Nanotechnology Perceptions
ISSN 1660-6795
www.nano-ntp.com

SEQABC: Revolutionizing Single Document Extractive Text Summarization with Quick Artificial Bee Colony

Jyotirmayee Rautaray[1], Sangram Panigrahi[2], Ajit Nayak[2], Meenakshi Kandpal[3], Pranati Mishra[4]

[1]Department of Computer Science & Engineering, Siksha 'O' Anusandhan(deemed to be University), Bhubaneswar Odisha, India
2Department of Computer Science & Information Technology, Siksha 'O' Anusandhan(deemed to be University), Bhubaneswar Odisha, India
[3]Department of Computer Science & Engineering, KIIT, Bhubaneswar, India
[4]Department of Computer Science & Engineering, Odisha University of Technology and Research, Bhubaneswar, Odisha, India

Amidst the vast ocean of textual content saturating the digital realm, Automatic Text Summarization (ATS) emerges as a critical tool. Our paper introduces an innovative single-document extractive summarization approach, harnessing the Quick Artificial Bee Colony (qABC) optimization algorithm to shift through source documents and extract pivotal lines for a concise summary. Motivated by honey bee foraging practices, qABC navigates the search space adeptly, identifying optimal snippets to construct a summary which encapsulates the core concepts of the original work. We evaluate our model by using the ROUGE score calculation on the BBC News Summary dataset, achieving improved ROUGE-1, ROUGE-2, and ROUGE-L measure results. Comparative analysis against existing algorithms underscores our model's superiority, setting new standards in single-document summarization. With our revolutionary method, come along as we alter the text summarization landscape.

Keywords: Automatic Text Summarization, Quick Artificial Bee Colony(qABC),Extractive Text Summarization, Nature-inspired algorithm , Single Document Text Summarization, ROUGE score,Generic Text Summarization.

1. Introduction

The internet is a wealth of knowledge regarding any subject in today's digital age. This has led to a large amount of data which is ever growing in size. The sheer volume of data makes it

impossible for a single person to read it all. People today want to access concise and relevant data on their topic as fast as possible. Using text summarization is one method of achieving it. The process of producing a clear, accurate, and cohesive overview of a lengthy text is known as text summarization. A summary is a condensed text that highlights the most crucial details from the original text, gathered from one or more sources. Usually, it doesn't extend past half of the text. Figure 1 demonstrates the categorization of Automatic Text Summarization models, where 1(a) displays the categorization on the basis of the size of input document, 1(b) shows the categorization on the basis of the output's nature and 1(c) displays the categorization relying on the summarization approach.

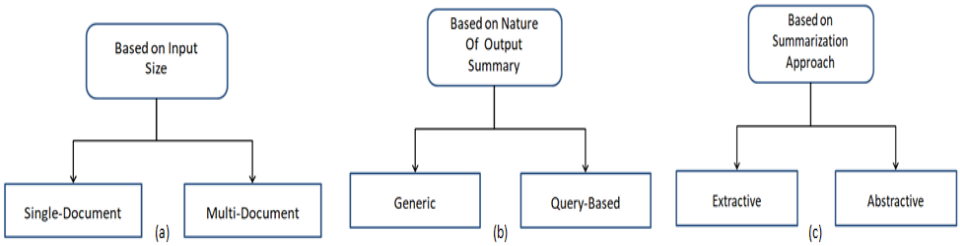

Figure 1(a). Categorization of text summarization on the basis of input size, 1(b). Categorization of text summarization on the basis of nature of output summary, 1(c). Categorization of text summarization on the basis of summarization approach

The text summarization methods are classified on the basis of method used for summarizing, type of summary produced and input size of the document [1]. The output summary can be either generic or query based on basis of the nature of output summary. In generic text summarization method, the goal is to produce a generic summary of the documents. In case of query based text summarization the summarizer generates the summary based on a given specific query. A multi-document summarizer that uses query-based summary works with a collection of uniform documents that have been selected from a large repository as a response to a query. The information related to the query is subsequently included in the summary that is generated. A query-based summary focuses on the information most relevant to the original search query, whereas a generic summary provides an overview of the whole content of the document.

The number of source documents needed to create the summary is referred to as the input size. Condensing the information of the input document while maintaining the essential points is the aim of Single-Document Summarization (SDS), which creates a summary from a single text document. Multi-Document Summarization (MDS) aims to eliminate repeated content from the input documents by creating a synopsis derived from a collection of input documents. Redundancy, compression ratio, temporal relatedness, coverage and other major difficulties are present in MDS, which is more complicated than SDS.

The text summarization task can be either extractive or abstractive, depending on the approach used for summarization. The process of extractive text summarization entails choosing the sentences with the greatest significance from the original text and arranging them so that a summary is produced. The objective of extractive summary is to retain the text's most crucial details while shortening it. This strategy frequently relies on statistical or machine learning

algorithms that pinpoint the most pertinent sentences based on attributes like keyword frequency or sentence length. Compared to the abstractive technique, the extractive approach is quicker and easier. Users may study the summary utilizing the precise terms found in the original document because the sentences have been directly extracted, improving accuracy. However in case of extractive text summarization the summary sentences lack semantics and coherence as a result of faulty sentence linking, unresolved co-reference relationships, and "dangling" anaphora.

Abstractive summarization entails constructing a new summary that is not always present in the original text, as opposed to extractive summarization, which requires choosing and copying significant sentences or phrases from the source material. Because abstractive summary calls for a more in-depth comprehension of the original text and the capacity to produce natural-sounding language, it is a more difficult assignment than extractive summarization. Abstractive summarization algorithms produce short, coherent summaries by combining machine learning strategies like deep learning with natural language creation. Abstractive summarization can produce more fluent and coherent summaries. However, abstractive summarization is still a challenging task for natural language processing systems, as it needs a deep comprehension of the meaning and context of original document. In addition, the quality of summary can be highly subjective and may vary depending on things like the target audience, the summary's goal, and the application's particular requirements.

We have attempted to utilize this paper to address the issue of generic extractive single document text summarization utilizing a swarm intelligence algorithm called as quick Artificial Bee Colony algorithm. A set of computer methods known as swarm intelligence algorithms is motivated by group behavior of social creatures like fish, birds, bees, and ants. These algorithms mimic the decentralized, self-organized, and collaborative nature of these organic systems to handle challenging optimization and decision-making problems. Swarm intelligence algorithms are typically used for optimization, search, and problem-solving tasks.When working with complex, high-dimensional, or dynamic problem spaces—where classic optimization techniques would have trouble—they are especially helpful. These algorithms leverage the power of collective intelligence and emergent behavior to find solutions that can be challenging to discover using other approaches.

The document's remaining sections are organized as follows: Section 2 covers the Background Study, while Section 3 covers the Methodology. Section 4 covers the experiment and results analysis, whereas Section 5 covers the conclusion.

2. Literature Review

Numerous endeavors have been undertaken to advance text summarizing systems through diverse methodologies, technologies, and resources. The past research done on extractive text summarization is presented in this section.

Verma et al. [2] offered a brand-new method for text summarization based on the clustering with gap statistics algorithm, TLBO evolutionary algorithm, and fuzzy inference system. Comparing the generated summaries by the suggested approach to reference summaries, the former encompass roughly 50% of the relevant topics. Three experts examined the prepared

summaries for deep observations, and kappa statistics was used to measure the data based on numerous features. The summarization result was good but the algorithm tuning was challenging and there was loss of important details.

Fang et al. [3] proposed a new extractive summarization model called CoRank which integrated an unsupervised ranking model based on graphs with the word-sentence relationship. The CoRank model worked by first constructing a bipartite graph, where the nodes on one side represented the words in the document and the nodes on the other side represented the sentences. The co-occurrence link between words and sentences was represented by the edges connecting the nodes. Once the graph was constructed, the CoRank model used a graph-based unsupervised ranking algorithm to order the document's sentences. The ranking algorithm took into account factors like the importance of the words in each sentence, the co-occurrence relationship between words and sentences and connectivity of the sentences in the graph. This method suffered from less efficiency for large documents.

Joshi et al. [4] put forth a novel unsupervised framework called SummCoder to perform extractive text summarization. Deep auto-encoders, a kind of neural network that can be used to learn data representations, served as the foundation for SummCoder. SummCoder worked by first learning a representation of the input document using a deep auto-encoder. Once the representation had been learned, SummCoder used a simple decoder to produce a summary of the document. The decoder was trained to generate summaries that were similar to the input documents in terms of their meaning. SummCoder had a number of advantages over other extractive summarization methods. First, it was unsupervised, which means that it did not require any labeled data to train. Second, it used a deep auto-encoder to learn representations of the input documents, which allowed it to capture complex relationships between words and sentences. Third, it used a simple decoder to generate summaries, which made it efficient and easy to implement. Although being good, it was computationally time consuming.

Rouane et al. [5] suggested a new biomedical text summarization method which combined frequent item-sets mining and clustering. The proposed method worked by first clustering the sentences in the biomedical text into semantically similar groups. The K-means clustering technique was employed in it. Once the sentences had been clustered, the proposed method used frequent item-sets mining to identify the most important concepts in each cluster. The proposed method then selected the sentences that contained the most important concepts from every cluster to produce the summary of the biomedical text. However, scalability of this method was low and there exist privacy concerns.

Fitrinah et al. [6], suggested a brand-new method for extractive summarization of journal articles in the field of science. Their approach used a combination of Gated Recurrent Unit (GRU) networks and Long Short-Term Memory (LSTM) for extracting significant sentences from a document. The proposed method consisted of feature extraction, sentence scoring and summary generation. It used LSTM or GRU networks to learn long-range dependencies in the text, which allowed it to better understand the context of each sentence. It considered a variety of features when scoring sentences, which made it more robust to different types of documents. It was able to generate summaries that are both informative and concise. Overall, it was a well-written and well-organized paper that proposed a novel and effective approach to extractive text summarization of journal articles in the field of science. However fine tuning this model

was difficult.

Pati et al. [7], suggested a novel method for single document extractive text summarization using the cuckoo search algorithm (CSA).The CSA is a nature-inspired algorithm that copies the breeding behavior of cuckoos. Other birds' nests are where cuckoos deposit their eggs, and baby cuckoos often outcompete the other chicks for food and attention. The authors use the CSA to select the most important sentences in a document for summary. They define a fitness function for each sentence that measures the importance of the sentence based on a variety of factors, such as the sentence position, the number of keywords, and the presence of certain phrases. The CSA then iteratively selects sentences from the document and adds the sentences to the summary. At each iteration, the CSA compares the fitness of the selected sentences to the fitness of the sentences in the summary. If a selected sentence has a higher fitness than a sentence in the summary, the sentence in the summary is replaced with this selected sentence. The CSA terminates when a certain number of iterations have been performed or when a satisfactory summary has been generated. The authors evaluate their approach on the DUC 2003 dataset, and show that it outperforms other cutting edge techniques for extractive text summarization with regard to ROUGE scores. It is simple to implement and computationally efficient. It is able to produce summaries of high quality for a variety of documents. It is robust to different types of noise in the data. Overall, it is a well-written and well-organized paper that proposes a new and effective strategy for extractive text summarization in context of a single document. The authors have conducted a comprehensive experimental evaluation of their approach, and the results are promising.

Ruan et al. [8], suggested HiStruct+, a novel approach to perform extractive text summarization that leverages hierarchical structure information within a text to improve summarization quality. HiStruct+ incorporates this information into an encoder-only Transformer model, leading to significant improvements in ROUGE scores compared to existing methods. Traditional extractive summarization methods often treat text as a linear sequence, neglecting the inherent hierarchical structure present in most text documents. This structure, evident through sections and headings, can provide valuable cues about the importance and organization of information within the text. HiStruct+ explicitly encodes hierarchical structure information into the summarization model. This is achieved by formulating hierarchical positions for each sentence based on its location within the section hierarchy, extracting section titles and embedding them into the sentence representations and utilizing two stacked inter-sentence Transformer layers for contextual learning with hierarchical structure information. A sigmoid classifier is used for final sentence selection, ensuring concise and informative summaries. HiStruct+ is evaluated on three benchmark datasets (CNN/DailyMail, PubMed, and arXiv). It achieves significant improvements in ROUGE scores compared to a strong baseline without hierarchical structure information. The results suggest that HiStruct+ effectively utilizes hierarchical structure information to generate more accurate and relevant summaries.

Aote et al. [9] suggested a novel hybrid approach for multi-document extractive text summarization that is applicable for Hindi documents, utilizing an improved Genetic Algorithm (IGA) along with Binary Particle Swarm Optimization (BPSO) .The proposed approach operated in two phases. BPSO phase generated candidate summaries by iteratively updating particles based on their fitness and the swarm's best positions and IGA phase refined

the best BPSO solution by applying genetic operations (selection, crossover, mutation) to optimize feature weights and generate the final summary. The scope of this method was narrow considering its application on native language.

Saini et.al [10] explored three multi-objective optimization algorithms: grey wolf optimizer, self-organizing differential evolution, and water cycle algorithm. Each algorithm was adapted to the specific task of extractive summarization, resulting in novel approaches called ESDS_MGWO, ESDS_SMODE and ESDS_MWCA respectively. These approaches utilized the strengths of the chosen algorithms to effectively balance informativeness and conciseness in the generated summaries. While the paper provided valuable contributions, some areas required further exploration. Additionally, a deeper analysis of the trade-off between informativeness and conciseness for various parameter settings would provide valuable insights.

Al-Saleh et al. [11] suggested an approach for automatic summarization using an Ant Colony System (ACS) algorithm. ACS is a computational algorithm inspired by the behavior of ants foraging for food. In the context of summarization, the "ants" represented potential summaries, and the "food" represented high-quality summaries that accurately captured the important information across all documents. The authors adapted the ACS algorithm to evaluate different sentence combinations for inclusion in the summary. Sentences were assigned weights based on factors like informativeness, redundancy, and novelty. Their approach incorporated a "pheromone" mechanism similar to real ant colonies. Pheromones represented the "goodness" of certain sentence selections, guiding the "ants" towards better summary combinations. Overall, this work contributed to the domain of text summarization by exploring a bio-inspired approach using an Ant Colony System.

Pattanaik et al. [12] explored the use of the Bat Algorithm for extractive summarization. Inspired by bats' echolocation behavior, the Bat algorithm is a metaheuristic optimization technique. The authors aimed to develop a method for extractive summarization that leverages the bat algorithm for sentence selection. The bat algorithm was employed to optimize the selection of sentences for summary. The sentences are evaluated based on features that indicate their importance to the overall meaning of the text. Reducing sentence redundancy in the final summary that was produced was the primary goal of this model.

Raed et al. [13] suggested extracting summaries from Arabic single documents using the Firefly algorithm. The paper proposed the combined use of both informative and semantic scores. Using the Firefly algorithm, the optimal sub path among the possible paths in the graph was discovered. The Firefly method is in charge of extracting the ideal path from every path in graph, each of which is a candidate summary. However, the evaluation method yielded positive findings for the Firefly-based summary approach, indicating an improvement in the performance and quality of Arabic summaries.

Krishnan et al. [14] proposed a novel approach to text summarization that leveraged knowledge graphs (KGs) and Flower Pollination Algorithm (FPA) to help in enhancing the completeness and quality of summaries. The method utilized knowledge graph to encode complex semantic relationships between concepts mentioned in the text. This allowed the model to reason beyond the surface level and consider relevant background knowledge. The core concept of Know Sum is semantic alignment, which involves aligning the text with

entities and relations present in the knowledge graph. This helped the model identify the most essential information in the document based on its connections to the broader knowledge base. This led to generation of summaries that are more informative and more comprehensive.

Karaboga et al. [15] introduced the ABC algorithm, mimicking the foraging behavior of honey bee colonies. Employing three distinct groups of bees, it leveraged their coordinated efforts to explore the search space and identify optimal solutions. Employed bees tirelessly explore the vicinity of known food sources (potential solutions), seeking better options. Onlooker bees, guided by the employed bees' findings, prioritized the most promising sources for further investigation. Lastly, scout bees played a crucial role in discovering new territories when existing food sources became exhausted, ensuring the algorithm's capacity to break out of local optima and fully explore the search space. The ABC algorithm had demonstrated its effectiveness in tackling a wide range of optimization problems across various domains. From engineering design and scheduling to signal processing and machine learning, it had proven to be a valuable instrument for scientists and practitioners alike. In power systems optimization and financial forecasting, ABC had also shown promising results, highlighting its potential for addressing complex real-world challenges. Simplicity and ease of implementation were key advantages of the ABC algorithm. Its ability to efficiently explore large and complex search spaces made it a valuable tool for tackling intricate problems. However high dimensionality problems posed challenges, and sensitivity to initial parameter settings required careful consideration. Additionally, convergence speed was slower compared to other algorithms, potentially impacting its performance on time-sensitive tasks.

After critically reviewing the existing methods used in extractive text summarization, we concluded that the existing methods are inefficient with regard to redundancy, speed of extraction of a sentence and accurate compression. Thus, to address the above issues, we proposed quick Artificial Bee Colony (qABC) algorithm as our optimization technique for generic single document extractive text summarization.

3. Methodology

The proposed model Single Document Extractive Summarization using Quick Artificial Bee Colony (SEQABC) is an extractive summarizer based on optimization using Artificial Bee Colony algorithm. Our proposed model SEQABC is a generic extractive single document text summarization model that takes a single document as an input for the task of text summarization. The steps involved in the SEQABC model are preprocessing, TF-ISF score calculation, mean vector calculation, quick artificial bee colony algorithm, summary generation and summary evaluation. The brief outline of the SEQABC model is illustrated in Figure 2.

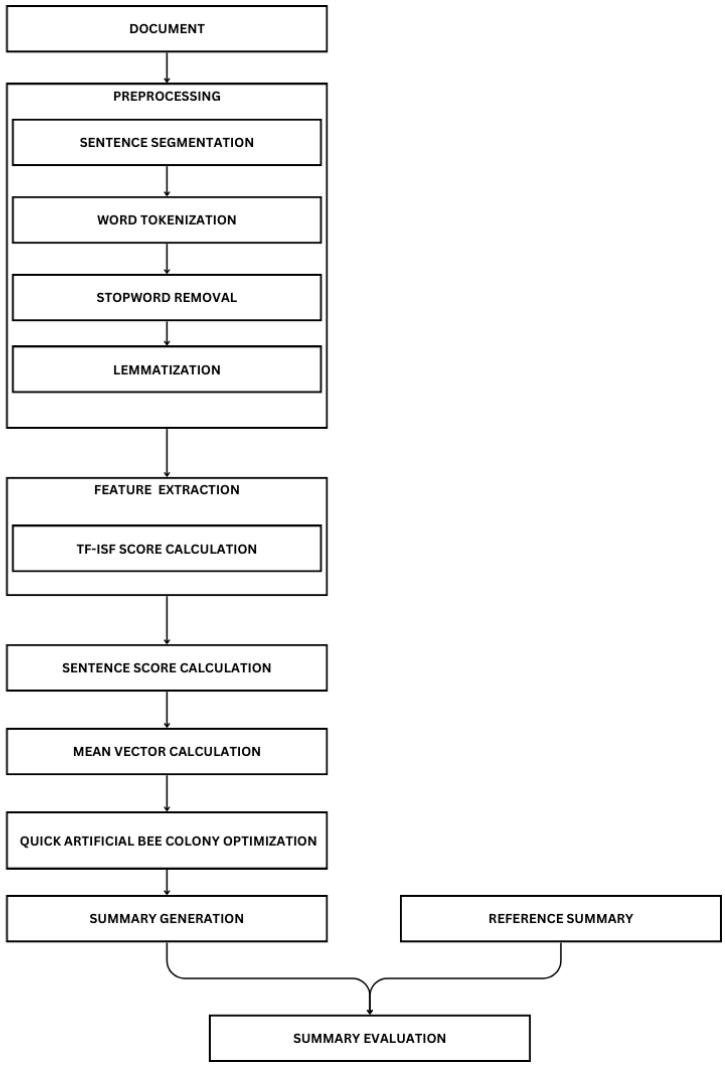

Figure 2. Block Diagram of SEQABC model

3.1. Text Preprocessing:

The first stage of text summarization is called text pre-processing, during which unprocessed text data is arranged, cleaned, and formatted so that summarization algorithms can use it. The steps include as following:

3.1.1. Sentence Segmentation: It is the process of breaking the text up into individual sentences so that we can work with smaller content units. From the document we get a set of sentences D such that D = $\{S_1,S_2,\ldots, Sn\}$, where n represents total number of sentences.

3.1.2. Word Tokenization: The individual words of each sentence are taken as tokens. T = $\{t_1,t_2\ldots t_m\}$, where m represents the total number of words in the document.

3.1.3. Stop word Removal: Common words like "the", "and", "in" etc. don't have any

significant meaning and can be safely eliminated from the text.

3.1.4. Lemmatization: It is the process of reducing the words into their base form.

3.2. Term Frequency – Inverse Sentence Frequency:

The TF-ISF is calculated using Eq. (1) as follows

$$w_{ik} = tf_{ik} * \log(n/n_k) \qquad (1)$$

where w_{ik} reperents the weight of the word t_k in the sentence vector S_i, tf_{ik} represents the term frequency of the word t_k in the sentence S_i and n_k represents the number of sentences containing the word t_k.

3.3. Sentence Score Calculation:

Each sentence S_i present in the document is represented as a vector in the m-dimensional space, such that $S_i = \{w_{i1}, w_{i2} \ldots w_{im}\}$ where i ranges from 1 to n and each element represents the weight w_{ik} of corresponding term.

3.4. Mean Vector Calculation:

The mean vector $o = \{o_1, o_2 \ldots o_m\}$ represents the mean of the main content of document quantitatively. It is calculated with the help of Eq. (2) as follows:

$$o_k = \frac{1}{n}\sum_{i=1}^{n} w_{ik} \quad \text{, where } k = 1,2 \ldots m \qquad (2)$$

where n represents total number of sentences and m represents the total number of words in the document.

3.5. Quick Artificial Bee Colony Optimization Algorithm:

Quick Artificial Bee Colony [16] is a population-based optimization that simulates the behavior of honeybee swarm to solve optimization problem. Divided into two equal parts, the colony consists of employed and onlooker bees. The possible candidate summaries that consist of set of sentences are the possible solutions for this algorithm and these solutions are considered as flowers or food sources. The fitness value of the objective function for a solution is considered as amount of nectar for that solution. We have taken content coverage as the objective function in our model. The aim of the qABC algorithm is to optimize this objective function by finding out the solution for which the value of content coverage is highest. The content coverage is calculated by using Eq. (3) as follows:

$$F_{coverage}(x) = \sum_{i=1}^{n} \mathrm{sim}(s_i, o).\, x_i \qquad (3)$$

Here x_i is a binary decision variable with possible values as 0 or 1 depending on whether or not the sentence s_i is present in the generated candidate summary. Moreover $\mathrm{sim}(s_i,o)$ represents the cosine similarity between the sentence s_i and mean vector o. The cosine similarity is calculated using the Eq. (4) as follows:

$$\mathrm{sim}(s_i, s_j) = \frac{\sum_{k=1}^{m} w_{ik} w_{jk}}{\sqrt{\sum_{k=1}^{m} {w_{ik}}^2 . \sum_{k=1}^{m} {w_{jk}}^2}}, \quad \text{where } i, j = 1,2 \ldots n \qquad (4)$$

The fitness value of a solution is calculated using the objective function F(x) as follows:

$$\text{fit}(x) = \begin{cases} {}^{1}/_{(1+F(x))}, & \text{if } F(x) \geq 0 \\ 1 + \text{abs}(F(x)), & \text{if } F(x) < 0 \end{cases} \quad (5)$$

Figure 3 given below represents the way qABC optimizes the problem to get the candidate summary with best fitness value. The output of the given algorithm is the summary with the highest content coverage.

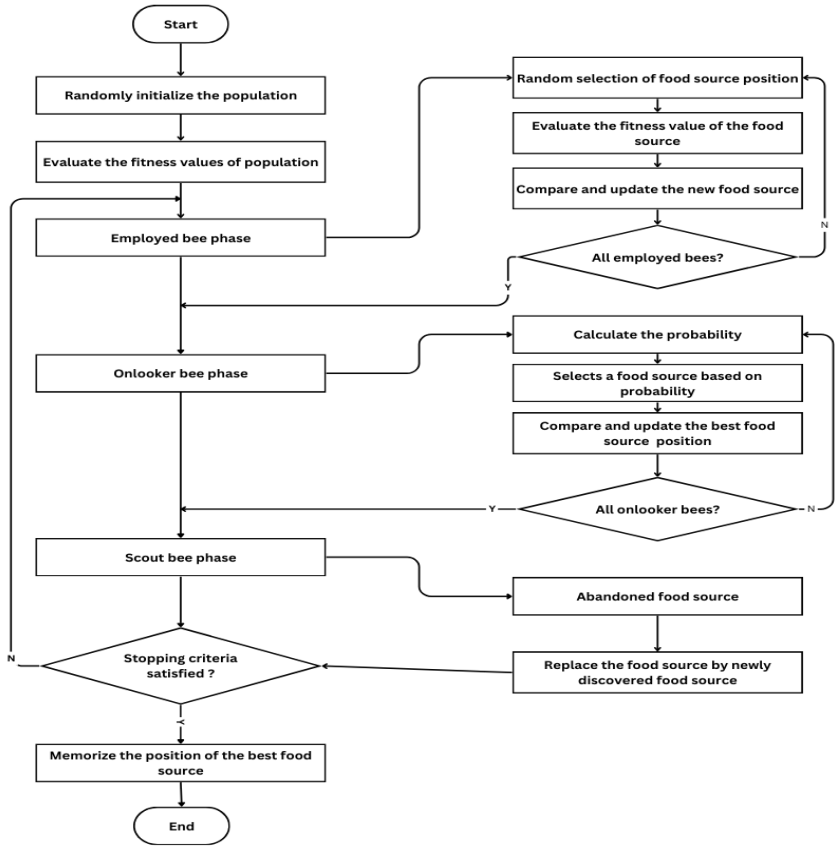

Figure 3. Flowchart of qABC optimization algorithm

The qABC algorithm takes the sentence vectors as input and gives the generated summary as output. It does some by the following steps:

Step 1: The employed bees are sent onto the initial food sources or solutions. These initial solutions are generated randomly by selecting the sentences for each solution randomly. The number of solutions generated is equal to the number of employed bees present in the bee colony.

Step 2: The subsequent 6 steps are reiterated until the termination condition is met, meaning the current iteration matches the maximum iteration count.

Step 3: The employed bees are sent onto the solutions and their fitness values are determined.

The fitness value of the solutions is compared and the solutions are updated.

Step 4: The probability value of the solutions with which they are preferred by the onlooker bees is calculated. This probability is calculated using the fitness values of the solutions.

Step 5: A random number between 0 and 1 is generated and the onlooker bees choose only those solutions for which the probability calculated in last step greater than the random number generated. The onlooker bees are sent to these solutions and their fitness values are calculated. The fitness value of the solutions is compared and the solutions are updated. In qABC we use the best neighbor in the neighborhood for updating the solution. The best neighbor is found by using the average Euclidean distance of all other food sources.

Step 6: The exploitation of the solutions exhausted by the bees is stopped. These exhausted solutions refer to the solutions which have not been updated for multiple continuous iterations.

Step 7: The employed bees of these exhausted solutions enter into scout bee phase and randomly select new solutions to replace the exhausted solutions.

Step 8: The best solution so far is memorized.

After completion of all the iterations of the qABC algorithm the best solution memorized in the last iteration is given as the output. This final generated summary is called as the candidate summary.

3.6. Summary Evaluation

The quality of the candidate summary is evaluated using Recall-Oriented Understudy for Gisting Evaluation (ROUGE) score. The ROUGE score is calculated using the candidate summary generated by the model and a reference summary already present in the dataset. For calculation of ROUGE score we make use of two values called as recall and precision, which are evaluated as follows:

$$\text{RECALL} = \frac{\text{Overlapping number of n-grams}}{\text{Number of n-grams in the reference summary}} \quad (6)$$

$$\text{PRECISION} = \frac{\text{Overlapping number of n-grams}}{\text{Number of n-grams in the candidate summary}} \quad (7)$$

We use precision and recall to calculate the F-score, which is considered as the ROUGE score. F-score is calculated follows:

$$\text{F} - \text{SCORE} = \frac{2*\text{RECALL}*\text{PRECISION}}{(\text{RECALL}+ \text{PRECISION})} (8)$$

Moreover, we are calculating 3 types of ROUGE scores here. In ROUGE-1 we calculate the number of common unigrams or words. In ROUGE-2 we calculate the number of common bigrams. In ROUGE-l we make calculations using the longest common sequences.

4. Experiment & Result Analysis

In this paper we have taken the BBC News Summary [17] dataset for evaluation of the performance of the proposed model. Moreover, we have set the colony size as 30, the maximum number of iterations as 5000 and the maximum number of trials as 100. The compression ratio is set at 40 % i.e. the number of sentences in the summary is 40% that of

the number of sentences in the document. The performance of our model is then compared with other single document text summarization models using Ant Colony Optimization (ACO), Bat Algorithm (BA), Firefly Algorithm (FA)and Flower Pollination Algorithm (FPA).

Table 1. ROUGE-1 Score Evaluation

z	ACO	BA	FA	FPA	qABC
Document 1	0.54	0.35	0.34	0.28	0.45
Document 2	0.35	0.35	0.57	0.55	0.68
Document 3	0.58	0.55	0.35	0.53	0.43
Document 4	0.35	0.30	0.32	0.35	0.56
Document 5	0.30	0.30	0.31	0.31	0.62

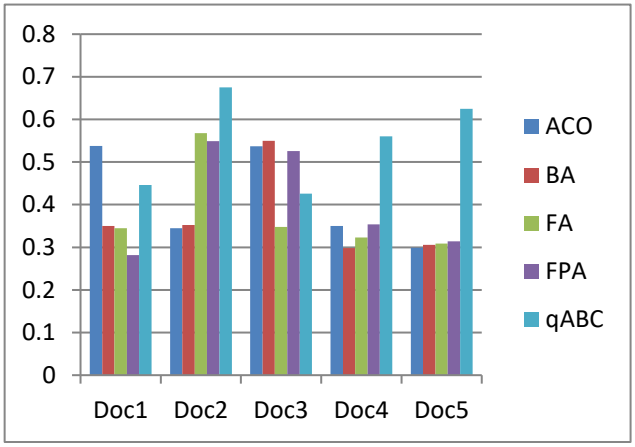

Fig 4: ROGUE Score comparison

Table 2. ROUGE-2 Score Evaluation

	ACO	BA	FA	FPA	qABC
Document 1	0.50	0.26	0.26	0.25	0.28
Document 2	0.25	0.25	0.50	0.51	0.62
Document 3	0.50	0.51	0.26	0.50	0.21
Document 4	0.22	0.22	0.21	0.23	0.45
Document 5	0.25	0.25	0.25	0.24	0.54

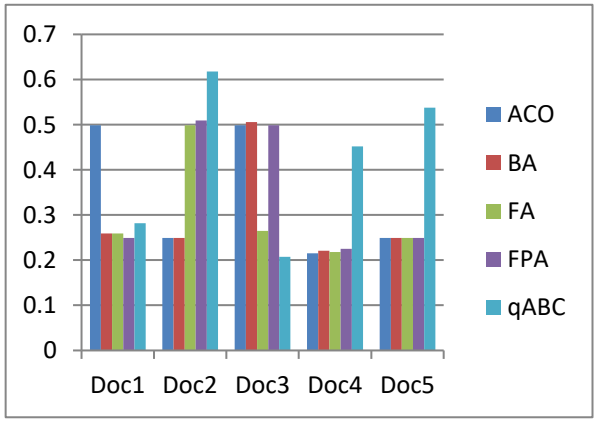

Figure 5. ROUGE-2 Score Comparison Graph

Table 3. ROUGE-L Score Evaluation

	ACO	BA	FA	FPA	qABC
Document 1	0.53	0.33	0.34	0.28	0.43
Document 2	0.32	0.30	0.54	0.54	0.68
Document 3	0.53	0.53	0.33	0.53	0.39
Document 4	0.31	0.30	0.29	0.33	0.56
Document 5	0.29	0.30	0.30	0.29	0.61

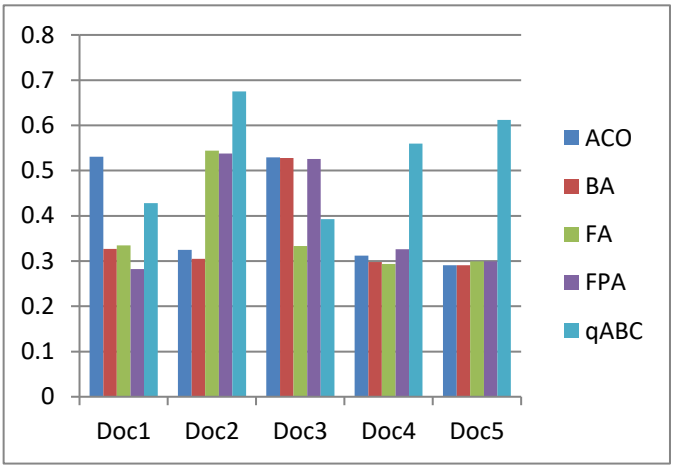

Figure 6. ROUGE-L Score Comparison Graph

By observing table 1 and figure 4 we see that the ROUGE-1 score of our model for documents 2,4 and 5 are 0.68,0.56 and 0.62 respectively and they outperform the other algorithms in case of these documents. As seen from table 2 and figure 5, for those same documents the ROUGE-2 scores are 0.62, 0.45 and 0.54, which are the highest scores among all the methods in that category. We see a similar pattern in case ROUGE-L as seen in table 3 and figure 6, where our model outperforms the other optimization methods. Moreover, in case of document 1 we can see that qABC optimization gives the second-best results for all three ROUGE scores. So, by observing all the above tables and graphs we can see that qABC performs better than other methods in most cases.

5. Conclusion and Future Scope

This paper delves into a generic single-document extractive text summarization model inspired by the Quick Artificial Bee Colony optimization algorithm. Utilizing Quick Artificial Bee Colony (qABC) optimization has demonstrated promise in generating informative and concise summaries. This technique harnesses the collective intelligence of the qABC algorithm, simulating honey bees' foraging behavior to navigate the summary search space and identify optimal summaries. The model's output is then assessed using various ROUGE F-Scores. By comparing it with several other nature-inspired algorithms like Ant Colony Optimization (ACO), Bat Algorithm (BA), Flower Pollination Algorithm (FPA), and Firefly Algorithm (FA), the effectiveness of the proposed model is demonstrated. Among these optimization techniques, the model employing the Artificial Bee Colony optimization algorithm outperforms the others.

In future combining ABC optimization with other techniques, such as machine learning, could

further enhance the effectiveness and efficiency of text summarization. Moreover, multi-objective optimization approach can be considered to enhance the performance of the model.

References

1. Wafaa S. El-Kassasa, Cherif R. Salamaa, Ahmed A. Rafeab, Hoda K. Mohameda (2020).Automatic text summarization: A comprehensive survey.
2. Pradeepika Verma, Anshul Verma, Sukomal Pal (2022). An approach for extractive text summarization using fuzzy evolutionary and clustering algorithms.
3. ChangjianFanga, Dejun Mua, Zhenghong Denga, Zhiang Wub (2016). Word-sentence co-ranking for automatic extractive text summarization.
4. Akanksha Joshi, E. Fidalgo, E. Alegrea, Laura Fernández-Robles (2019). SummCoder: An unsupervised framework for extractive text summarization based on deep auto-encoders.
5. Oussama Rouanea, HaceneBelhadef, Mustapha Bouakkaz (2019). Combine clustering and frequent itemsets mining to enhance biomedical text summarization.
6. Devi Fitrianah, Raihan Nugroho Jauhari (2021). Extractive text summarization for scientific journal articles using long short-term memory and gated recurrent units.
7. Siba Prasad Pati, Rasmita Rautray (2022). Single document extractive text summarization using cuckoo search algorithm.
8. Qian Ruan, Malte Ostendorff, Georg Rehm (2023). HiStruct+: Improving Extractive Text Summarization with Hierarchical Structure Information.
9. Shailendra S. Aote, Anjusha Pimpalshende , Archana Potnurwar , Shantanu Lohi (2023) . Binary Particle Swarm Optimization with an improved genetic algorithm to solve multi-document text summarization problem of Hindi documents.
10. Naveen Saini, Sriparna Saha, Anubhav Jangra, Pushpak Bhattacharyya (2019). Extractive single document summarization using multi-objective optimization: Exploring self-organized differential evolution, grey wolf optimizer and water cycle algorithm.
11. Asma Al-Saleh, Mohamed El Bachir Menai (2018).Ant Colony System for Multi-Document Summarization.
12. Anshuman Pattanaik, Santwana Sagnika, M.N Das, B.S.P Mishra (2019). Extractive Summary: An Optimization Approach Using Bat Algorithm.
13. Raed Z. Al-Abdallah, Ahmed T. Al-Taani (2019). Arabic Text Summarization using Firefly Algorithm.
14. N Krishnan, Gerard Deepak (2021). KnowSum: Knowledge Inclusive Approach for Text Summarization Using Semantic Allignment
15. Dervis Karaboga, Selcuk Aslan, Hasan Badem (2020). A new artificial bee colony algorithm employing intelligent forager forwarding strategies.
16. Dervis Karaboga, Beyza Gorkemli (2014). A quick artificial bee colony (qABC) algorithm and its performance on optimization problems.
17. BBC News Summary, https://www.kaggle.com/datasets/pariza/bbc-news-summary.