule-based technique that will chop out the extra part of a word without considering morphological analysis or context information; lemmatization will reduce a word back to its dictionary form depending on the context [5]. For example, “change, changing, changes” will become “chang” after stemming but “change” after lemmatization. Stemming may cause misunderstandings on some words or when over-stemming or under-steaming. For example, “caring” or “universe” will transfer to “car” and “universe” instead of “care” and “universe”. Even though it looks like lemmatize is more accurate than stemming, stemming will run faster and be easier to implement.

The last method I want to point out is phrase chunking. It works on top of part of speech tagging (POS tagging) and will divide a sentence into its sub constituents like coordinating conjunction, verb, adverb, noun (single or plural), pronouns, adjectives, etc with notation. Libraries like NLTK, and spaCy provide functions to chunk sentences easily.

2.3 Fuzzy Logic

After a long period of development, the sentences/phrase ranking stage in TS has been improved to consider more and more features. The earliest model created by Luhn extracts summaries from documents based on high-frequency terms [6]; later on, developers discover that information such as locations, length, sentence similarity, numerical data, and so on should also

be considered. In Widyassari's paper, she reviewed TS models from 2008 to 2019 and presented the top ten features used are title word, keywords, thematic word, proper noun, numerical data, sentence position, sentence length, sentence certainly, semantic term, and frequent semantic [5]. However, classic logic is restricted to two values that can not be applied to all situations. For example, a sentence with three thematic words has more value than one with one thematic word. In the standard ways, however, there is no distinction between these two phrases. Also, some features may be more important than others. How can we design which one has a higher score, if we have a shorter sentence that includes two numerical data and a longer one with one proper noun? Fuzzy logic would be the key to overcoming this problem. It can derive an imprecise and possible result from a potential and imperfect beginning set because it uses relative reasoning logic [7]. With fuzzy logic, we can also assign weight and provide membership functions to each feature to solve the inequality problem of features [6].

To investigate if the fuzzy logic method can improve the performance of TS, Suanmali analysis results from a fuzzy-logic-based text summarizer, general statistic method, Microsoft Word 2007 Summarizer, and baseline summarizer with eight features: proper noun, sentence length, sentence position, title feature, term weight, sentence similarity, numerical data and thematic word[6]. She used DUC 2002 as a dataset and used ROUGE-1 scores to do the comparison. Her results show that the fuzzy summarizer has the best score in average precision, recall, and F-measure [6].

3. DATASET

For this project, datasets must include summaries that can be compared to the one generated using our model. Because our experiment comprises both extractive and abstractive

models, we'd like to have a dataset for each. To execute the experiments using the Tweet dataset, we discovered a dataset of 1100 dialogs from Twitter customer service discussions. Each dialog contains three extractive and three abstractive summaries generated by humans. The BBC News dataset includes 2225 items from the BBC's new website, divided into five distinct areas (technology, sport, politics, business, and entertainment), each with an extracted summary. It has no connection with tweets, but we'd want to test the model with a different type of dataset so we can compare the Neural Network results and investigate the impact of dataset type on feature weights.

Name	Size	Description	Summaries Information
TweetSum	1100	A collection of 1,100 dialogs rebuilt from Tweets in the Kaggle Customer Support On Twitter dataset https://github.com/guyfe/Tweetsumm	• Created by humans • 3 extractive summaries • 3 abstractive summaries
BBC News	2225	The data utilized in the research by D. Greene and P. Cunningham are from the BBC news website and relate to stories in five topical areas from 2004 to 2005 for "Practical Solutions to the Problem of Diagonal Dominance in Kernel Document Clustering," Proc. ICML 2006; the BBC owns all rights, including copyright, to the original articles' content, visit http://mlg.ucd.ie/datasets/bbc.html for further information.	• Extractive • Articles and summaries are divided into 5 topics

Fig.4 Dataset Information

4. DESIGN AND IMPLEMENTATION

This project will mainly use Python to build the text summarization model and use Drupal and Django to build a web-based application for demonstration. The main step for the TS model is shown in Fig.6. We will first pre-process the source text and apply fuzzy logic with the proper weight we get from the neural network then summarize it using abstractive models and

compare results using ROUGE scores. The web-based application will be hosted in localhost to avoid fees during the development and testing.

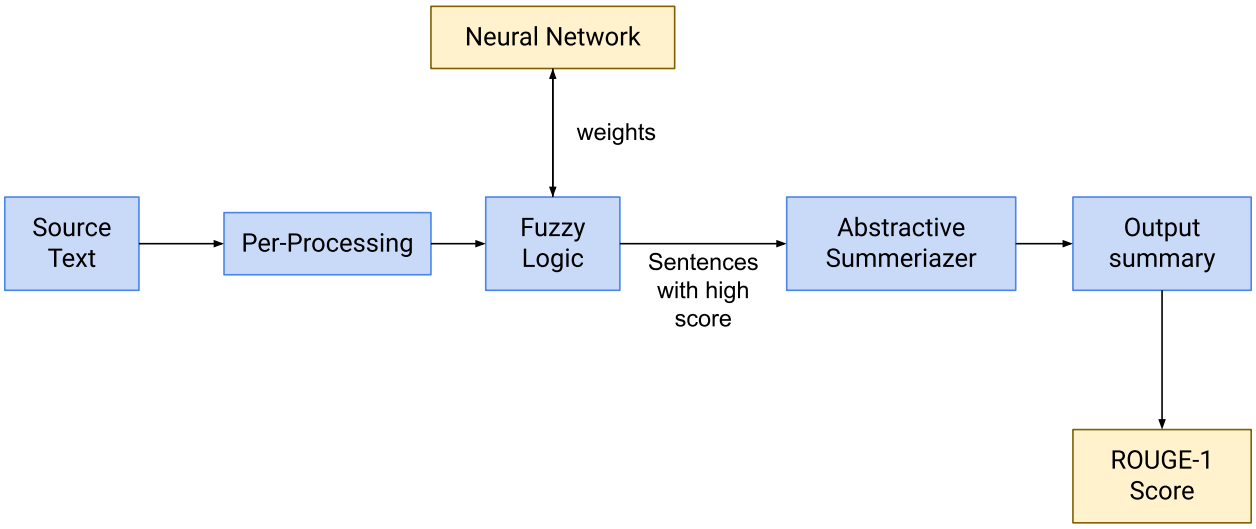

Fig.5. Project Architecture

4.1 Data Pre-Processing

We will be pre-processing the dataset using Python libraries. Based on the need of our model, we decided to apply to “stop word”, “bag of words”, “noun-phrase chunking”, and “word to number” methods. For “stop word” and “bag of words”, we used CountVectorizer from Sklearn. It contains a list of English stop words and allows us to choose if we would like to convert all characters to lowercase. To convert numerical words into numbers for later extraction, we used a library called word2number in Python. For noun-phrase chunking, we used RegexpParser, from NLTK. Fig.8 shows an example of a text after pre-processing, the input text is a sentence from the BBC News dataset.

Text: *"A US government claim accusing the country's biggest tobacco companies of covering up the effects of smoking has been thrown out by an appeal court."*

Bag of Word: *['US', 'accusing', 'an', 'appeal', 'been', 'biggest', 'by', 'claim', 'companies', 'country', 'court', 'covering', 'effects', 'government', 'has', 'of', 'out', 'smoking', 'the', 'thrown', 'tobacco', 'up']*

Stop Word: *['US', 'accusing', 'appeal', 'biggest', 'claim', 'companies', 'country', 'court', 'covering', 'effects', 'government', 'smoking', 'thrown', 'tobacco']*

Noun-Phrase Chunking: `grammar = "NP: {<DT>?<JJ><NN|NNS>+}"`
[('A', 'DT'), ('US', 'JJ'), ('government', 'NN'), ('claim', 'NN'), ('accusing', 'VBG'), ('the', 'DT'), ('country', 'NN'), ("'s", 'POS'), ('biggest', 'JJS'), ('tobacco', 'NN'), ('companies', 'NNS'), ('of', 'IN'), ('covering', 'VBG'), ('up', 'RP'), ('the', 'DT'), ('effects', 'NNS'), ('of', 'IN'), ('smoking', 'NN'), ('has', 'VBZ'), ('been', 'VBN'), ('thrown', 'VBN'), ('out', 'RP'), ('by', 'IN'), ('an', 'DT'), ('appeal', 'JJ'), ('court', 'NN'), ('.', '.')]

Text: *"He will spend ten million for two cars"*

Word2Number: *[10, 1000000, 2]*

Fig.6. Example of a text after pre-processing

4.2 Fuzzy Logic

After pre-processing, we used eight features to calculate feature scores (zero to one) of each sentence.

1. Title/Keyword: We believe that words appearing in an article's title are likely to include relevant information. For text without titles like tweets, we can use this feature for topic-based summarization. We count the number of words in the sentence that matches with title/keyword, divided it by the total number of words in the title/keyword, and divided it with the maximum score:

$$F1 = \frac{\#of\ keyword\ in\ sentence}{Total\ \#\ of\ keyword} / max$$

2. Top 5 Frequency Words: High-frequency words are highly related to the document. We got the top five frequency words from the text. To avoid high-frequency but meaningless words ("an", "the", and "is") to be selected as the top five, we removed stop words in pre-processing stage. The score is calculated by the number of top-frequency words

appearing in the sentence over the maximum number of top-frequency words in the sentence for the whole article.

$$F2 = \frac{\#of\ high\ frequency\ word\ in\ sentence}{Max\ \#\ of\ high\ frequency\ word\ in\ sentence}$$

3. Number of Proper Nouns: Sentences containing more proper nouns (person, organization, place, etc) are sentences carrying more significant information [9]. To calculate the number of proper nouns in a sentence, we applied noun-phrase chunking in the text pre-processing step. To be fair with shorter sentences, we used the total number of proper nouns in a sentence to divide by the length of the sentence.

$$F3 = \frac{\#of\ proper\ nouns\ in\ sentence}{Length\ of\ sentence}$$

4. Number of Numerical Data: A statement containing numerical data is significant, and it is most likely included in the reference summary. We converted all numerical words into integers in the pre-processing stage. For words like "one hundred and three", it will turn into "1 100 and 3". We scored sentences depending on the existence of numerical data to avoid double counting.

$$F4 = 1\ (inculed)\ ||\ 0\ (not\ inculed)$$

5. Sentence Length: This feature is good to filter out short sentences like journalist and datelines [9]. Also, while Twitter provides an abundant amount of information that can be analyzed for text summarization, it goes without saying that it also contains large amounts of irrelevant and unnecessary sentences such as common texting abbreviations, acronyms, and slang. This feature can help users to lower the chance of those sentences appearing in the output summary. We calculated the score by using the length of the sentence divided by the length of the longest sentence in the document.

$$F5 = \frac{Length\ of\ sentence}{Length\ of\ longest\ sentence}$$

6. Sentence-to-Sentence Similarity: This feature calculated the similarity of a sentence with all other sentences in the articles. Important material will be discussed or mentioned several times in a document, thus if a sentence is quite similar to previous sentences, it is likely to be connected to the primary topic.

$$F6 = \frac{similarity\ of\ the\ sentence}{highest\ similarity\ in\ all\ sentence}$$

7. Sentence Location: Since the first five sentences in the text could include crucial information, we score them by dividing the sentence's position by five and assigning zero to all other sentences.

$$F7 = (5\ /\ position\ of\ the\ sentence)\ \ for\ frist\ five\ sentence$$

8. TF/IDF Score: The term frequency-inverse document frequency is a score to reflect the importance of a word to a document. The TF/IDF score of a sentence can be computed by summing the TF/IDF scores of all words in the sentence [9]. To rank the sentence based on TF/IDF score, we will use the score of a sentence to divide the highest score.

$$F8 = \frac{TF/IDF\ score\ of\ the\ sentence}{highest\ TF/IDF\ score\ in\ all\ sentence}$$

After defined all features, we used Python library Scikit-fuzzy for the implementation and designed to use Gaussian as membership function. The equation of the function is:

$$\mu(x) = e^{-f1*(x-f2)^2}$$

f1 is the spread and f2 is the midpoint; the function is beneficial if the membership is close to a certain value [10]. For each feature, we designed the value for “poor”, “average”, and “good”; Fig.7 shows a graph of what the Title/Keyword feature with membership function looks like after we defined the values. Next, we created IF-THEN rules for the engine. It is the most important

part of the inference engine, our features criteria are used to extract the important sentences from these regulations [9]. A simple rule will be: IF (F1 is important) and (F2 is important) and (F3 is not important) and (F4 is medium) and (F5 is important) and (F6 is medium) and (F7 is important) and (F8 is not important) THEN (important). Fig.8 is an example of one sentence's score after being applied to two features in the rule.

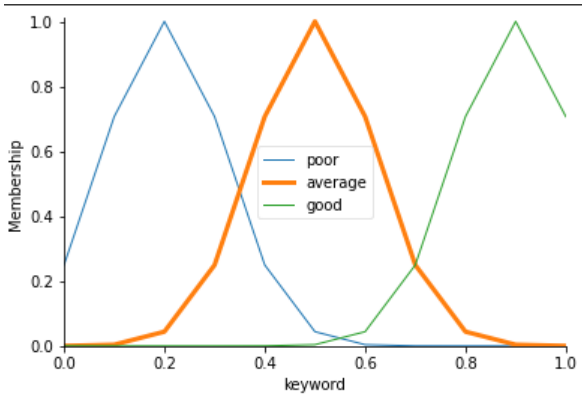

Fig.7. Sample Fuzzy Membership Graph

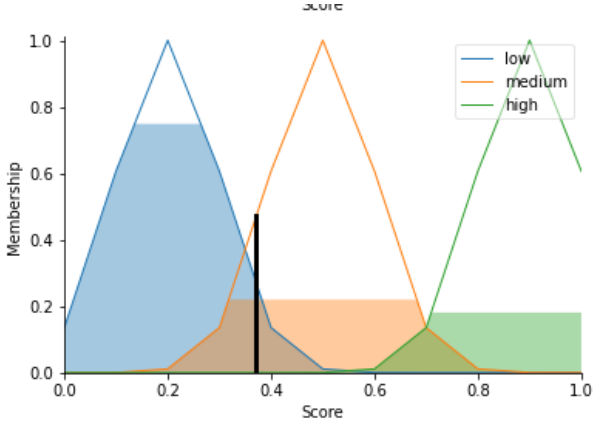

Fig.8. Sample Fuzzy Score Graph

4.3 Neural Network

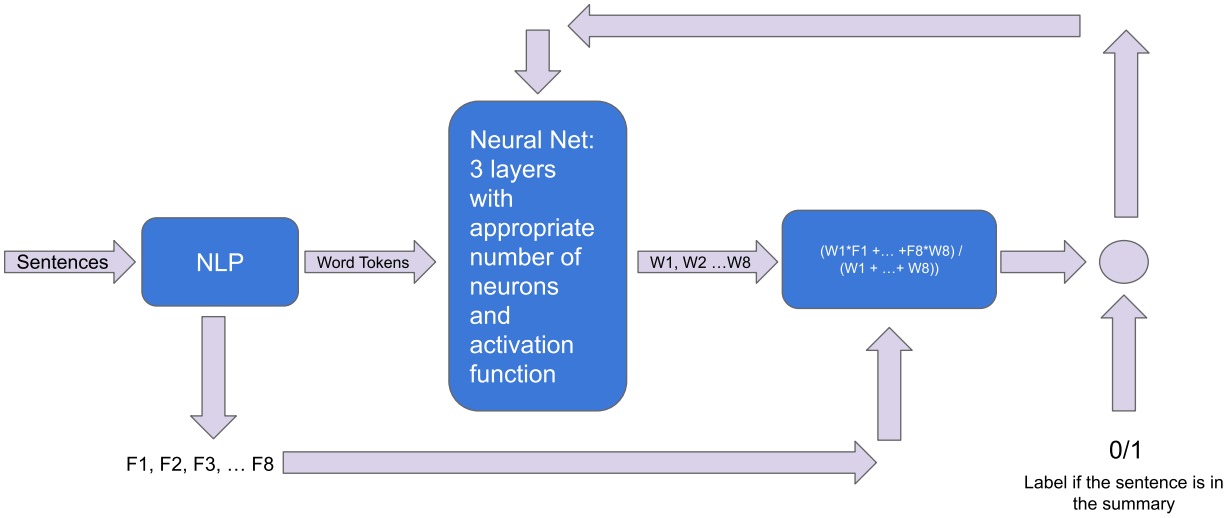

Fig.9. Neural Network Architecture

When we use different types of data, a particular feature may be more important. For example, when we handle tweets, "sentence location" may not be as important as "number of proper nouns" since tweets are usually short. To make sure our model can overcome situations like this, we used a neural network model to find out the importance of each feature and designed our fuzzy logic rules based on that.

First of all, we created a model using Tensorflow Kears. The model (see Fig.9) takes sentences as input and included three layers with an appropriate number of neurons. The activation function of the first two-layer is ReLU and we used sigmoid for the last hidden layer. The output of the model will be the eight weights. Unlike the common neural networks model, the training target we fit into the model are not weights of features but 0/1 to represent if the sentence exists in target summaries.

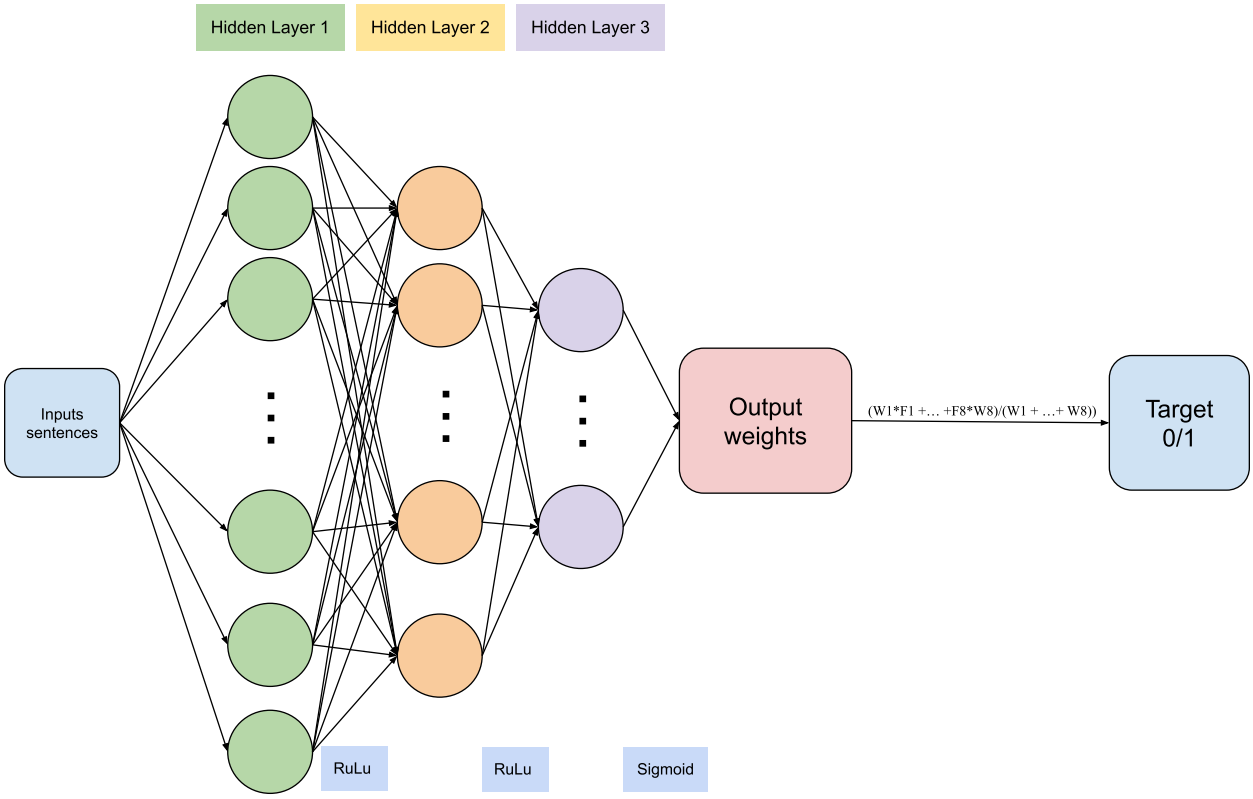

Fig.10. Example of Neural Network Set Up

To handle the difference between output size and target we created a custom loss function and metric function. During the stage we compare the prediction with the target, we included feature scores in the custom functions and calculate the predicted target using the formula below:

$$y_{pred} = (W1 * F1 + \ldots + F8 * W8)/(W1 + \ldots + W8))$$

Tthe losses are calculated as:

$$loss = \frac{1}{2} * (y_{target} - y_{pred})^2$$

The metrics are calculated as:

$$metrics = round(y_{pred}) == y_{target} \text{ (True/False)}$$

From the sample graph(Fig.11, Fig12, Fig.13) of some weight's values when training with 2984 sentences from BBC News Summary with 50 epochs, we can see that the draft weights are around 0.5. Near the end of the experiment, the weights of all sentences are getting closer to a final value.

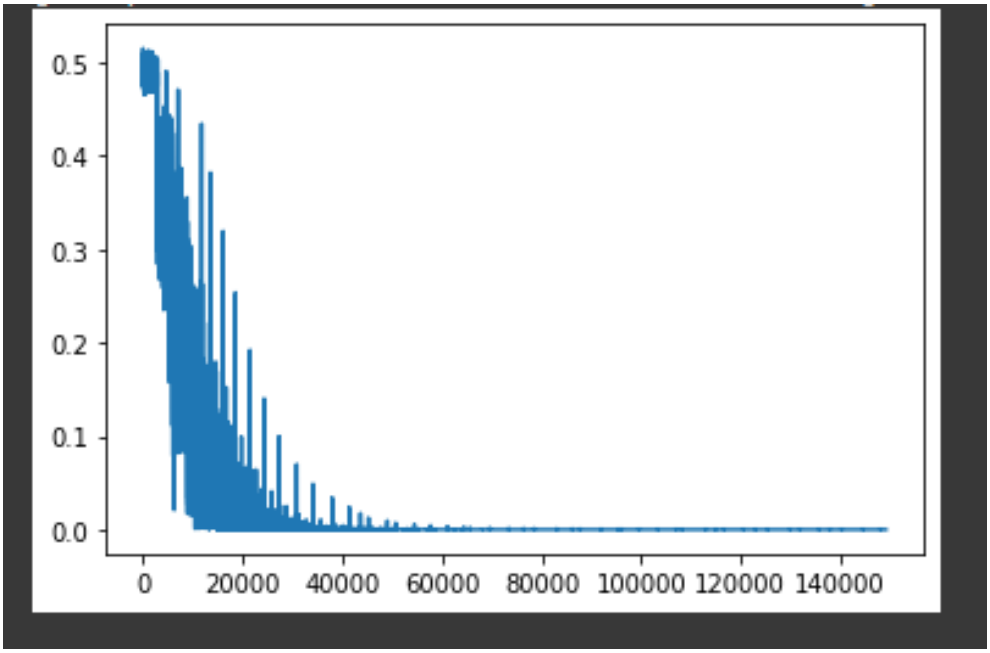

Fig.11 Example of Unimportant Result

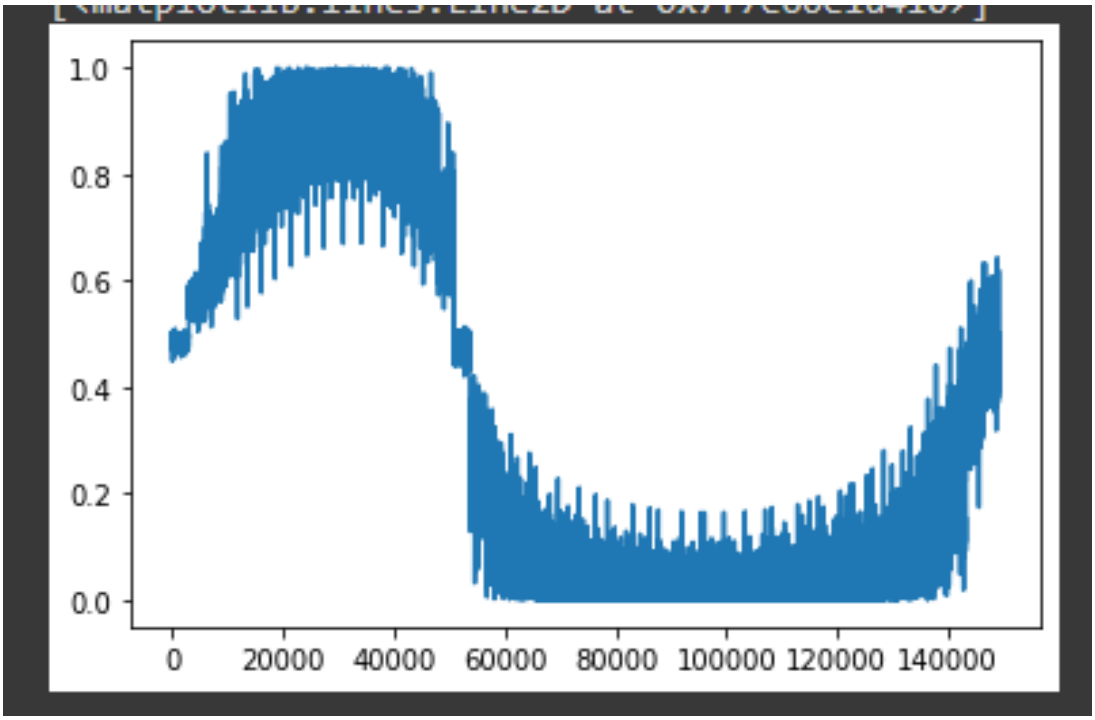

Fig.12 Example of Medium Importance Result

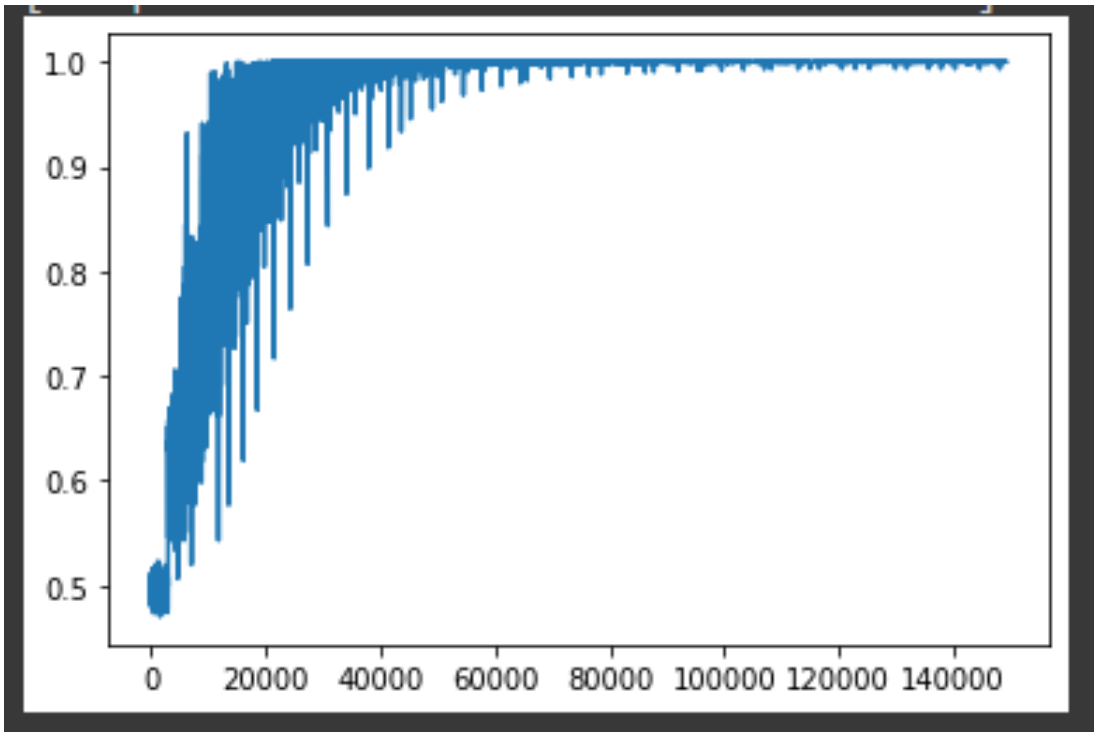

Fig.13 Example of Important Result

4.4 Summriazation Model

The summarization model we will be using in this project is an abstractive summarizer. The main goal is to compare the performance of the abstractive summarizer with and without applying fuzzy logic and neural networks. We decided to use the T5 model in transformers version 2.8.0 and torch 1.4.0. The model is pre-trained by Google using deep learning on an unlabeled big text corpus dubbed Colossal Clean Crawled Corpus [11]. It provides five per-trained paths and the difference between those paths is the number of parameters: T5-small contains 60 million, T5-base contains 220 million, T5-large contains 770 million, T5-3B contains 3 billion and T5-11B contains 11 billion. Because we are only running simulations and the dataset is vast, we chose the T5-small pre-trained route to save time.

4.5 ROUGE Score

The method we use to compare the result is the ROUGE score. It is one of the most common scoring systems used for text summarization models. There are four different measures: n-gram overlap compares (ROUGE-N), longest common subsequence (ROUGE-L), weighted longest common subsequence (ROUGE-W), and skip-bigram (ROUGE-S)[8]. The one we designed to use is ROUGE-1 because our dataset included abstractive and extractive summaries, the longest common subsequence may not be a good fit for abstractive summaries comparison.

$$ROUGE - N = \frac{Number\ of\ n-grams\ match\ in\ reference\ and\ output}{Number\ of\ n-grams\ in\ output}$$

$$F1 - Score = \frac{2*Precision*recall}{Percision + recall}$$

Fig.14 ROUGE-N and F1-Score Formulas

"N" in ROUGE-N is representing the length of overlap words between the reference summary and output summary (formulas shown in Fig.13). It will output three different scores: precision, recall, and F1-score. For example, we have two sentences here: "I am a student at San Jose State University" and "Tom is a student at the University of San Jose"; if we use ROUGE-1, overlapping words are "a", "student", "San", "University", and "Jose". The precision score will be around 0.56, the recall is 0.5 and the F1 score is around 0.53. Python has a library called Rouge-Score which provided functionality for ROUGE-N and ROUGE-L, we will use the library in our project to get the scores.

4.6 Web-based Application

To provide an application for users to use the model simply, we used Drupal to build a website with a backend server using Django. The idea is to have two servers: a front-end and a back-end. The front-end server will receive the text from the user by simply copy-and-paste to the text box or uploading a file, and sending a POST request to the backend server. The back-end server will receive the request and process it with our model then send responses back to the front end and display it to the user. Figure 16 is showing the architecture of the web application.

First of all, we used Drupal 9 with MySQL version 8.0.27 and phpMyAdmin version 5.1.1 to create a simple website on my local host. To be able to use JavaScript and CSS in Drupal, we added the Asset Injector module and made sure it preprocess my JS/CSS code. The JS code is used to send a request with user input and the CSS code is for styling.

For the backend server, we used Django to create a local server in another endpoint, it will receive the POST request from our Drupal and pass the text into our summarizer. The

Python version I used is 3.10 and the Django version is 4.0.4. To make sure we can receive requests from the front end. We will need to all origins and set up trusted origins in the setting.

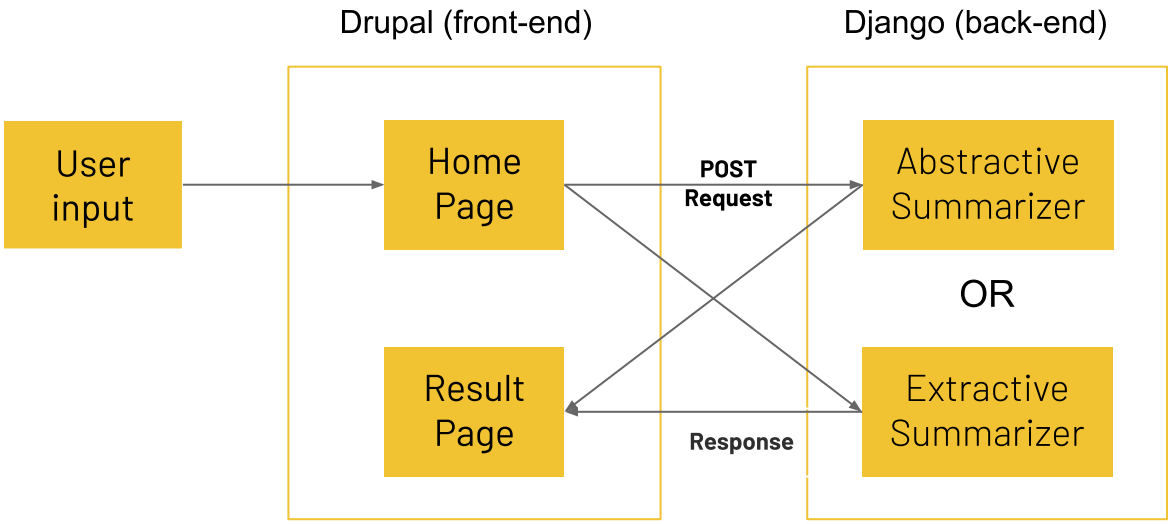

Fig.15 Architecture of Web-Based Application for Text Summarization

Figure 17 is showing the home and result pages of the website. The home page is where users can input text for summarization. Users can select between extractive and abstractive summaries and the percentage they would like to extract for extractive summarization. After inputting text or uploading a file, it will automatically jump to the result page. It contains a summary and original text, and users are able to back to the home page by using the back button.

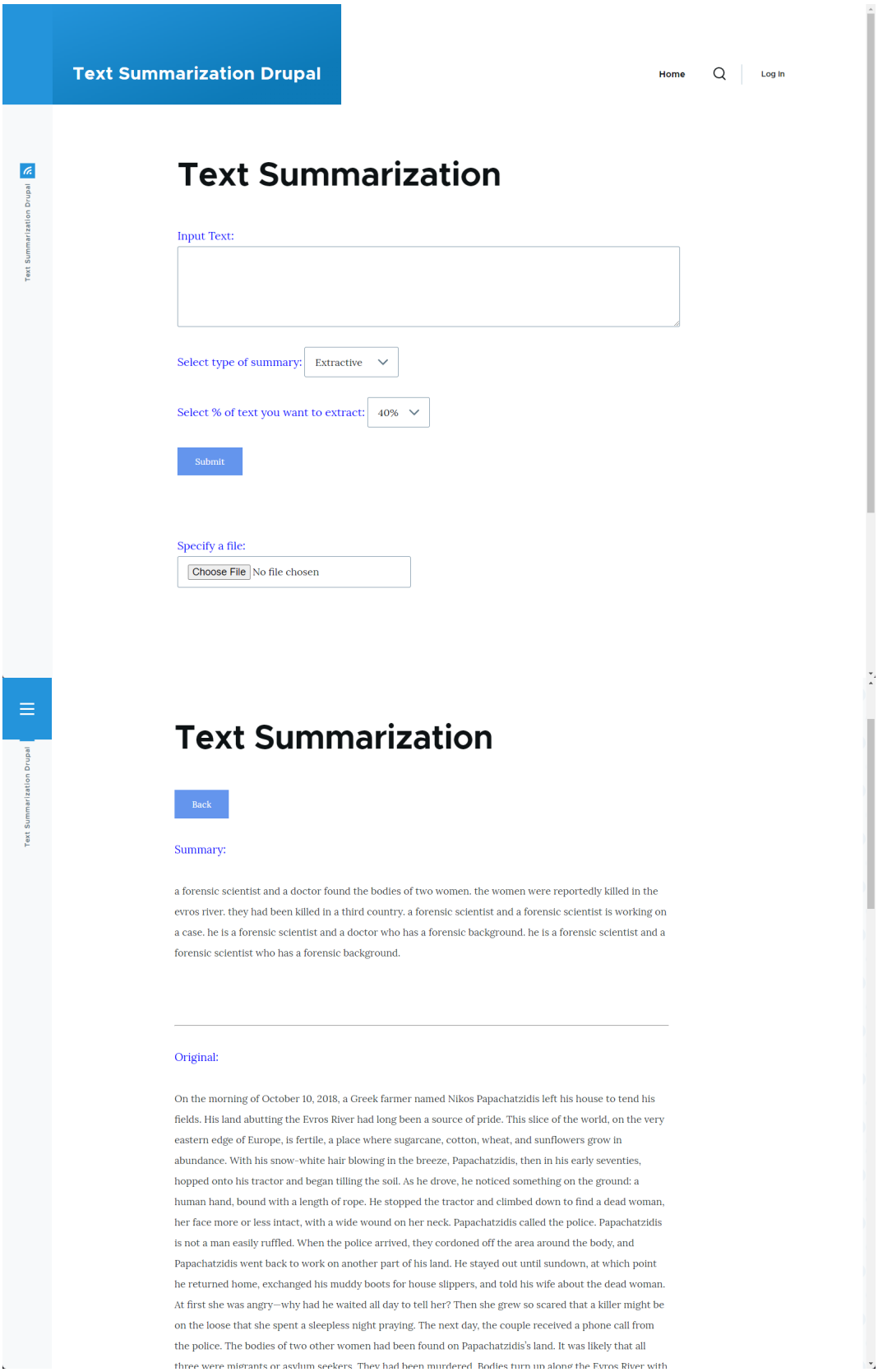

Fig.16 Sample Page of Web Application

5. EXPERIMENTS AND RESULTS

After pre-processing, we built the Neural Network model to determine the weights of each feature. We experimented six times to acquire an average result because the neural network models may produce data with minor gaps.

5.1 Experiment 1: BBC News Summary

In this experiment, we are utilizing the BBC News Summary dataset. It is unrelated to tweets, but we would like to have additional data to test the model and see whether the weights of features will vary in different data types. Table 1 displays the outcomes of all six studies. The result from one of the experiments is shown in Fig.18, where we can see that the initial value for each feature is close to 0.5. After training with all of the articles in the dataset, the values are close to zero or one. We determined that traits with a weight of zero are not important, but those with a weight close to one are. However, when we see a characteristic change significantly during the experiment, such as the third in the picture, we want to label it as "in-between" or medium values. From the data, we found the best-fit features are “Keyword/Title” and “Numerical Data”. Due to the nature of modern-day journalism, the title of articles often pinpoints key findings or important information regarding a subject. Numerical data found in articles also often emphasize the magnitude of events and also assists in providing key statistics and relevant information. My personal anecdotal observations further solidify the data found from our model and find that “Keyword/Title” and “Numerical Data” alone would commonly provide a sizeable amount of details regarding a specific article.

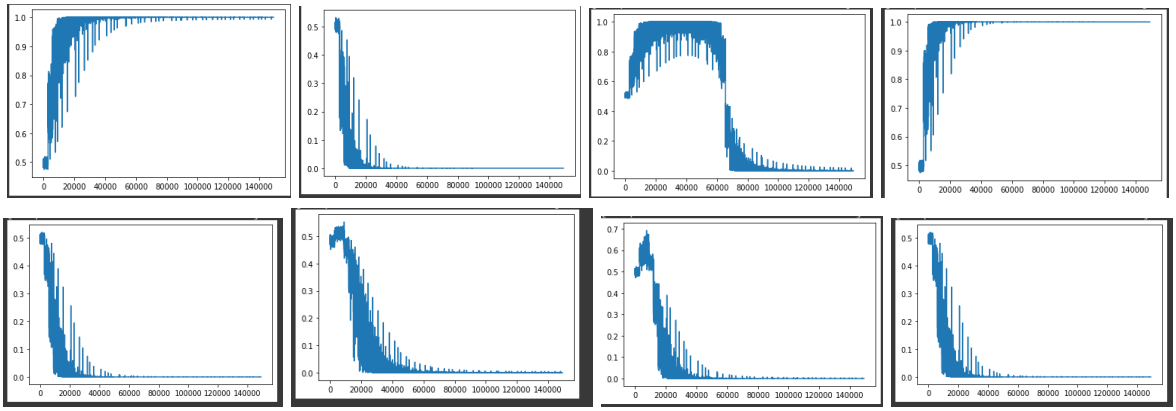

Fig.17. Snapshot of Neural Network Outcome

Table 1. Neural Network Result of BBC News Summary

	W1	W2	W3	W4	W5	W6	W7	W8
1	Important	Not Important	In between	Important	Not Important	Not Important	In between	Not Important
2	Important	Not Important	In between	Important	Not Important	Not Important	Not Important	Not Important
3	Important	Not Important	Not Important	Important	Not Important	Not Important	Important	Not Important
4	Important	Not Important	Important	Important	Not Important	Not Important	Important	Not Important
5	Important	Not Important	In between	Important	Not Important	Not Important	In between	Not Important
6	Important	Not Important	Not Important	Important	Not Important	Not Important	Important	Not Important
Result	Important	Not Important	Medium	Important	Not Important	Not Important	Medium	Not Important

Next, we created fuzzy logic rules based on the importance from above and extractive out sentences with high scores. We compared the summary created by the T5 model and the Fuzzy Neural Network T5 model with the summary provided by the dataset in Table 2. The highest score is the summaries created using the Fuzzy logic score with Neural Network which is an extractive summarization. As we mentioned before, extractive summaries are easier to achieve

since we are comparing words in common. By applying Fuzzy Logic with Neural Network, our F-measure score increased about 7.5%.

Table 2 ROUGE-N Score Comparison for BBC News Summary

Model	Average Precision	Average Recall	Average F-measure
T5	35.3%	75%	48%
T5 with Fuzzy Neural Network	40.9%	85.0%	55.5%
Fuzzy Neural Network (Extractive)	70.9%	90.8%	79.6%

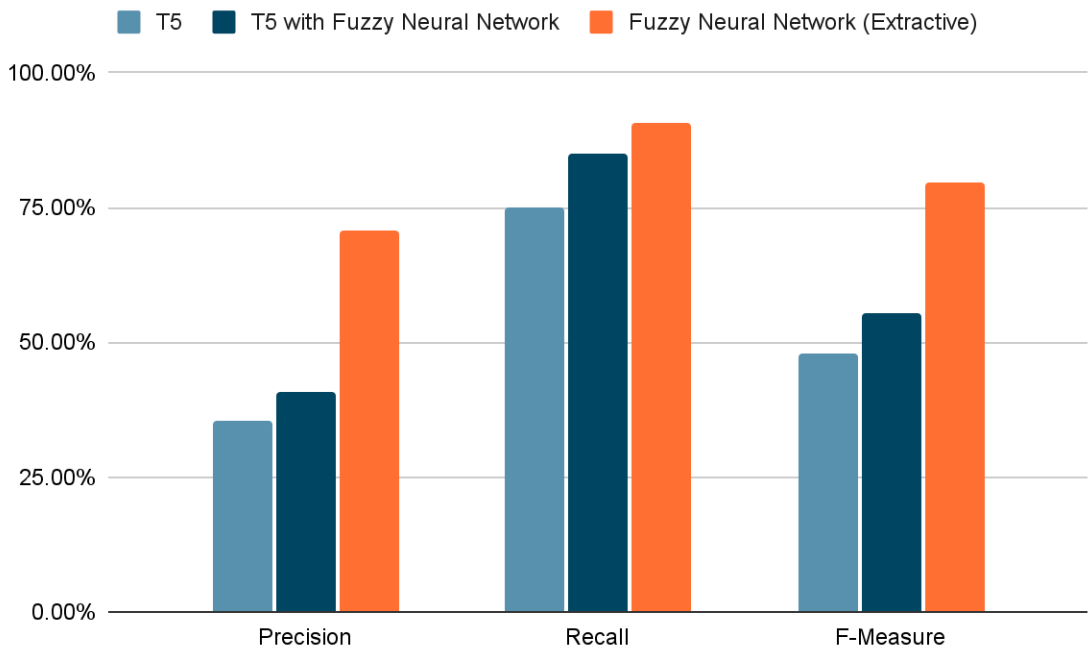

Fig.18 Graph of Experiment One Result

5.2 Experiment 2: TweetSum

To check the performance of our model when processing tweets, we experimented using the TweetSum dataset. This dataset does not have headlines or titles for each dialog, so we only have 7 features. Figure 19 shows a snapshot of one of the results from the neural network, we also ran six times for this experiment, and the output results are shown in Table 3.

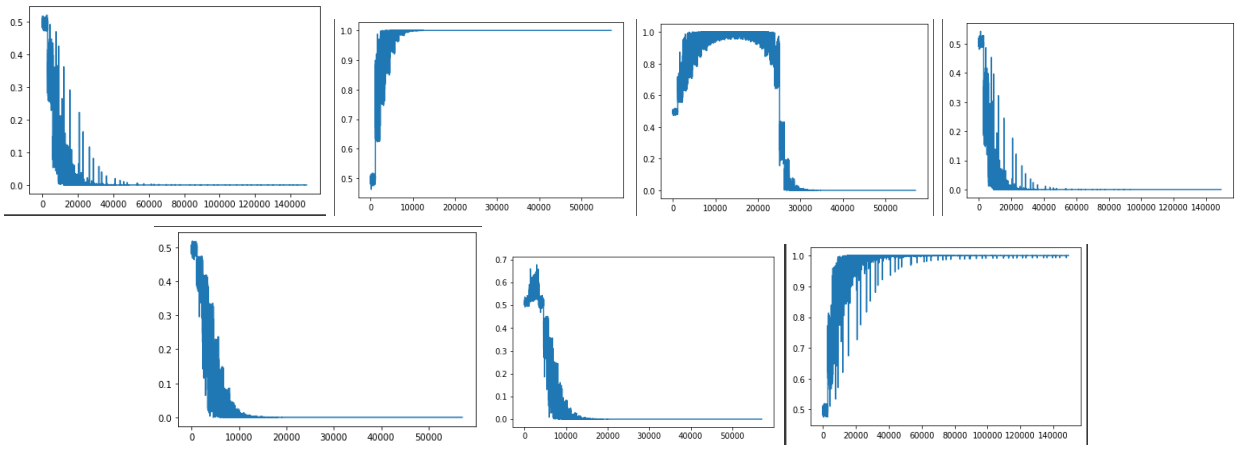

Fig.19. Snapshot of the Neural Network Outcome of TweetSum

Table 3. Neural Network Result of TweetSum Dataset

	W1	W2	W3	W4	W5	W6	W7	W8
1	\	Not Important	Not Important	Important	Not Important	Not Important	In between	Not Important
2	\	Not Important	In between	Important	Not Important	Not Important	In between	Not Important
3	\	Not Important	Important	Important	Not Important	Not Important	In between	Not Important
4	\	Not Important	Important	Important	Not Important	Not Important	Important	Not Important
5	\	Not Important	Important	Not Important	Not Important	Not Important	Important	Not Important
6	\	Not Important	Important	Important	Not Important	Not Important	Important	Not Important
Result	\	Not Important	Important	Important	Not Important	Not Important	Important	Not Important

The most essential characteristics as a consequence are "Proper Nouns," "Numerical Data," and "Location." Nouns and numerals are quite prevalent in dialogues, and the major issue will generally appear at the start of a conversation. This dataset contains customer service dialogs from Twitter, and because people seldom repeat themselves in a short period, phrases and sentences have a decreased likelihood of being similar to others.

Table 4 ROUGE-N Score Comparison for TweetSum

Model	Target Summary Type	Average Precision	Average Recall	Average F-measure
T5	Abstractive	40.6%	30.2%	34.7%
	Extractive	28.6%	41.9%	33.9%
T5 with Fuzzy Neural Network	Abstractive	50%	27.1%	35.1%
	Extractive	42.3%	47.8%	43.1%
Fuzzy Neural Network (Extractive)	Abstractive	52.9%	14.1%	21.4%
	Extractive	73.8%	43.4%	52.8%

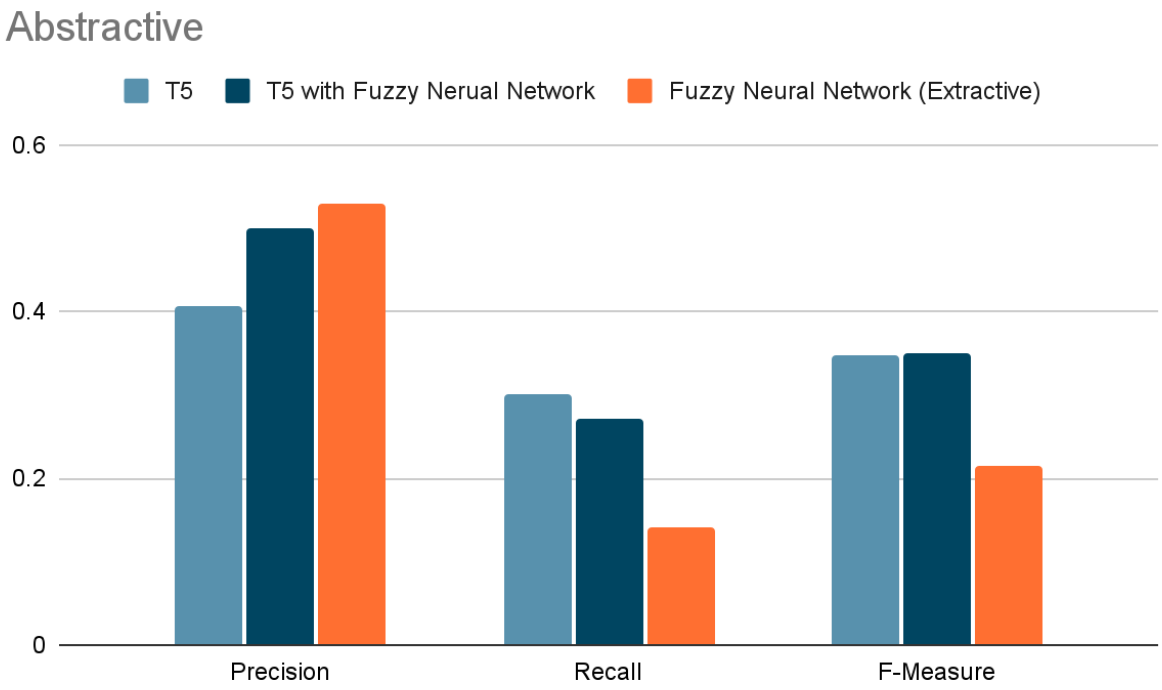

Fig.20. Graph of Experiment Two Result Compare to Abstractive Summary

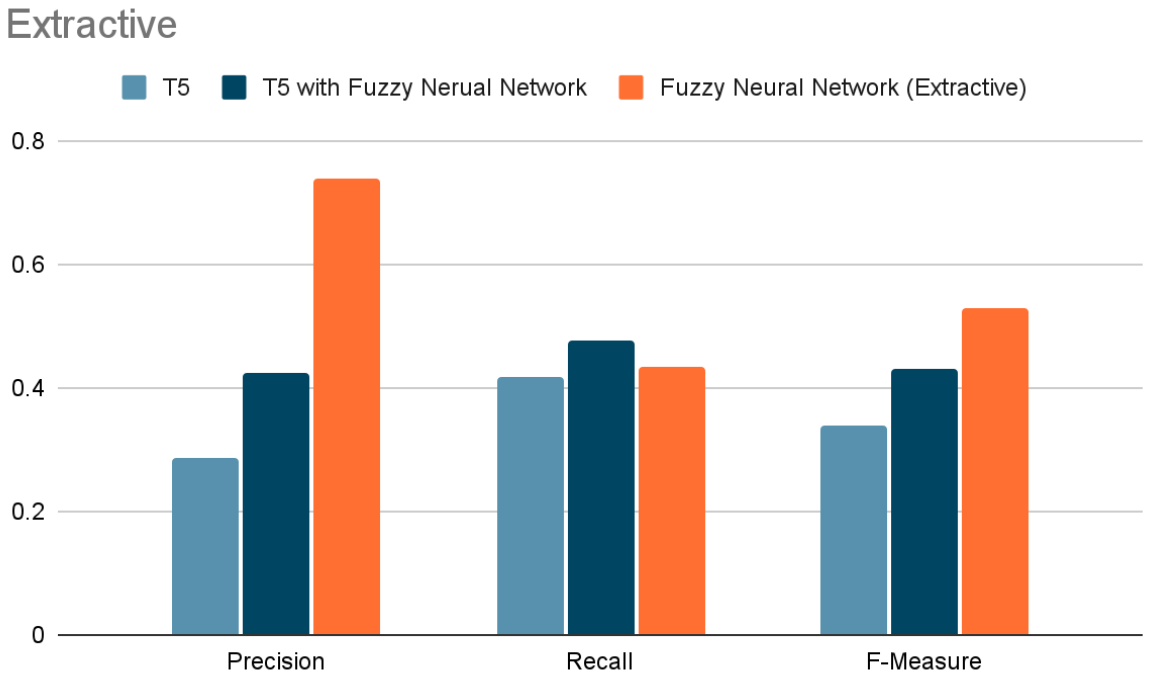

Fig.21. Graph of Experiment Two Result Compare to Extractive Summary

Because the TweetSum dataset contains both abstractive and extractive summaries, we can compare our results to both types. The T5 model received 34.7%, while the abstractive summary received 33.9%. It is also the only model that received a better score in abstractive summary than extractive. The T5 model was created and trained specifically for abstractive summarization so it performs better in abstractive summarization. The extractive model, which we created using Fuzzy Logic and Neural Network, got 52% when compared to an extractive summary, which is the highest score we have on tweets data, but its abstractive summary score is the worse in all models. It is reasonable since the model was not built as abstract. T5 with the Fuzzy Neural Network model improved by 9.2% compared to the pure T5 model in extractive comparison and improved by 0.4% in abstractive. For the abstractive comparison, we improved way less than the previous dataset and also other scores for all models. Tweets data is hard to summarize when it is just a short dialog or a few tweets because text summarizers are doing better in handling larger text.

6. CONCLUSION AND FUTURE WORK

In this project, we discussed different types and methods of Text Summarization. To determine the importance of sentences in an article, eight different features, fuzzy logic, and neural network was applied to the abstractive model. We utilize Neural Network to determine the relevance of each feature in the text to better comprehend and analyze the weight of different features in various kinds of data. Then we compared and analyzed the model when dealing with tweets and articles. We also built a web-based application for users to simply summarize the content using Drupal and Django.

In the future, we can generate or find a more extensive tweet or another social media dataset to improve the training. Currently, very few larger tweet datasets are including a human-generated summary since it is hard for people to read hundreds and thousands of tweets and come up with a summary. Another part we can improve will be including more features in our experiment to consider more possibilities. By using the neural network model, we can identify the more important features from a large set of features.

On the other hand, social media platforms such as Twitter have users from all over the world, so applying multilingual and cross-lingual approaches to the model would be beneficial. There are numerous non-English messages on Twitter, and our model is currently unable to handle tweets in other languages. Many pieces of information are unable to be processed, and as a result, many distinct views are lost.

Last but not least, we may enhance the web-based application with additional functionality such as creating accounts, recording the summarization history, selecting alternative models, and topic-based summarization.

REFERENCES

[1] El-Kassas, W. S., Salama, C. R., Rafea, A. A., & Mohamed, H. K. (2021). Automatic text summarization: A comprehensive survey. *Expert Systems with Applications*, *165*, 113679. https://doi.org/10.1016/j.eswa.2020.113679

[2] Wang, W. M., Li, Z., Wang, J. W., & Zheng, Z. H. (2017). How far we can go with extractive text summarization? heuristic methods to obtain near Upper Bounds. *Expert Systems with Applications*, *90*, 439–463. https://doi.org/10.1016/j.eswa.2017.08.040

[3] Kouris, P., Alexandridis, G., & Stafylopatis, A. (2019). Abstractive text summarization based on deep learning and semantic content generalization. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. https://doi.org/10.18653/v1/p19-1501

[4] Etemad, A. G., Abidi, A. I., & Chhabra, M. (2021). A review on abstractive text summarization using Deep Learning. *2021 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*. https://doi.org/10.1109/icrito51393.2021.9596500

[5] Widyassari, A. P., Rustad, S., Shidik, G. F., Noersasongko, E., Syukur, A., Affandy, A., & Setiadi, D. R. (2022). Review of Automatic Text Summarization Techniques & Methods. *Journal of King Saud University - Computer and Information Sciences*, *34*(4), 1029–1046. https://doi.org/10.1016/j.jksuci.2020.05.006

[6] Suanmali, L., Salim, N., & Binwahlan, M. S. (2009, June 1). *Fuzzy logic based method for improving text summarization*. Universiti Teknologi Malaysia Institutional Repository. Retrieved November 26, 2022, from http://eprints.utm.my/id/eprint/18963/

[7] Kyoomarsi, F., Khosravi, H., Eslami, E., Dehkordy, P. K., & Tajoddin, A. (2008). Optimizing text summarization based on Fuzzy Logic. *Seventh IEEE/ACIS International Conference on Computer and Information Science (Icis 2008)*. https://doi.org/10.1109/icis.2008.46

[8] Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics

[9] Shinde, R.D., Routela, S.H., Jadhav, S.S., & Sagare, S.R. (2014). Enforcing Text Summarization using Fuzzy Logic.

[10] *Fuzzygaussian*. FuzzyGaussian-ArcGIS Pro | Documentation. (n.d.). Retrieved November 28, 2022, from https://pro.arcgis.com/en/pro-app/latest/arcpy/spatial-analyst/fuzzygaussian-class.htm

[11] *Abstractive summarization using Google's T5*. Turbolab Technologies. (2021, November 12). Retrieved December 4, 2022, from https://turbolab.in/abstractive-summarization-using-googles-t5/