International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys.13, No.1 (Apr-23)

http://dx.doi.org/10.12785/ijcds/130181

A Statistical and Machine Learning Approach for Summarising Computer Science Research Papers

Sheik Muhammad Wakeel Bauboorally[1] **and Sameerchand Pudaruth**[1]

[1]*Department of Information and Communication Technologies, University of Mauritius, Mauritius*

Received 12 Jul. 2022, Revised 30 Dec. 2022, Accepted 6 Feb. 2023, Published 16 Apr. 2023

Abstract: Academics, researchers and students usually read a lot of papers for their research or to keep up-to-date with the latest works. The high number of papers available makes the process time-consuming. A solution is to summarise the papers and allow the reader to decide if the papers are relevant to their work and whether they require more attention. A system has been built to generate extractive summaries of computer science research papers. We demonstrate how the intrinsic statistical characteristics of computer science research papers such as the document length or the presence of certain keywords can help train a machine learning classifier model that can achieve state-of-the-art performance. Human and automatic evaluation using ROUGE has been carried out to measure performance. Results show that the proposed model performs better than TextRank and BERT on both human and automatic evaluation. It also does better than BART on human evaluation.

Keywords: research paper, computer science, extractive summaries, statistical, machine learning

1. Introduction

The number of academic publications worldwide has increased from 0.66 million to 2.85 million in 2015, and further increased to 3.16 million in 2018 [1]. White et al. (2017) found that Computer Science accounts for nearly 9% of all research work [2]. The high number of papers, therefore, makes the research process highly time-consuming. Many of these papers are often irrelevant to the reader. A solution to these problems is to summarise the papers and allow the reader to decide if they are relevant to their work and whether they require more attention. A system has been built to generate summaries of computer science research papers. The system harnesses the power of Natural Language Processing (NLP) and text summarisation.

There are 2 types of text summarisation namely, extractive and abstractive. Extractive text summarisation picks the most important sentences of the document and does not modify the sentence structure of the original text. Abstractive text summarisation generates new sentences based upon the main ideas of the original text. This work focuses on extractive text summarisation only. Text summarisation can also be single document or multi-document. For single document, the summariser takes just one document as input and generates a summary. Multi-document summarisers take many documents of the same topic as input to generate a single summary. This study is limited to single document summarisation only.

Many websites exist for text summarisation but most of them are generic and do not take into account the specific properties of scientific papers and especially computer science research papers. These types of publications usually have a predetermined structure, containing tables, figures, pseudocode along with domain-specific keywords and concepts [3]. Therefore, a summariser trained on computer science papers could improve the output summary as it learns the fundamental characteristics of this type of literature. The summariser should be able to retrieve the most important and relevant information from the document and express those key ideas in the final summary while optimising topic coverage and readability.

This study is domain-specific and is concerned with computer science research papers. With this in mind, it has been observed that certain statistical features such as length, absence or presence of foreign words, digits count, symbols count, and the appearance of certain keywords in a sentence can be used to predict whether the sentence is likely to be part of the final summary. The problem, therefore, boils down to a binary classification problem. As of 2021, only a few summarisers have been developed, or models trained, specifically for computer science papers. Thus, we propose a machine learning approach that uses the intrinsic statistical properties of computer science research papers to predict whether or not a sentence belongs to the final summary. This approach is implemented using different classification algorithms such as Logistic Regression, K-

E-mail address: wbauboorally@gmail.com, s.pudaruth@uom.ac.mu

Nearest Neighbors, Support Vector Classifiers (SVC Linear and SVC RBF - SVC RBF uses the Radial Basis Function kernel), Decision Tree, and Random Forest. The best performing model is compared to existing text summarisation techniques such as TextRank [4], BERT [5], and BART [6]. Human evaluators and the ROUGE (Recall-Oriented Understudy for Gisting Evaluation) score [7] have been used to evaluate the different models.

ROUGE is a set of metrics that can enable us to measure the accuracy of the summary compared to a base/reference summary. ROUGE-N determines the number of matching 'n-grams' between our generated summary and the reference summary. An n-gram refers to a group of words/tokens. ROUGE-1 for instance measures the number of similar unigrams(1-gram) between two texts. ROUGE-2 uses bigrams. ROUGE-L measures the longest matching sequence of words in the two texts. Those words need not be in consecutive order. ROUGE allows us to calculate the recall, precision, and f1 score. For recall, the matching n-gram is divided by the total n-gram in the reference summary. For precision, the matching n-gram is divided by the total n-gram in the generated summary. The f1-score is a combination of recall and precision.

3686 papers from the arXiv database have been used to train and test the classification models. The arXiv database contains 1.7 million scientific articles, along with related features such as the title of the article, the authors, the abstract, and the article id which can be used to download the papers' PDFs. The articles engage many different fields. Since we are interested mainly in computer science research papers, the dataset has been filtered to include only computer science articles. The papers have been preprocessed, analysed, and split into sentence datasets which are used to train the models to predict the probability of a sentence being a summary or non-summary sentence. The best summary sentences are then picked for the final summary.

This paper proceeds as follows. Section 2 goes through different works related to text summarisation. Section 3 consists of the methodology. Section 4 contains the results obtained from the different experiments. The paper concludes with Section 5.

2. Related Works

There are different approaches to text summarisation. In this section, we review the statistical methods, graph-based methods, machine learning approaches, and deep learning methods.

Works on automatic text summarisation began in the 1950s when Baxendale (1958) introduced the Positional method [8]. Upon analysing 200 paragraphs from scientific documents, he found out that 85% of topic sentences were the first sentence of the paragraph and 7% were the last sentence. He could obtain a summary by extracting the first and last sentence of the document. Luhn (1958) came up with a method to extract important sentences from a text using word and phrase frequency [9]. The weight of a sentence was given as a function of high-frequency words, disregarding very high-frequency common ones. Fattah & Ren (2009) argue that the most important sentences in a paragraph are the first 5 sentences [10]. Ishikawa et al. (2010) gave more weight to words that are present in the title of the text because the title of the document generally gives a good indication about the topic [11][12].

In graph-based methods, sentences from one or multiple documents are represented as vertices. The edges between the sentences show how similar the two sentences are. LexRank [13] and TextRank [4] are graph-based methods. The idea is to create a similarity matrix by finding the similarity score between each sentence in the document or collection of documents. Each sentence is represented as a node in a graph. Nodes are connected based on the similarity matrix. A sentence connected to many other sentences has a high probability of belonging to the final summary. It uses the bag-of-words model to represent sentences as vectors of N-dimension (numbers instead of words). Only then the similarity between sentences can be computed.

Kupiec et al. (1995) introduced the first trainable method for text summarisation [14]. They used classification, a supervised machine learning method, to identify sentences as summary sentences and non-summary sentences. They created a dataset to train the model, consisting of documents and their respective extractive summaries. They used a Naive Bayes classifier to find the probability of a sentence appearing in the final summary given a set of features. However, Osborne (2002) found that maximum entropy models have better performance over Naive Bayes approaches [15]. Machine learning can also be used to estimate the weights of features. Yeh et al. (2005) proposed an algorithm to estimate the weights of certain features such as sentence position, centroid, keywords, and title similarity [16]. This makes the system dependent on the genre of the document. Yatsko et al. (2010) trained a system on three genres, namely scientific, news and arts [17]. The system can detect the genre of the document before using the proper scoring model. Radev et al. (2000) introduced a centroid-based method for single and multi-document summarisation [18]. Sentences that are similar belong to a cluster. For each cluster, the sentence closest to the centroid is selected for the final summary. Ghodratnama et al. (2020) combined supervised and unsupervised algorithms for extractive document summarisation [19]. Their model gives higher ROUGE scores than most state-of-the-art models. They combined classification and clustering algorithms to select sentences that are logical and non-redundant. Haez Shamsfakhr (2022) represent extractive text summarisation as a Bayesian state estimation problem [20]. They rated the importance of sentences using a sequential Markov model equipped with Bayesian inference

BERT was released by Google AI Language and was a breakthrough in the field of Natural Language Processing

[5]. BERT is trained on an extremely huge dataset of texts. It uses a Transformer, which is an attention mechanism, to understand the meaning and relationships between words in a text. Generally, a Transformer is made up of two parts, namely the encoder and decoder. BERT consists only of the encoder part because its aim is to generate a language model. Instead of reading a text from left to right or right to left, BERT reads the entire sequence of words at once. This is why it is called bidirectional. This is the key to learning context in a text as the model has a better overview of each word's surroundings. Liu and Lapata (2019) proposed a framework on top of BERT that can be applied to both extractive and abstractive text summarisation [21]. They called it BERTSum.

BART was introduced by Facebook AI [6]. It combines BERT (encoder) and OpenAI's Generative Pre-trained Transformer [22], which is also known as GPT (decoder). BART merges BERT's ability to understand the context in texts with GPT's capacity to generate texts. BART achieves state-of-the-art results in text summarisation. It can be considered one of the best text summarisation algorithms right now.

Many works have targeted specific domains for text summarisation. LetSumm is a summarisation system for juridic documents [23]. Reeve et al. (2007) created a summariser for biomedical texts [24]. They used two techniques, one to locate important sentences and the other to discard redundant information. Gao et al. (2019) released the readeraware summary generator (RASG) which produces abstractive summaries of social media documents with the help of reader comments [25]. Their model achieved state-of-the-art performance. Erera et al. (2019) presented a system to summarise computer science papers using the IBM Science Summarizer [26]. 270,000 computer science papers were retrieved from arXiv.org and the ACL anthology. The system allows users to interact with it using natural language queries or filters (location, year, author, datasets, etc.). The summarizer generates a summary for each section of the paper (section-based summarisation). Hartl & Kruschwitz (2021) approached fake news detection by employing text summarisation as the main text transformation strategy prior to document classification [27]. They did not have much time to investigate several other experimental setups but the results suggest a positive direction for future work using their approach. Abdullah (2022) proposed a semi-extractive text summarisation system to extract the research objectives from academic literature [28]. Experiments were carried out using research papers from IEEE and Elsevier Scopus. The method shows promise when it comes to extracting objective sentences from the abstract section.

3. Methodology

A. Dataset Building

Figure 1 shows the steps to process the papers obtained from the arXiv database. After downloading the dataset, only computer science papers are kept. Using the papers' ids provided, the PDFs of the papers are downloaded. Using a package called Science Parse, text information is extracted from the PDFs, they are grouped by section and saved in JSON format. 3686 papers are picked for further processing. For each section (except for the Abstract and References), the texts are appended to one another to form the source text. The source text is concatenated with the paper record in the CS dataset. Unnecessary columns are removed. Papers that have no defined introduction and conclusion are also removed. This last extra step is done because it often means the information was not properly extracted from the PDFs.

B. Dataset Splitting and Filtering

The dataset is split into training (80%) and testing (20%) datasets. Several filters are applied to them. It was found that review papers and foreign language heavy papers are poorly summarised. Review papers are difficult to summarise using an extractive approach because they usually discuss several other papers and not a single work. Often each new paragraph discusses a different work from the previous one. Furthermore, since the models are trained mainly on papers written in English, the foreign language heavy papers are not summarised well.

Therefore, it was decided to include filters to remove those papers from the dataset (Review Paper Filter, Foreign Paper Filter) and analyse how the results differ from the dataset with no filter. Another filter used is the Long/Short Abstract Filter. This filter removes papers with too short or too long abstracts. When there is a big difference in the length of the generated summary and the abstract (reference summary), the evaluation score is negatively impacted. After applying different filters, 5 new datasets are created: no filter dataset, review paper filtered dataset, foreign paper filtered dataset, review + foreign paper filtered dataset, and long/short abstract filtered dataset.

A dataset of source texts is not ideal for the task at hand. The aim is to classify each sentence in a paper as a 'summary' sentence or a 'non-summary' sentence. Hence, a sentence dataset is required. Bearing in mind that the abstracts are used as the reference summary, the abstract of each paper is split into individual sentences. These sentences will be marked as 'summary' sentences. Then, for each sentence in the source text, it was decided to compute their similarity with the title of the paper. The sentences are then sorted in descending order of similarity and 6 sentences (to make a balanced dataset) from the 70th percentile are chosen and marked as 'non-summary' sentences. The 70th percentile was chosen as it is below the average sentence similarity score but not extremely different from the other sentences in general (as would be the case for the worst 6 sentences). Finally, the abstract sentences are concatenated with the non-summary sentences to form the sentence dataset. The same steps are applied to the testing datasets.

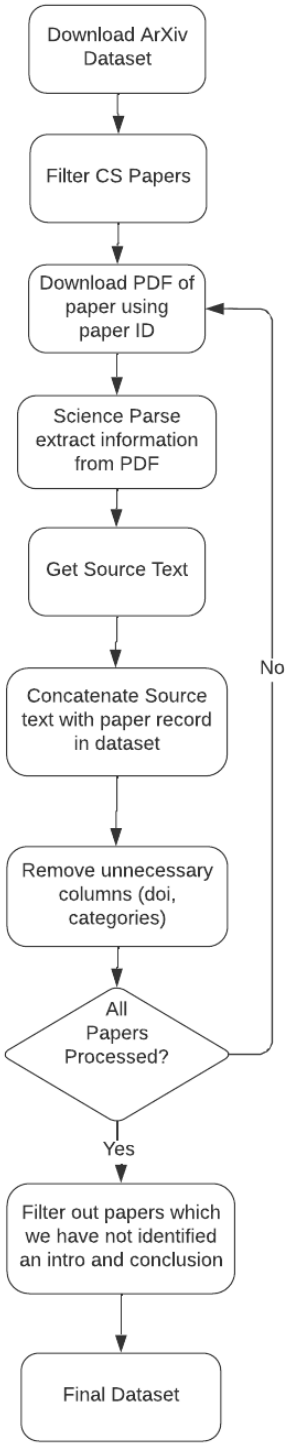

Figure 1. Dataset Building

C. Data Preprocessing

Before proceeding to feature extraction, the source text needs to be cleaned. It is converted to lowercase to make it easier to compare words later (e.g 'program' should be interpreted the same as 'Program'). Unwanted characters (#, *, ~, etc.), URLs, and inline citations are removed.

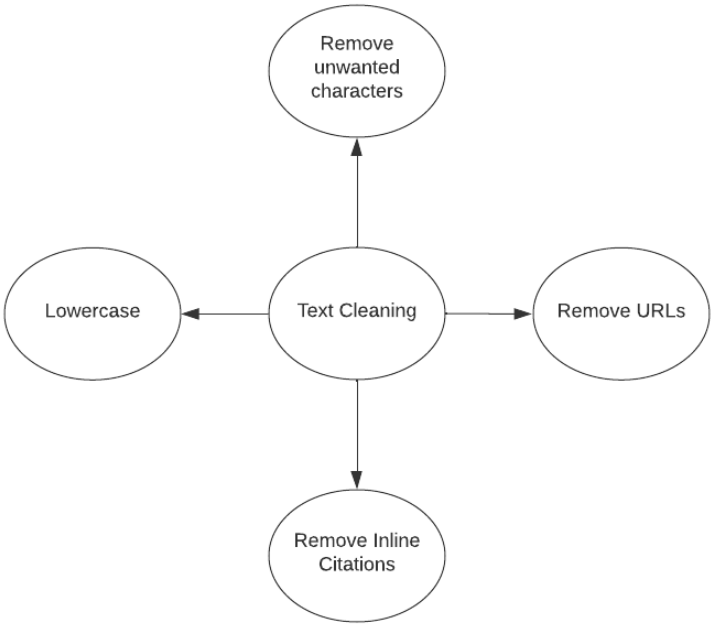

Figure 2. Data Preprocessing

D. Feature Extraction

Feature extraction is done on the sentence dataset. For each sentence, relevant features are extracted such as length of sentence, total foreign words present, digits and symbols count, and the presence of keywords. Keywords are those words that appear often in summary sentences. After analysing the summary sentences, several of them were found (e.g 'paper', 'data', 'algorithm', etc.). In addition to that, the presence of discourse markers is also determined. Discourse markers are those words that are used to connect sentences. Examples of discourse markers are 'moreover', 'furthermore', and 'in addition' among others. Those sentences containing discourse markers are less likely to be part of the final summary. Symbols count is done on the original text before text cleaning. The feature extraction steps are shown in Figure 3.

E. Classification and Evaluation

Once the features are extracted from each sentence in the different training datasets (with different filters), different machine learning classification algorithms mentioned in Section 1 are trained. Using each trained model, summaries are then generated for every paper in the testing

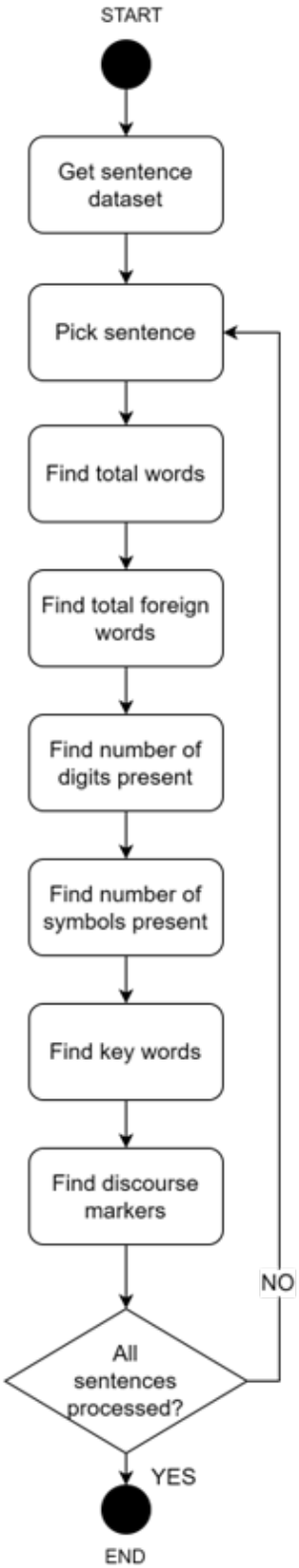

Figure 3. Feature Extraction

datasets (with the different filters mentioned previously). The summaries are compared with the abstract of their respective paper and the ROUGE scores are calculated using the ROUGE python package. Finally, the average ROUGE score is determined.

4. Results and Discussion

A. *Automatic Evaluation*

Experiments were carried out on each new dataset created after the different filters were applied. ROUGE-1, ROUGE-2 and ROUGE-L were used for evaluation. Generated summaries are compared to the abstract of the papers. Logistic Regression and SVC Linear do well on precision and overall f-score. Random forest obtains the best score on recall all the time but does poorly on precision, indicating that the sentences picked by the algorithm often contain irrelevant information.

Logistic regression and SVC Linear have the highest precision scores. K-Nearest Neighbors receive the highest recall scores but does worse than Logistic Regression, SVC Linear, and Decision Tree.

Logistic Regression and SVC Linear are again the best models on precision and f-score. SVC RBF does poorly on recall and overall f-score.

From the results, Logistic Regression and SVC Linear are the best models based on the F-scores. The best ROUGE-1 score was 0.335 obtained by SVC Linear, closely followed by Logistic Regression's 0.334 f-score. Both models do equally well on ROUGE-2 and ROUGE-L where they both obtained an f-score of 0.113 and 0.303 respectively. Logistic Regression does slightly better on Recall than SVC Linear. SVC Linear does marginally better on precision than Logistic Regression. The f-scores obtained by Logistic Regression are very close to that of SVC Linear in most cases. The high precision and recall of those classifiers mean they generate summaries that have many matching and relevant words when compared to the reference summaries.

Random Forest gives the best recall in all cases for ROUGE-1 and ROUGE-L but does poorly on precision. This means Random Forest retrieves many matching words to the reference summary but also contains many irrelevant and unnecessary ones. SVC RBF looks like the worst model as it does poorly on recall, precision, and f-score. It does poorly in retrieving sentences that have relevant matching words to the reference summary. K Nearest Neighbours gives the best Rouge-2 recall scores. This means it can retrieve matching bigrams better than the other models.

The 'long/short abstract' filter gives the best results. This is mainly because the generated summaries were being compared with an abstract of roughly the same length. Too short abstracts give a low ROUGE score as the calculated precision will be lower. This is because many words from the generated summary will be deemed irrelevant. Similarly, a long abstract much larger than the generated summary gives a low recall value. This is because it is less likely for

ROUGE-1

TABLE I. ROUGE-1 Scores

Filter	Metric	Logistic Regression	K Nearest Neighbors	SVC Linear	SVC RBF	Decision Tree	Random Forest
No Filter	Recall	0.264	0.284	0.261	0.237	0.261	0.300
	Precision	0.418	0.334	0.420	0.325	0.380	0.297
	F-score	0.313	0.291	0.312	0.265	0.297	0.280
Foreign Papers	Recall	0.263	0.287	0.261	0.243	0.263	0.305
	Precision	0.420	0.334	0.422	0.333	0.377	0.287
	F-score	0.314	0.293	0.313	0.272	0.298	0.276
Review Papers	Recall	0.264	0.285	0.261	0.236	0.264	0.301
	Precision	0.418	0.335	0.420	0.324	0.376	0.303
	F-score	0.313	0.291	0.312	0.264	0.298	0.283
Foreign + Review	Recall	0.263	0.287	0.261	0.239	0.263	0.299
	Precision	0.420	0.336	0.421	0.330	0.379	0.294
	F-score	0.314	0.294	0.313	0.269	0.299	0.279
Long/Short Abstract	Recall	0.292	0.308	0.296	0.268	0.288	0.319
	Precision	0.398	0.330	0.398	0.276	0.348	0.296
	F-score	0.334	0.312	0.335	0.268	0.311	0.299

ROUGE-2

TABLE II. ROUGE-2 Scores

Filter	Metric	Logistic Regression	K Nearest Neighbors	SVC Linear	SVC RBF	Decision Tree	Random Forest
No Filter	Recall	0.084	0.085	0.083	0.055	0.083	0.084
	Precision	0.145	0.102	0.146	0.080	0.128	0.083
	F-score	0.102	0.087	0.102	0.063	0.097	0.077
Foreign Papers	Recall	0.083	0.086	0.083	0.058	0.086	0.083
	Precision	0.146	0.101	0.146	0.085	0.129	0.079
	F-score	0.102	0.087	0.101	0.066	0.098	0.074
Review Papers	Recall	0.084	0.085	0.083	0.055	0.084	0.084
	Precision	0.145	0.103	0.146	0.080	0.128	0.088
	F-score	0.102	0.088	0.101	0.062	0.097	0.080
Foreign + Review	Recall	0.083	0.086	0.083	0.057	0.085	0.084
	Precision	0.145	0.102	0.146	0.084	0.129	0.083
	F-score	0.101	0.088	0.101	0.065	0.098	0.078
Long/Short Abstract	Recall	0.097	0.100	0.098	0.065	0.096	0.094
	Precision	0.139	0.108	0.140	0.068	0.117	0.087
	F-score	0.113	0.101	0.113	0.065	0.103	0.088

the generated summary to retrieve most words occurring in the reference summary.

It can be observed that the effect of the Foreign and Review papers filters was negligible. There is very little difference in their scores compared to those without a filter.

B. Comparison with Baseline Models

The best model obtained from the experiments in Section 4-A, which will be called 'custom ML model', is used for comparison with the baseline models TextRank (Gensim implementation), BERT, LSA, and BART. For comparison, we also use a fine-tuned deep learning model that uses the existing Doc2Vec sentence embedding model but is further trained on the computer science dataset. To generate summaries using the fine-tuned Doc2Vec model, the sentences in the text are grouped into clusters. For each cluster, the sentence closest to the centroid is picked.

One hundred computer science research papers were used for evaluation. Those papers were not part of the training and testing datasets mentioned previously.

TABLE IV. Average Summary Length

Model	Average Length (words)
TextRank	154
BERT	276
BART	194
Fine-Tuned Doc2Vec	358
Custom ML Model	252

TextRank produces the shortest summaries while the fine-tuned Doc2Vec model generates the longest one. Since on average only about 10 sentences are picked for the

ROUGE-L

TABLE III. ROUGE-L Scores

Filter	Metric	Logistic Regression	K Nearest Neighbors	SVC Linear	SVC RBF	Decision Tree	Random Forest
No Filter	Recall	0.240	0.259	0.237	0.214	0.237	0.274
	Precision	0.379	0.303	0.381	0.292	0.344	0.270
	F-score	0.285	0.265	0.283	0.238	0.270	0.255
Foreign Papers	Recall	0.239	0.262	0.237	0.218	0.240	0.278
	Precision	0.381	0.303	0.382	0.299	0.344	0.261
	F-score	0.285	0.266	0.284	0.244	0.272	0.251
Review Papers	Recall	0.240	0.260	0.237	0.213	0.240	0.274
	Precision	0.379	0.304	0.381	0.291	0.341	0.275
	F-score	0.285	0.265	0.283	0.238	0.271	0.258
Foreign + Review	Recall	0.239	0.262	0.237	0.215	0.240	0.272
	Precision	0.381	0.305	0.381	0.297	0.345	0.267
	F-score	0.285	0.267	0.284	0.242	0.273	0.254
Long/Short Abstract	Recall	0.266	0.281	0.268	0.243	0.261	0.290
	Precision	0.361	0.301	0.359	0.250	0.315	0.268
	F-score	0.303	0.284	0.303	0.243	0.282	0.272

summaries, we can conclude that BERT, the custom ML model, and the fine-tuned Doc2Vec model are more inclined toward picking longer sentences. The length of the generated summaries varies a lot. BART generates summaries of length greater than 250 words in some cases and less than 100 words in others. The length of the reference summary (abstract) also varies a lot, despite the average length being 183 words.

human evaluation that BART gives the best recall and f-score. BART can retrieve about 40% of the words present in the reference summary (ROUGE-1). BART is currently one of the best extractive text summariser. Nevertheless, our custom ML model does better on precision for ROUGE-1 and ROUGE-L. The higher precision obtained for the custom ML model is probably due to the fact that it was trained specifically on a computer science dataset.

Our model does better on recall than BERT. While BERT retrieves 22% of the words on average, the custom ML model retrieves about 27%. There is little difference in the recall scores obtained from TextRank and the custom ML model. But, the custom ML model does much better on precision. This indicates that our model can retrieve more relevant sentences from the source text. The fine-tuned Doc2Vec model performs worse than the other baseline models on recall but does well on precision.

The results obtained from the custom ML model are very encouraging given that it obtains higher scores than most of the existing text summarisation models, including TextRank and BERT. It can be argued that the custom model is a better fit for an online summariser because the summaries generated are much faster with the custom model than with BART or BERT which require high computational resources such as a GPU to summarise papers in a few seconds. With a GPU, BART can summarise papers in 3.71 seconds on average. The custom ML model takes on average 2.46 seconds. Without a GPU, the time taken becomes consequent for BART.

C. Human Evaluation

A human evaluation has been carried out. Summaries are generated using the custom ML model and the other baseline models mentioned. Three computer science undergraduate students were asked to rate summaries generated from 5 different papers. Each paper has 6 summaries (one for each model). The evaluators were asked to read the papers before reading the summaries. To avoid any bias, they were not told which of the summaries were from our own models. They were given 2 weeks to rate all the summaries. The 5 papers chosen for evaluation were all computer science research papers written in English. The papers covered different topics - machine learning(2), deep learning(1), cybersecurity(1), and networking(1). The documents contained on average 10 pages, with the shortest document containing 5 pages and the longest one containing 18 pages. The average length of summaries generated for each of the 6 models are shown in Table VI.

TABLE VI. Length of Summaries

Model	Average Length
BERT	276
BART	175
LSA	422
TextRank (Gensim)	154
Fine-Tuned Doc2Vec	298
Custom ML Model	237

The summaries are rated on a scale of 1 - 5 according to the scale that we developed as shown in Table VII.

TABLE V. Baseline and Custom Model Comparison

Model	Metric	Rouge-1	Rouge-2	Rouge-L
TextRank (Gensim)	Recall	0.30	0.10	0.27
	Precision	0.29	0.09	0.25
	F-score	0.29	0.09	0.255
BERT	Recall	0.22	0.07	0.20
	Precision	0.34	0.11	0.31
	F-score	0.26	0.08	0.24
BART	Recall	0.40	0.18	0.38
	Precision	0.32	0.15	0.30
	F-score	0.33	0.15	0.31
Fine-Tuned Doc2Vec	Recall	0.20	0.05	0.18
	Precision	0.34	0.09	0.31
	F-score	0.24	0.06	0.22
Custom ML Model	Recall	0.27	0.10	0.25
	Precision	0.35	0.13	0.33
	F-score	0.30	0.11	0.28

TABLE VII. Scale for Rating Summaries

1	Very poor summary. No important ideas were covered. No sentence cohesion. Off-topic. No idea what the paper is about.
2	Poor summary. Many important ideas are missing. Sentences are not very coherent (not a good flow of ideas). Unnecessary sentences. Difficult to understand what the paper is about.
3	Decent summary. Some ideas in the paper are covered. Some sentences are coherent. Can get a broad idea of what the paper is about.
4	Clear summary. Most ideas in the paper are covered. Good sentence cohesion. Small number of unnecessary sentences. Can get a grasp of what the paper is about.
5	Very clear summary. Contains the main ideas in the paper. Very good sentence cohesion. Very small or no unnecessary sentences. Very easy to understand what the paper is about.

Table VIII shows the average rating obtained for each model.

TABLE VIII. Human Evaluation Results

Model	Average Score
BERT	2.7
BART	3.7
LSA	3.3
TextRank (Gensim)	2.6
Fine-Tuned Doc2Vec	2.4
Custom ML model	3.8

The results show that the custom ML model receives the best scores on average. It is closely followed by BART. LSA also performs quite well. We conclude that the custom model generates clear summaries, covers most of the ideas in the papers, and has good sentence cohesion. There may be some unnecessary sentences but the reader can get a grasp of what the papers are about.

5. Conclusions

The high volume of papers available makes the research process time-consuming. Summarising papers can help researchers get an overview of their main ideas. In this paper, we have developed a system that summarises computer science research papers. We have shown that using statistical features such as document length, the number of foreign words, digits and symbols count and the presence of certain high frequency words to train a machine learning classifier algorithm for a domain-specific literature (Computer Science) can help achieve state-of-the-art results. The proposed solution performs better than TextRank and BERT. It can be argued that the proposed model is also better than BART for an online application as the time our model takes to summarise papers is shorter and requires less computation power than with BART. Our model also receives the best rating on average for the human evaluation.

Currently, we are using the abstract as the gold summary but creating a manual summary may increase the accuracy of the module. Moreover, the performance of the model may increase if training on a larger dataset. We also plan to use other machine learning classifiers and deep learning algorithms in future works.

References

[1] W. To and B. T. Yu, "Rise in higher education researchers and academic publications," *Emerald Open Research*, vol. 2, p. 3, 2020.

[2] K. E. White, C. Robbins, B. Khan, and C. Freyman, "Science and engineering publication output trends: 2014 shows rise of developing country output while developed countries dominate highly cited publications," *National Center for Science and Engineering Statistics InfoBrief*, pp. 1–7, 2017.

[3] N. I. Altmami and M. E. B. Menai, "Automatic summarization of scientific articles: A survey," *Journal of King Saud University-Computer and Information Sciences*, 2020.

[4] R. Mihalcea and P. Tarau, "Textrank: Bringing order into text," in *Proceedings of the 2004 conference on empirical methods in natural language processing*, 2004, pp. 404–411.

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[6] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," *arXiv preprint arXiv:1910.13461*, 2019.

[7] C.-Y. Lin, "Rouge: A package for automatic evaluation of summaries," in *Text summarization branches out*, 2004, pp. 74–81.

[8] P. B. Baxendale, "Machine-made index for technical literature—an experiment," *IBM Journal of research and development*, vol. 2, no. 4, pp. 354–361, 1958.

[9] H. P. Luhn, "The automatic creation of literature abstracts," *IBM Journal of research and development*, vol. 2, no. 2, pp. 159–165, 1958.

[10] M. A. Fattah and F. Ren, "Ga, mr, ffnn, pnn and gmm based models for automatic text summarization," *Computer Speech & Language*, vol. 23, no. 1, pp. 126–144, 2009.

[11] K. Ishikawa, S. Ando, and A. Okumura, "Hybrid text summarization method based on the tf method and the lead method," in *In Proceedings of the 2nd National Institute of Informatics Test Collection Information Retrieval (NTCIR) Workshop*. Citeseer, 2001.

[12] H. Edmundson, "P. 1969. new methods in automatic extracting," *Journal of the Association for Computing Machinery*, vol. 16.

[13] G. Erkan and D. R. Radev, "Lexrank: Graph-based lexical centrality as salience in text summarization," *Journal of artificial intelligence research*, vol. 22, pp. 457–479, 2004.

[14] J. Kupiec, J. Pedersen, and F. Chen, "A trainable document summarizer," in *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, 1995, pp. 68–73.

[15] M. Osborne, "Using maximum entropy for sentence extraction," in *Proceedings of the ACL-02 Workshop on Automatic Summarization*, 2002, pp. 1–8.

[16] J.-Y. Yeh, H.-R. Ke, and W.-P. Yang, "ispreadrank: Ranking sentences for extraction-based summarization using feature weight propagation in the sentence similarity network," *Expert Systems with Applications*, vol. 35, no. 3, pp. 1451–1462, 2008.

[17] V. Yatsko, M. Starikov, and A. Butakov, "Automatic genre recognition and adaptive text summarization," *Automatic Documentation and Mathematical Linguistics*, vol. 44, no. 3, pp. 111–120, 2010.

[18] D. R. Radev, H. Jing, M. Styś, and D. Tam, "Centroid-based summarization of multiple documents," *Information Processing & Management*, vol. 40, no. 6, pp. 919–938, 2004.

[19] S. Ghodratnama, A. Beheshti, M. Zakershahrak, and F. Sobhanmanesh, "Extractive document summarisation based on dynamic feature space mapping," *IEEE Access*, vol. 8, pp. 139 084–139 095, 2020.

[20] S. G. Haez and F. Shamsfakhr, "Extractive text summarisation using bayesian state estimation of sentences: A markovian framework," *Journal of Information Science*, 2022.

[21] Y. Liu and M. Lapata, "Text summarization with pretrained encoders," *arXiv preprint arXiv:1908.08345*, 2019.

[22] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever *et al.*, "Improving language understanding by generative pre-training," 2018.

[23] A. Farzindar and G. Lapalme, "Legal text summarization by exploration of the thematic structure and argumentative roles," in *Text Summarization Branches Out*, 2004, pp. 27–34.

[24] L. H. Reeve, H. Han, and A. D. Brooks, "The use of domain-specific concepts in biomedical text summarization," *Information Processing & Management*, vol. 43, no. 6, pp. 1765–1776, 2007.

[25] S. Gao, X. Chen, P. Li, Z. Ren, L. Bing, D. Zhao, and R. Yan, "Abstractive text summarization by incorporating reader comments," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 1, pp. 6399–6406, 2019.

[26] S. Erera, M. Shmueli-Scheuer, G. Feigenblat, O. P. Nakash, O. Boni, H. Roitman, D. Cohen, B. Weiner, Y. Mass, O. Rivlin *et al.*, "A summarization system for scientific documents," *arXiv preprint arXiv:1908.11152*, 2019.

[27] P. Hartl and U. Kruschwitz, "University of regensburg at checkthat! 2021: Exploring text summarization for fake news detection," in *CLEF (Working Notes)*, 2021.

[28] S. Abdullah, "Semi-extractive text summarization approach to extract research objective of academic literature," *International Journal of Computing and Digital Systems*, vol. 12, no. 1, pp. 665–674, 2022.

Sheik Muhammad Wakeel Bauboorally is currently a final year student of the BSc (Hons) Computer Science programme at the Department of Information and Communication Technologies, Faculty of Information, Communication and Digital Technologies, University of Mauritius. As part of the Industrial Training module, he had the opportunity to work as an intern at PwC Mauritius for a period of 10 weeks in 2021, where he had the opportunity to refine his skills as a programmer as well as his communication social skills. He is on his way to complete his undergraduate degree and will potentially join the ICT industry in October 2022. His research interests include full-stack web and mobile development, game development, machine learning, and deep learning.

Dr Sameerchand Pudaruth is an Associate Professor from the ICT Department at the University of Mauritius. He has a PhD in Artificial Intelligence (AI) from the University of Mauritius. He is a Senior member of IEEE, founding member of the IEEE Mauritius Subsection and has been a past Vice-Chair of the IEEE Mauritius Section. He is also a senior member of the Association for Computing Machinery (ACM). His research interests are Artificial Intelligence, Machine Learning, Data Science, Machine Translation, Computer Vision, Robotics, Mobile Applications, Web Technologies, Multimedia, Blockchain and Information Technology Law. He has written more than 70+ papers for national international journals and conferences. He has been in the organising committee of many successful international conferences such as Africon 2013, IST Africa 2014, Africon 2015, AfriCHI 2015, ICCCS 2015, BigData 2015, DIPECC 2015, Africhi 2016, Emergitech 2016, NextComp 2017, ISCMI 2017, Mauritian Academic Conference 2018, icABCD 2018, Mauricon ICONIC 2018, icABCD2019, NextComp 2019, icABCD2020, Elecom 2020 and Mauricon ICONIC 2020. He has also written a book entitled, 'Python in One Week'. He is a reviewer for IEEE Access and Environment, Systems and Decisions (Springer) journals, amongst many others.