CHAPTER - 1

SUMMARY OF THE BASE PAPER

Title: **Multi-Objective Extractive Text Summarization**

Journal Name: Engineering Applications of Artificial Intelligence

Publisher: Elsevier.

Year: 2023.

Title of Base Paper: A new multi-objective evolutionary algorithm for citation-based summarization: comprehensive analysis of the generated summaries.

Summary of the Base Paper:

As the number of publications in all areas of scientific literature are increasing significantly, it has become difficult for collecting and analysing a set of papers from a certain topic with respect to time consumption. One of the potential solutions for this problem will be using automatic text summarization techniques. To be more specific scientific summarization methods.

Scientific Summarization helps in giving the readers a short and precise knowledge about the important representation of the most relevant impacts, information, and other factors of the paper. Scientific papers usually abide by a template such as – introduction, hypothesis, methods, results, discussions and conclusions.

1.1 Introduction:

There are two classifications of scientific paper summaries. Traditionally, abstract present in the paper acts as a reference summary. Even though it gives an overview about the publication, it sometimes fails to portray the viewpoint of the author in a more complete manner. Also, the abstract cannot involve all the aspects and contributions provided in the paper because of the size limitations. The contributions included in the abstract may also not be relevant to the scientific community. These issues paved the way for the introduction of citation-based summary. It includes not only the text of the reference paper but also the subsequent citations referring to them.

Usually, the summary will be of a greater quality if several objectives are optimized such as content coverage and redundancy reduction [6]. This project focuses on using multi-objective optimization since Single-Objective optimization methods cannot optimize conflicting objectives like coverage and redundancy reduction at the same time.

1.2 Demerits of Existing System:

- Single objective methods tackle **only one objective**
    - In this case, all criteria are weighted to obtain a **unidimensional function**
    - The definition of the weights **is subjective** and very influential for the final solution.
- Existing systems do not use:
    - Topic Level Information.
    - Faster Repair schemes.
    - Better sentence scoring schemes.

1.3 Merits of Proposed System:

- Text document summarization problem **involves more than one objective.**
- Multi Objective optimization approaches – tackle **tradeoff between conflicting objectives**
- Population-based approaches
    - allows for a balance between exploration and exploitation.
- Domain information such as **topic modelling** is done to try and improve the accuracy.

1.4 Dataset:

- The model performance results have been assessed against the multi-document summary dataset given by Document Understanding Conference (DUC).
- It is open benchmark for generic automatic summarization evaluation that is provided by National Institute of Standards and Technology.

- The dataset has been obtained from DUC2002.

1.5 Motivation:

1. The main motivation for doing this project is to enable research and other readers to access information with the help of concise summaries that contain the key insights, instead of going through larger volumes of text.
2. Many contemporary Single – Objective Optimization Algorithms have failed to address several conflicting factors involved in Text Summarization problem at the same time like : coverage, redundancy reduction, length, topic modelling and so on.
3. To understand how Optimization Algorithms can tackle real-time problems like text Summarization.

CHAPTER - 2

OBJECTIVE

Extractive Text Summarization (ETS) is crucial in information retrieval, helping users quickly understand large volumes of text by selecting and presenting the most relevant sentences. We have chosen to use Multi Objective Optimization (MOO) as a research direction in tackling Extractive Text Summarization [20]. A summary must have good coverage of topics and less occurrence of redundant sentences [7].

Also, a major constraint in ETS is to ensure summary length is not exceeded in selecting sentences. We aim to incorporate topic-based information, to increase the convergence speed of the chosen algorithm. In the context of Extractive Text Summarization (ETS), MOO is useful because it allows for a balance between achieving comprehensive topic coverage and minimizing redundancy within a specified summary length [8, 9].

To achieve this, Multi-Objective Artificial Bee Colony (MOABC) and Non-dominated Sorting Genetic Algorithm II (NSGA-II) are powerful methods for tackling large multi-objective problems [5]. MOABC uses the foraging behaviour of bees to efficiently explore and exploit the solution space, leading to rapid convergence. NSGA-II, on the other hand, employs genetic algorithms with a non-dominated sorting mechanism to maintain a diverse set of solutions, allowing it to handle a wide range of objectives effectively.

By using MOO techniques like MOABC and NSGA-II, we can create summaries that are both concise and comprehensive, satisfying the constraints of coverage, redundancy, and length. These algorithms' speed and robustness make them valuable tools in ensuring that the final summary meets all desired criteria, providing users with informative and efficient content summaries.

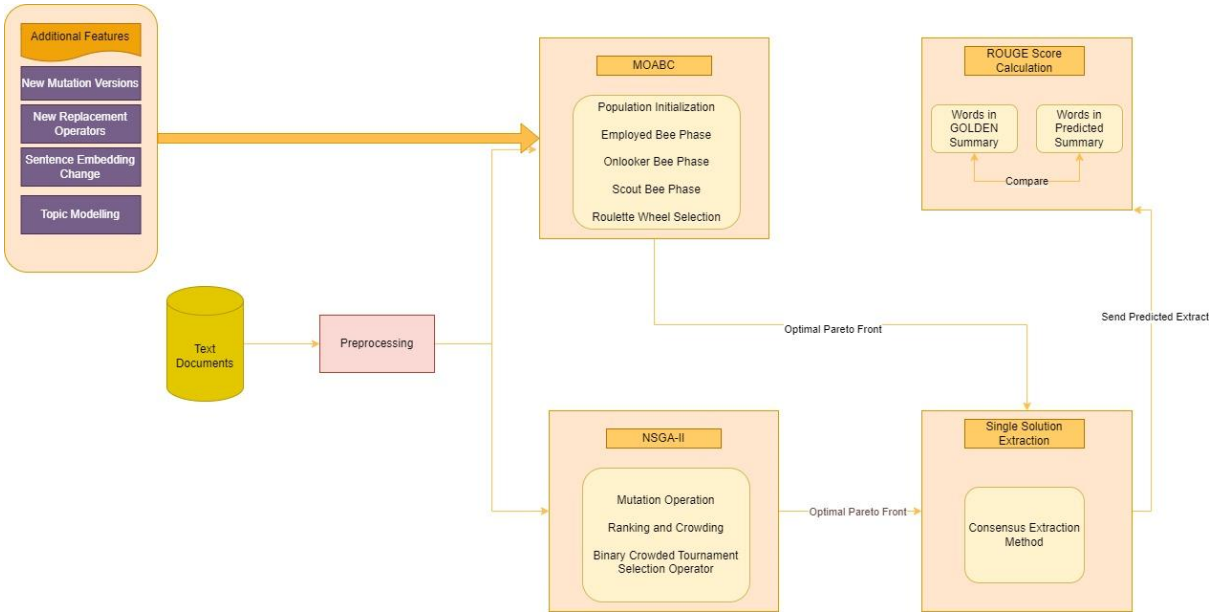


Fig 2.1 - Architecture Diagram

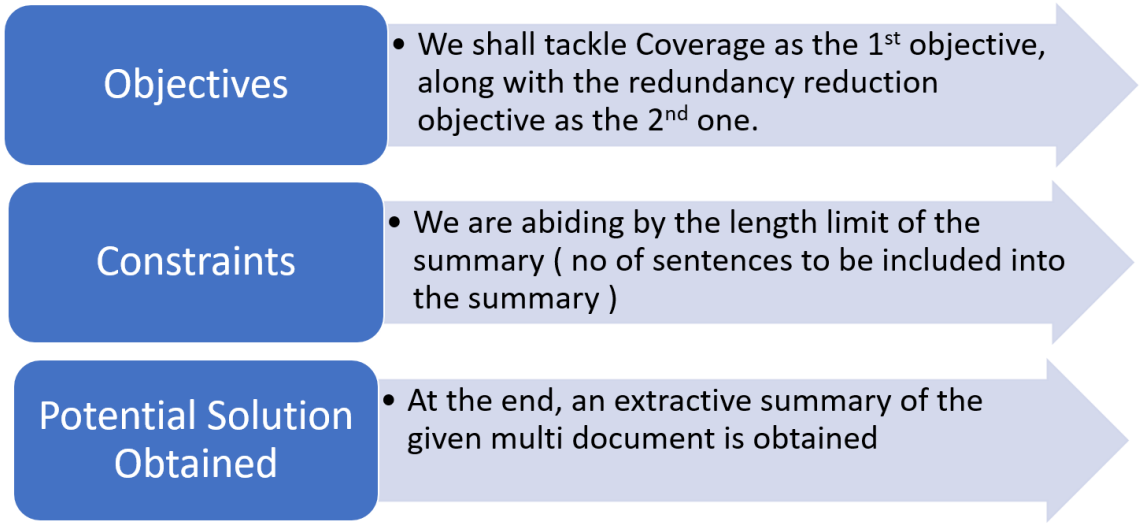


Fig 2.2 – Objectives, Constraints and Solution

CHAPTER-3

METHODOLOGY

3.1 Data Preprocessing and Representation:

3.1.1 Data Pre-processing:

- Data pre-processing consists of several steps as mentioned below
- **Segmentation:** This step involves the splitting of each and every separate sentence from the document.
- **Tokenization:** This step involves the process of splitting each sentence obtained after segmentation, into separate words. Interrogation, exclamation marks and other special symbols are removed in this step.
- **Remove stop words:** In this step the stop words are removed. They are nothing but the common words that do not have any meaning.
- **Stemming:** This step involves the extraction of the root form of each and every word using NLTK package (Porter Stemmer or lemmatizer).

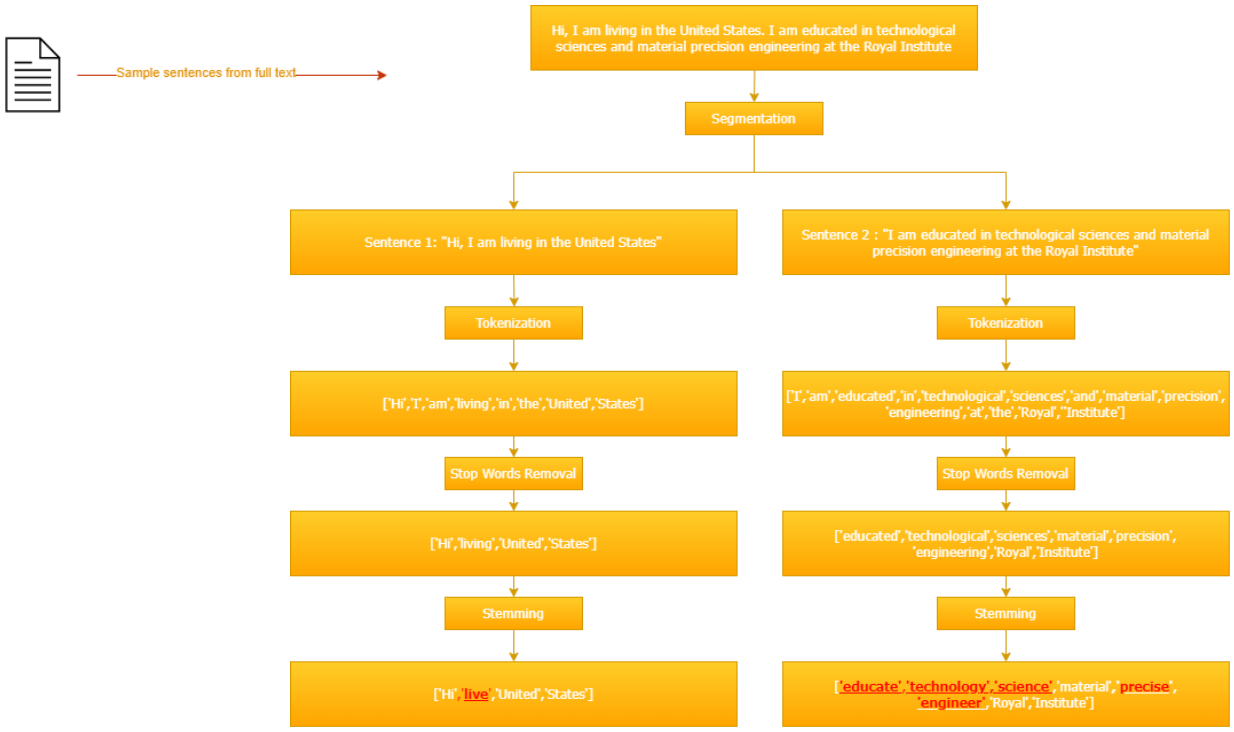


Fig 3.1 - Pre-processing of data.

3.1.2 Sentence Representation:

- We have chosen to use TF-ISF or term frequency – inverse sentence frequency representation.
- This is the most used representation in unsupervised learning approaches in extractive text summarization [4].
- We use the TF-ISF representation since the ISF part captures the presence of unique terms across the document that occur less frequently.
- At the end 2 components are obtained:
    - The w_{ik} matrix – which reflects the weight of a term k in a sentence i.
    - The mean document vector o – this reflects the centre of the document and is the average of w_{ik} vectors

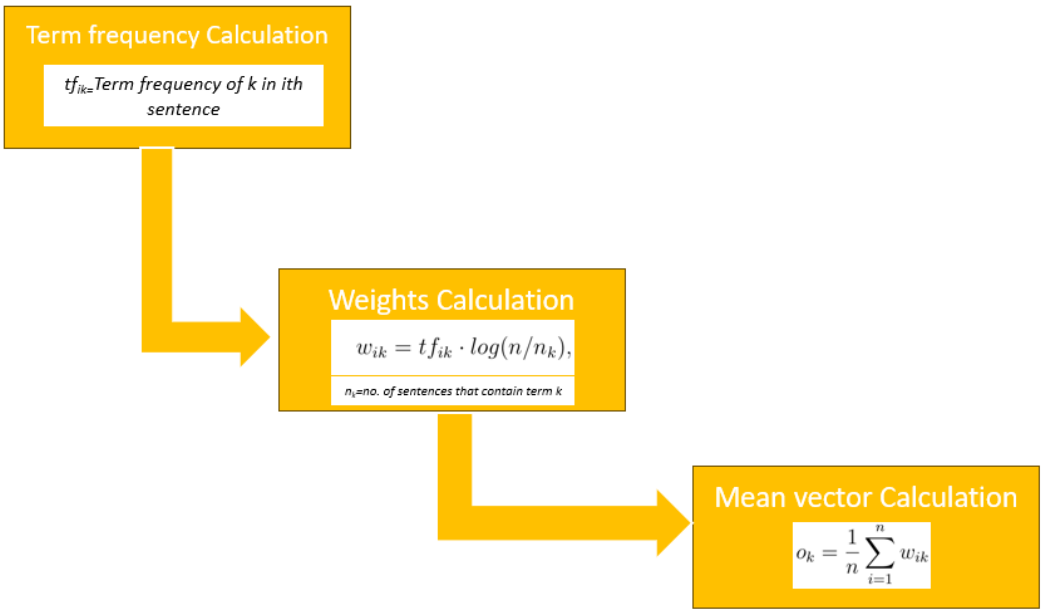
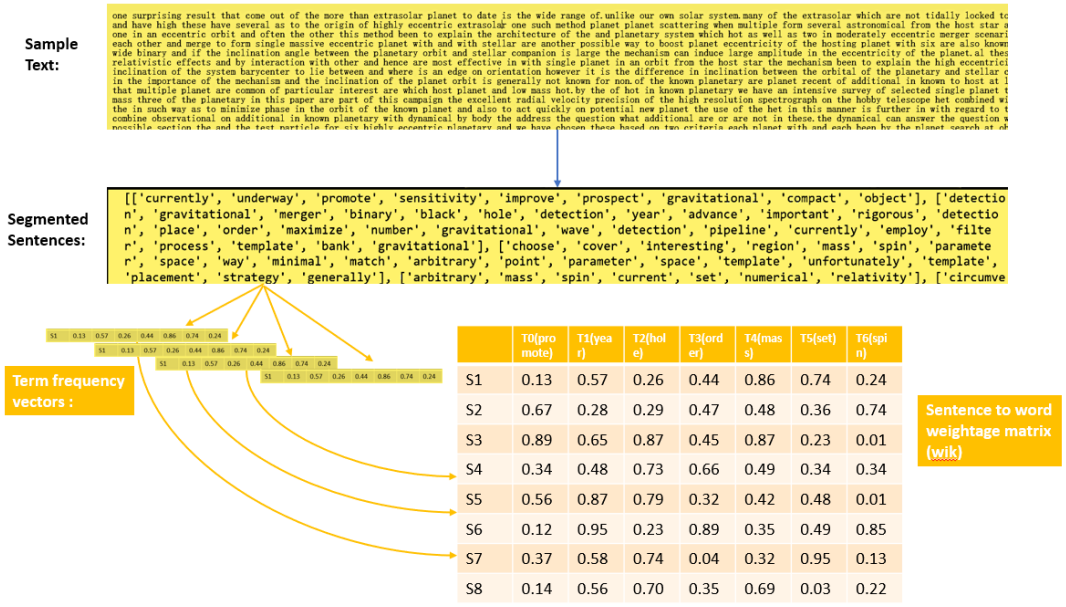


Fig 3.2 – Calculations involved in Sentence Representation.



	T0(promote)	T1(year)	T2(hole)	T3(order)	T4(mass)	T5(set)	T6(spin)
S1	0.13	0.57	0.26	0.44	0.86	0.74	0.24
S2	0.67	0.28	0.29	0.47	0.48	0.36	0.74
S3	0.89	0.65	0.87	0.45	0.87	0.23	0.01
S4	0.34	0.48	0.73	0.66	0.49	0.34	0.34
S5	0.56	0.87	0.79	0.32	0.42	0.48	0.01
S6	0.12	0.95	0.23	0.89	0.35	0.49	0.85
S7	0.37	0.58	0.74	0.04	0.32	0.95	0.13
S8	0.14	0.56	0.70	0.35	0.69	0.03	0.22

Fig 3.3 – Representation of sentences in term space

3.1.3 Similarity Measure:

The main role of a similarity measure is to quantify the similarity between any 2 numerical vectors. Set T consists of m distinct terms which is present in the document D, In the document, each sentence is represented as a vector $s_i = (w_{i1}, w_{i2}, \ldots, w_{im})$ in an m-dimensional space, here every element represents the weight of the associated term. The weight (w_{ik}) is calculated using Term Frequency and Inverse Sentence Frequency (TF-ISF) scheme, it is the combination of both Term Frequency and Inverse Term Frequency. The first element calculates how frequently a term occurs in a sentence while the second measure tells the number. of sentences that contain the specific term. Using this helps in finding out both the internal importance of the sentence and also the broader significance of the sentence in the entire document. A mean vector, or the average weights of the m words in T, can quantitatively summarize the primary content of the document or in other words it is the centre factor of the document represented in a vector form.

Cosine similarity of two sentences, depends upon the previously calculated weight values.

$$\text{sim}\left(s_i, s_j\right) = \frac{\sum_{k=1}^{m} w_{ik} w_{jk}}{\sqrt{\sum_{k=1}^{m} w_{ik}^2 \cdot \sum_{k=1}^{m} w_{jk}^2}}, \qquad i, j = 1,2, \ldots, n \tag{3.1}$$

- Here the similarity values of each sentence s_i with all the other sentences s_j are cached in a dictionary.
- The similarity value is calculated based on the expression shown in the equation (3.1)
- This is done to increase the computational speed.

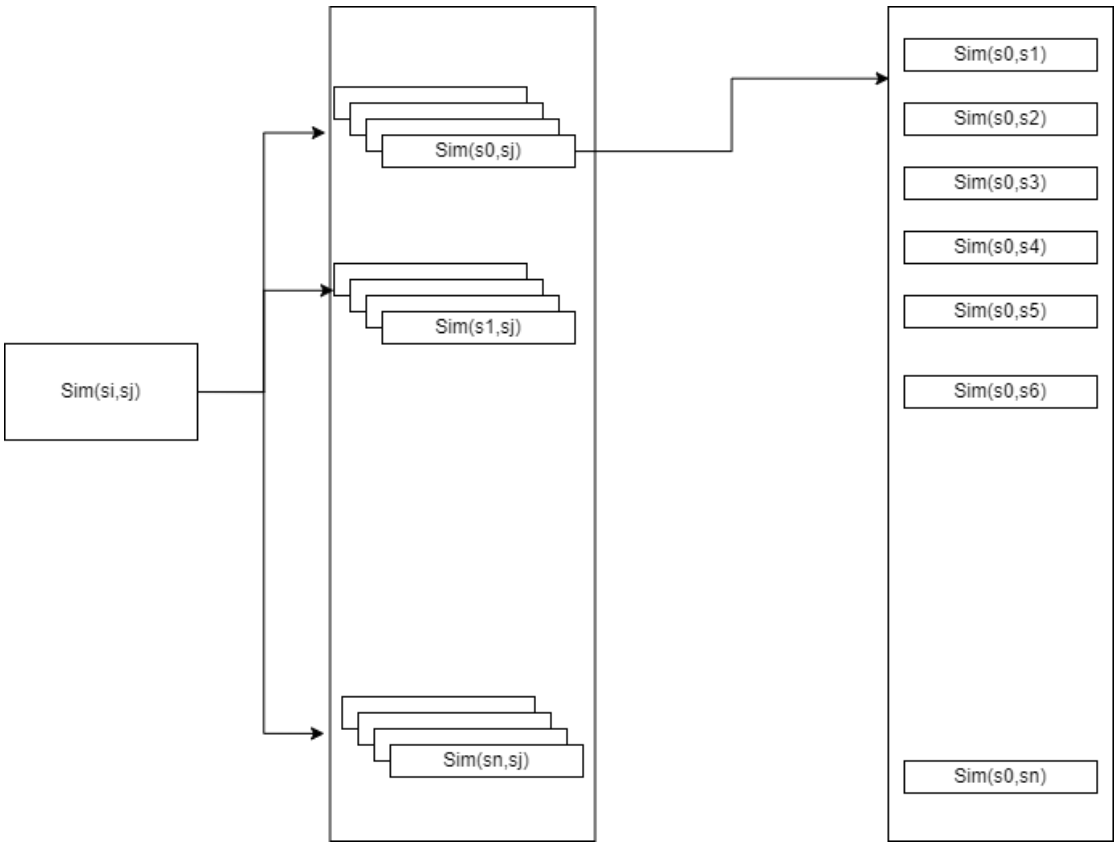


Fig 3.4 – Caching of cosine similarity values between various sentences

3.2 Multi-Objective Artificial Bee Colony Algorithm:

MOABC (Multi-Objective Artificial Bee Colony) is a multi-objective optimization algorithm inspired by the foraging behaviour of honey bees [1]. It extends the Artificial Bee Colony (ABC) algorithm, originally designed for single-objective optimization, to address problems with multiple conflicting objectives.

In MOABC, there are three types of bees: employee bees, onlooker bees, and scout bees. Employee bees explore the solution space to find food sources (candidate solutions) and share information with onlooker bees, who then decide which food sources to explore

based on a probability proportional to the quality of the sources. Scout bees search randomly for new food sources, allowing for broader exploration.

The below figure summarizes the major steps in MOABC. The following sections explain the details w.r.t each step-in detail.

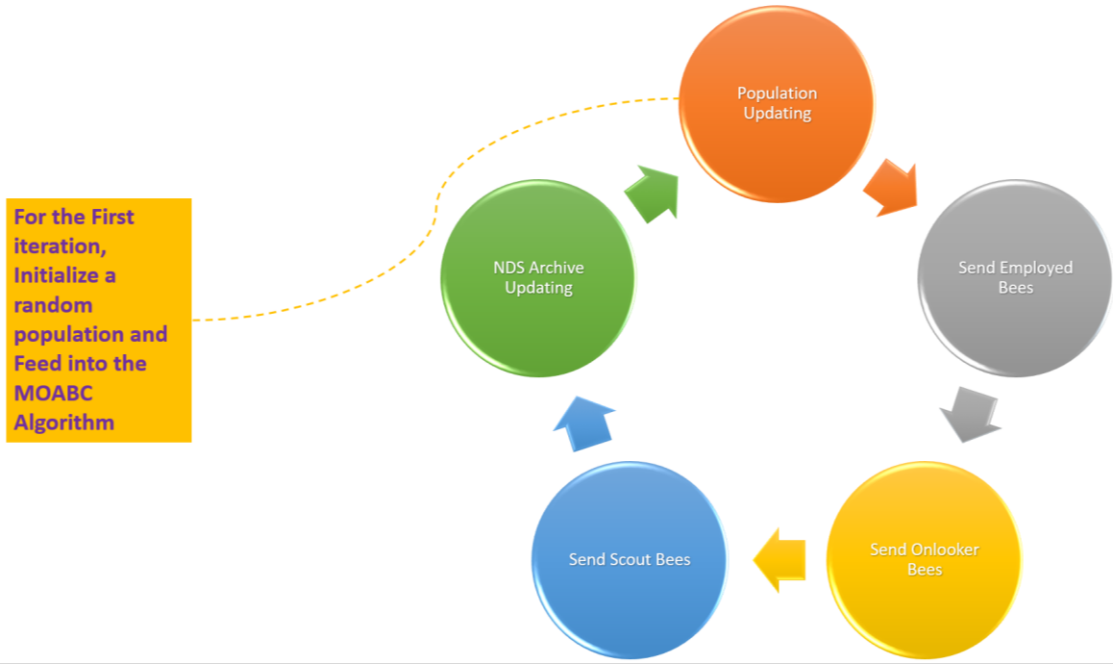


Figure 3.5 – Steps in MOABC

3.2.1 Colony Initialization:

As a first step in all optimization algorithms, the algorithm starts by initializing a random population of members, where each member is a potential solution to the problem. In this case the problem is generating an extractive summary of the document collection, and a solution is nothing but a vector consisting of 0's and 1's, where 0 represents that the sentence is not selected for summary, while 1 represents that the sentence is selected for the candidate summary. Thus, the number of bit positions of each member will be equal to the number. of sentences in the document collection. We are initializing 50 members randomly in the first iteration.

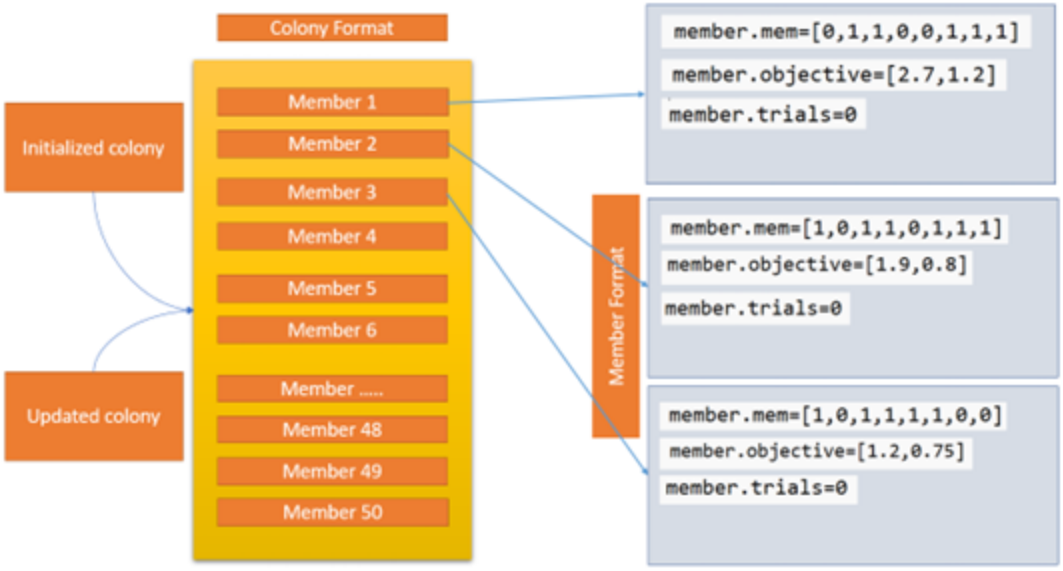


Figure 3.6 – Colony Initialization for MOABC

3.2.2: Sending Employee Bee phase:

In this phase, all members of the population are mutated. Then each mutated member is subject to a dominance criteria check. If the mutated member surpasses the original member in this check, it is replaced, else the original member is retained. The below figure summarizes the employed bee phase in MOABC.

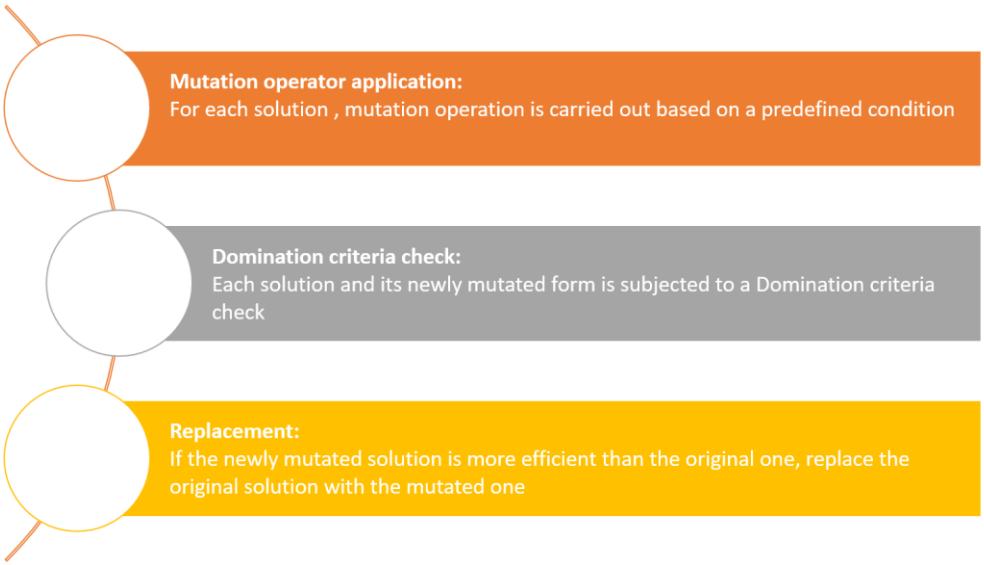


Fig 3.7 Steps in Employed Bee Phase

- **Mutation Operation:**
  - **Version-1:**

Here, mutation is done based on a criteria called the **"average sentence similarity check"** [2]**.** A candidate member has a list of sentences that are either included into the candidate summary or not. We take a random sentence, and its sentence similarity with the mean vector is computed. If the sentence is not included in the summary and if the sentence similarity with the mean vector is greater than the average sentence similarity, then the sentence is included (i.e the 0 becomes a 1). However, if it was lesser, than the sentence continues to be excluded from the summary. Similarly if the sentence was earlier included, but its similarity was less, then now it gets excluded (1 becomes a 0), and if it was higher, then it continues to be included in the candidate summary(1 remains as a 1). This operation and the similarity checks are explained in a detailed manner below.

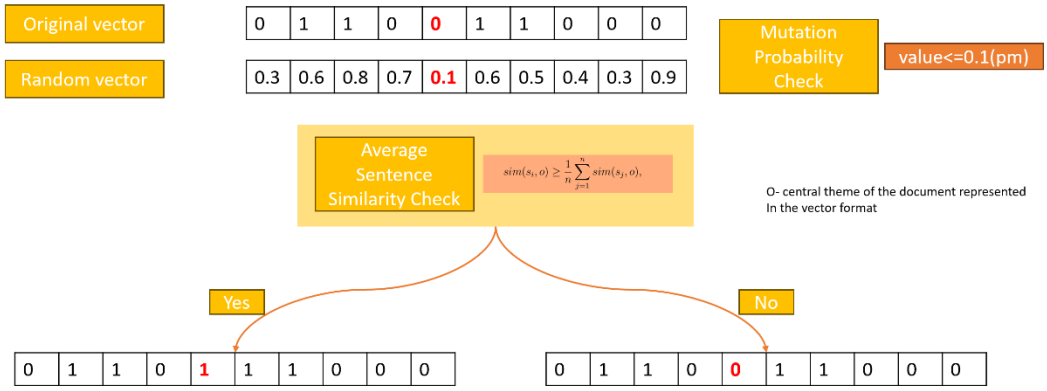


Fig 3.8 – Version 1 - Mutation process

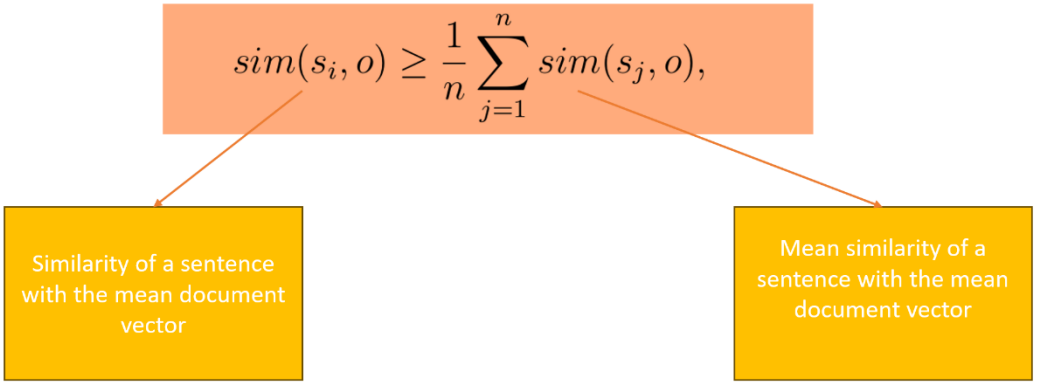


Fig 3.9– Average Sentence Similarity Check

- **Version-2:**

We first mutate X using Version 1 mutation operator. Then, we get a consensus score by passing the current NDS archive. i.e Once the NDS archive has been passed it takes all the members and identifies which member has voted for which sentence. This results in a sentence count dictionary. Then we sort the sentence counts in a descending manner, and take the top 10 voted sentences. We then use these voted sentences in a random manner as indicated in the algorithm previously stated.

The main difference between Version-2 and Version-1 arises in the **incorporation of the NDS archive**. The NDS archive contain the best set of non-dominated solutions obtained until the ith iteration. Considering its effect, might help in mutating the candidates in a slightly better manner.

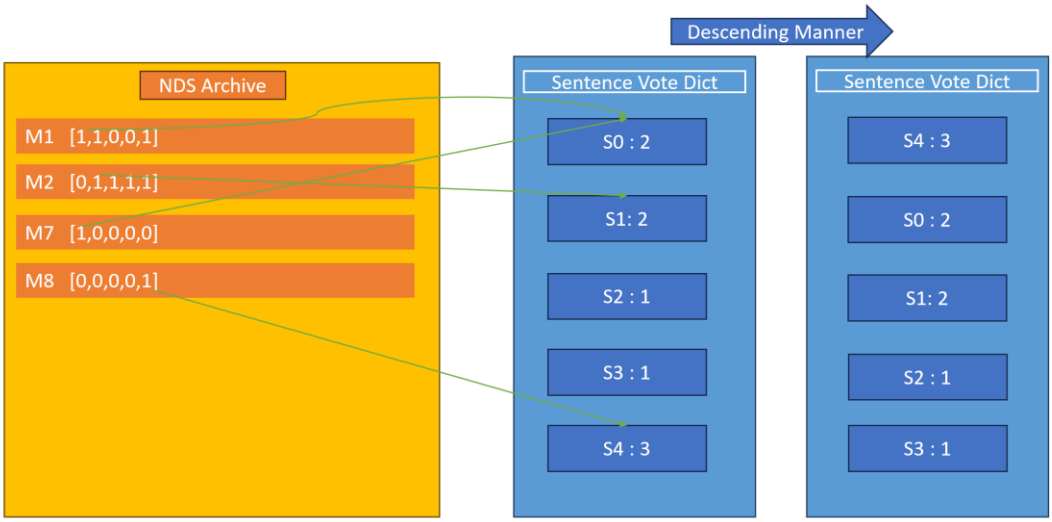


Fig 3.10 – NDS information updation in a voting dictionary

- **Replacement Operator:**
  - **Version-1 (Domination Criteria Check):**

    The dominance criteria is used to compare 2 members or objective vectors**.** In the below equation (3.2), x and y represent the solutions, and $f_i(x)$ and $f_i(y)$ represent the objective values for the i-th objective dimension.

Given two solutions x and y, solution x dominates solution y if and only if :

$$\forall i \in \{1,2,\dots,n\}: (f_i(x) \geq f_i(y)) \wedge \left(\exists j \in \{1,2,\dots,n\}: f_j(x) > f_j(y)\right) \qquad (3.2)$$

In our case, since we have 2 objectives, the objective vectors shall be of dimension = 2. Let the members be i and j, and their corresponding objective vectors be v_i and v_j .

The dominance criteria can return either of the 3 below outputs.

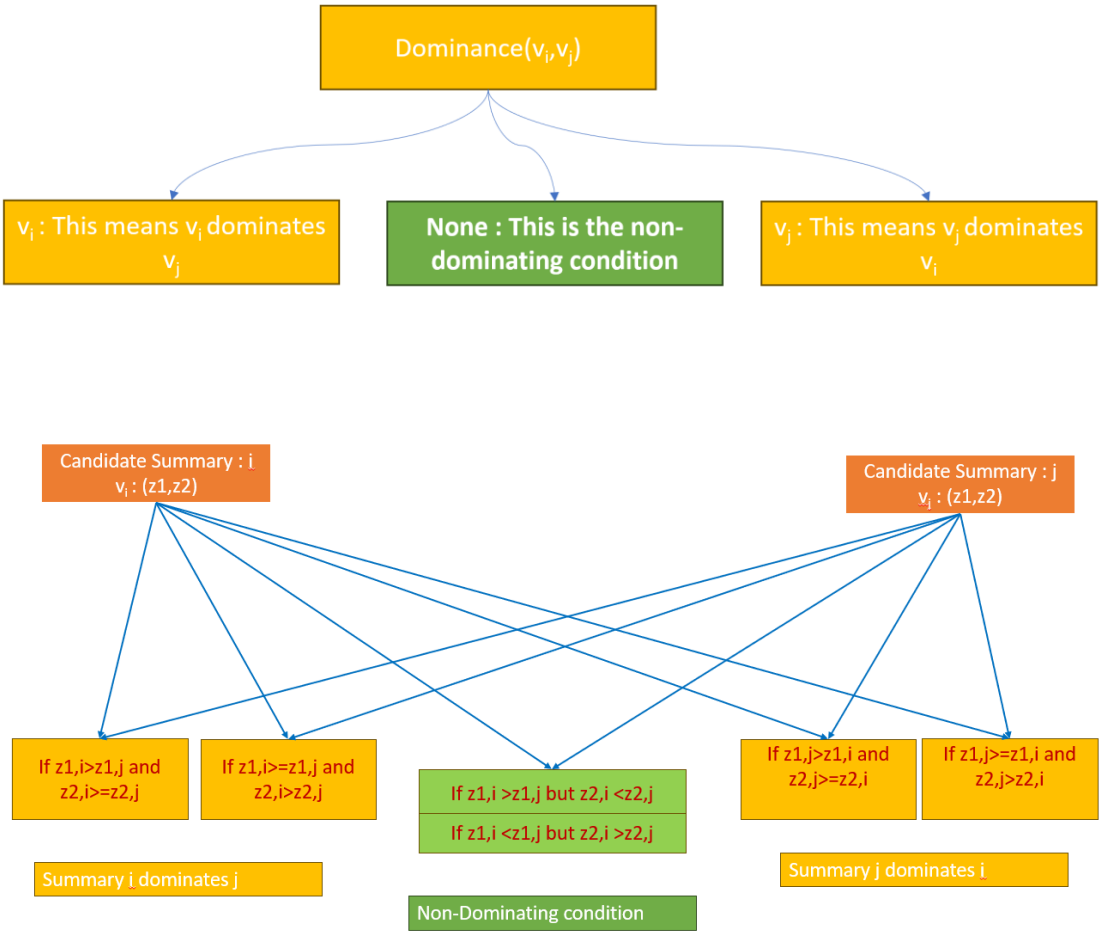


Fig 3.11 – 3.12 – Domination criteria check comparison.

➢ **Version-2 (Epsilon Dominance Motivated Replacement) :**

This version of Domination criteria check is **Epsilon Dominance Motivated Replacement.** Here, we find the percentage change and then check its percentage improvement(+ve) or deterioration(-ve)

- If the improvement > ϵ_1 or deterioration is > ϵ_2 (Where ϵ_1>=0 and ϵ_2<=0) then, it is accepted
- Else it is not accepted

The core idea is that:

- We need a minimum percentage improvement(ϵ_1) on certain objectives
- But at the same time cannot lose more than a max percentage deterioration(ϵ_2)
- If this is possible, then the solution is considered better (ϵ_1, ϵ_2) dominates. Else it is not considered good for comprehensive exploration.

Here, a solution x' can dominate x if it is:

- Not facing a considerable loss > ϵ_2
- Having a considerable improvement > ϵ_1
- No. of improvement objectives(winners) > n/2

Where e1 and e2 are threshold values of maximum improvement and deterioration accepted respectively

Algorithm Epsilon - Motivated Dominance Check

Data: Objective values x_i and x'_i, thresholds ϵ_1 and ϵ_2, number of objectives n
Result: Whether x' dominates x
Condition: $\epsilon_1 \geq 0$ and $\epsilon_2 \leq 0$

```
Initialize Boolean list List of length n with all elements set to False
Initialize variable winners = 0
for i ← 1 to n do
    Calculate v_i = (x'_i − x_i) / x_i
    if v_i ≥ 0 then
        if v_i ≥ ε_1 then
            List[i] ← True
            winners += 1
        end
    end
    else if v_i < 0 then
        if v_i ≥ ε_2 then
            List[i] ← True
        end
    end
end
if Any element in List is False or winners < floor(n/2) then
    x' does not dominate x
end
else if winners ≥ floor(n/2) then
    x' dominates x
end
```

- **Version-3 (Ranking and Crowding distance based replacement) :**

This version is a replacement operator based on the ranking and crowding. We incorporate this idea from NSGA and use it MOABC in a slightly modified format to see if this type of replacement criterion might be better. It is observed that **lower** ranked solutions are closer to optimality than higher ranked solutions.

Higher crowding distance solutions represent more variety and hence more preferred than lower crowding distance solutions.

Algorithm Replace Based on Rank and Crowding Distance

Data: List of members *members*, new member x'
1. Take the list of members and new member x' to a list;
2. Calculate the ranking and crowding distances of all the members;
3. Compare ranks and crowding distances to determine replacement position:;

```
for i ← 1 to length of members do
    if rank[x'] == rank[members[i]] then
        if cd[x'] > cd[members[i]] then
            Replace members[i] with x';
        else if cd[x'] < cd[members[i]] then
            Pass;
        else if cd[members[i]] == cd[x'] then
            Pass;
    else if rank[x'] < rank[members[i]] then
        Replace members[i] with x';
    else if rank[x'] > rank[members[i]] then
        Pass;
```

3.2.3: Sending Onlooker Bee phase:

Here, members are assigned probability values based on their ranking and crowding distance values. Then, a member is picked using roulette wheel selection. For the selection, the cumulative probability values are used. This member is then mutated and it then undergoes the dominance criteria checks as the previous, employed bee phase step. The replacement with the mutated member happens only if the mutated member dominates the original member. The below figure shows the steps in Onlooker Bee phase in detail.

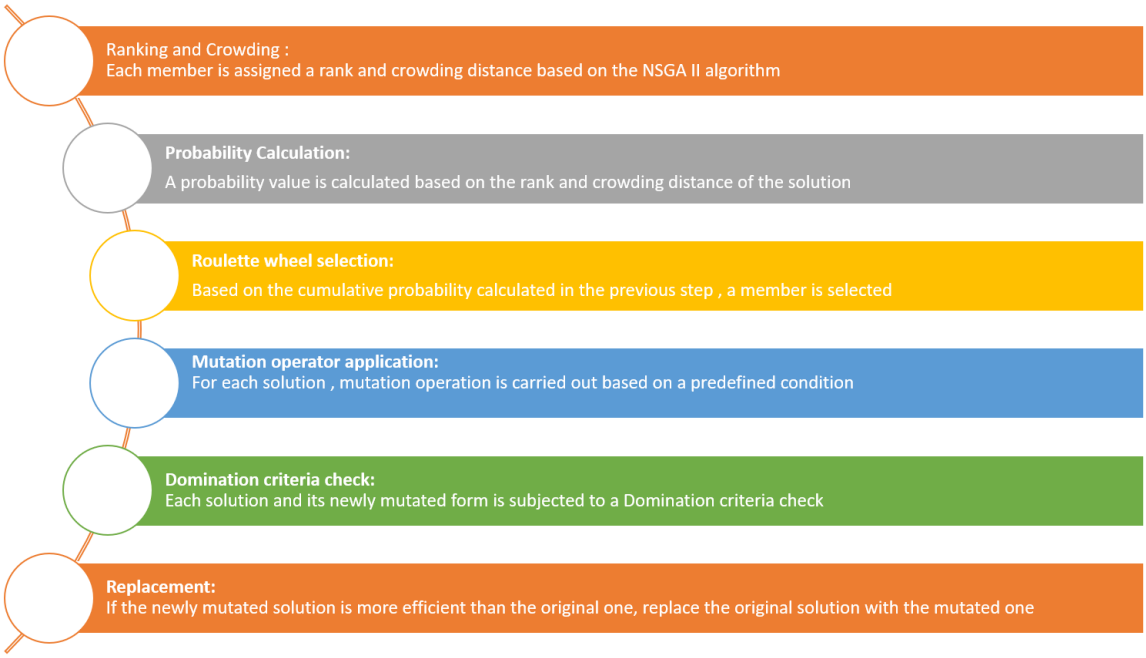


Fig 3.13 Steps in Onlooker Bee Phase

- **Ranking and Crowding distance:**
  - Each member is assigned a rank and crowding distance based on the NSGA II algorithm. In order to calculate ranking and crowding distances:
    - Calculate the objective values.
    - Perform fast non-dominated sorting algorithm.
    - Assign ranks based on the sorting algorithm.
    - Perform Crowding Distance calculations.
    - Assign fitness values based on rank and CD.

- **Fast Non-Dominated Sorting Algorithm:**
  - **Pareto Dominance:** A solution A is said to dominate solution B if A is at least as good as B in all objectives and strictly better in at least one objective. Solutions that are not dominated by any other solutions are considered non-dominated and belong to the first front.
  - **Non-dominated Fronts:** NSGA-II aims to partition the population into multiple fronts, where the first front contains non-dominated solutions, the second front contains solutions dominated only by solutions in the first front, and so on. This process continues until all solutions are assigned to fronts.

- **Efficient Sorting:** It performs this efficiently by iteratively evaluating the dominance relationships between solutions. It starts by initializing a set of solutions with no incoming dominance relationships (i.e. first front) and then iteratively identifies solutions that are dominated by the current front**.**
- **Ranking:** Evaluate each solution for both the objective values (In our case, Coverage and Redundancy). Assign rank to each solution using dominance depth. The strongly efficient candidates are ranked as 1, and the lesser strongly efficient candidates are ranked as 2 and so on. To increase the ranking speed, **fast non-dominated sorting algorithm** is used.

Stage 1 : Front 1 formation

Algorithm Fast Non-Dominated Sorting(P)

```
Stage-1
for each p ∈ P do
    S_p = ∅, n_p = 0
    for each q ∈ P do
        if (p ≺ q) then
            S_p = S_p ∪ q
        else if (q ≺ p) then
            n_p = n_p + 1
        end if
    end for
    if n_p = 0 then
        p_rank = 1
        F_1 = F_1 ∪ {p}
    end if
end for
```

Stage 2: Subsequent Front(s) formation

```
Stage-2
i = 1
while F_i ≠ ∅ do
    Q = ∅
    for each p ∈ F_i do
        for each q ∈ S_p do
            n_q = n_q − 1
            if n_q = 0 then
                q_rank = i + 1
                Q = Q ∪ {q}
            end if
        end for
    end for
    i = i + 1
    F_i = Q
end while
```

- **Crowding Distance (CD)** : CD is used to maintain **diversity** among solutions in the population. The intuition behind crowding distance is to encourage solutions to spread out evenly across the objective space. Crowding distance measures how crowded a solution is relative to its neighbors. The greater the crowding distance less crowded are the solutions, smaller the crowding distance more crowded are the solutions, thus less diverse. This diversity helps **prevent** the algorithm from **converging prematurely** to a single region of the objective space. Crowding distance is assigned for each solution in each front. Crowding distance across pareto fronts is meaningless. **Higher crowding distance** solutions represent more variety and hence more preferred than lower crowding distance solutions.
- The expression for Crowding Distance between any 2 members is shown below.

$$\text{CROWDING DISTANCE} = \frac{|f_m(i+1) - f_m(i-1)|}{f_m^{\max} - f_m^{\min}} \qquad (3.3)$$

Where $f_m(i+1)$ represents the objective value of the m^{th} dimension of the $(i+1)^{th}$ member. $fmax_m$ represents the maximum objective value in the pareto front in the m^{th} objective dimension. Similarly, $f_m min$ represents the minimum objective value in the pareto front in the m^{th} objective dimension.

The below figure shows the above expression in a graphical form.

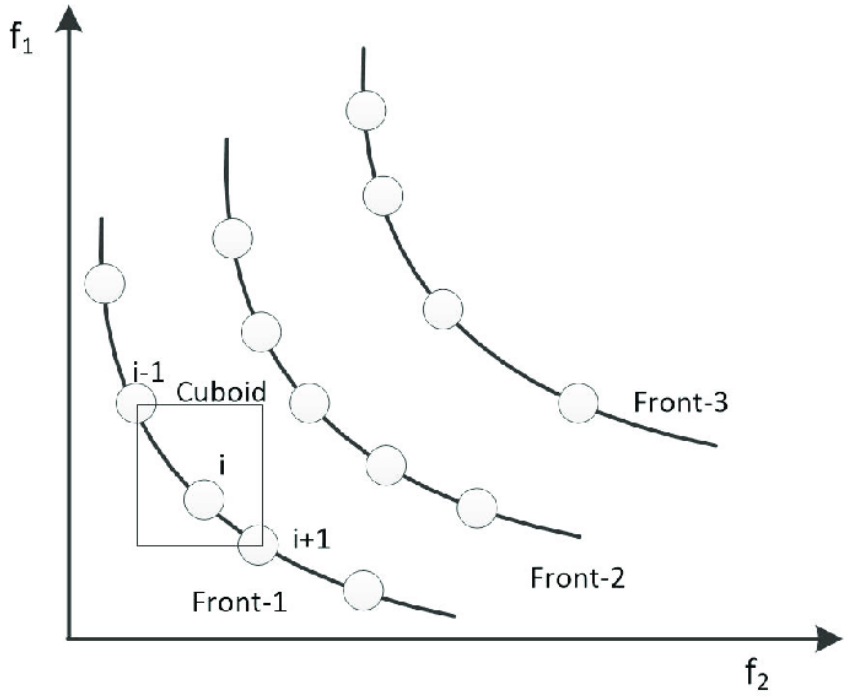


Fig 3.14 – Crowding Distance expression in graphical form

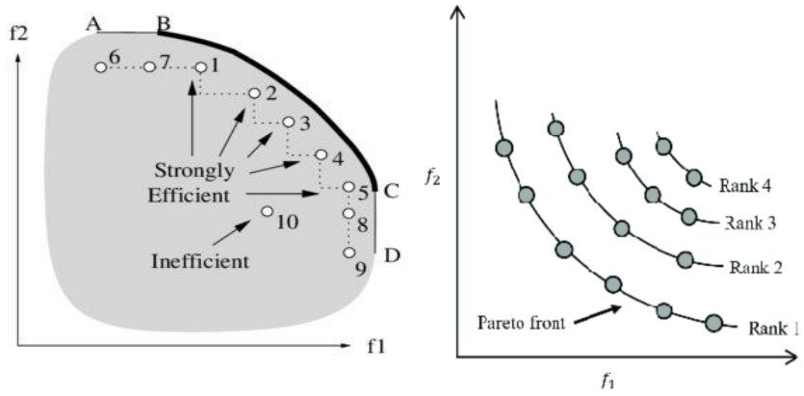


Fig 3.15 – 3.16 – Ranking and crowding distances in a pareto view

The crowding distance algorithm is shown below in detail.

Algorithm Crowding Distance (F)

$r = |F|$
for each $i \in F$ **do**
  $d_i = 0$
end for
for each objective m **do**
  Sort F by objective m
  $d_1 = d_r = \infty$
  for i from 2 to $r - 1$ **do**
    $d_i = d_i + \frac{f_{m+1} - f_{m-1}}{f_m^{\max} - f_m^{\min}}$
  end for
end for

- **Probability Calculation**:

  A probability value is calculated based on the rank and crowding distance of the solution. Formula for probability calculation:

$$MO_{\text{fitness}} = \frac{1}{\text{sol}\,[i]\cdot \text{rank} + \left(1 + \frac{1}{1 + \text{sol}\,[i]\cdot \text{crowding}}\right)} \quad (3.4)$$

$$\text{sol}\,[i] \cdot prob = (0.9 \times MO_{\text{fitness}}) + 0.1 \quad (3.5)$$

  The probability of selecting an individual is directly proportional to its fitness value **relative** to the total fitness of the population.

- **Roulette Wheel Selection:**

  Think of it as a roulette wheel where **each individual occupies a portion** of the wheel's circumference proportional to its fitness. Roulette Wheel Selection favors individuals with higher fitness values, but it still allows fewer fit individuals to have a chance of being selected. This helps **maintain diversity** in the population and prevents **premature convergence** to a suboptimal solution.

Algorithm Roulette Wheel Selection

Input: Individual probabilities of members
Output: Selected member from population

1. Calculate cumulative probabilities using the probabilities.
2. Generate a random number r in $U(0, 1)$.
3. Find the individual in the population that has cumulative probability just exceeding r.
4. Return that individual as the selected individual.

- **Mutation operation:**

  The mutation operator which was carried out in the employee bee phase is also carried out here.

- **Domination criteria check:**

  The domination criteria check which we have used in the previous stage has been employed here again.

- **Replacement Operator:**

  If the newly mutated solution is more efficient than the original one, replace the original solution with the mutated one.

3.2.4: Sending Scout Bee phase:

Here, the trial values of the members in the population are checked. If the member's trial value has exceeded a certain pre-defined limit, it means that there have been many tries to mutate the member and improve it, but the attempts have not being successful. This member is then taken and mutated several times (=10) and check if it can be replaced. If not, a random member is generated and replaced. The below figure that shows the steps in the scout bee phase in detail.

Fig 3.17 – Steps in Scout Bee Phase

- **Trial Check:**

  Each candidate contains a trial value which indicates the no. of unsuccessful mutation operators performed on it. For each candidate, the trial value is checked, if the trial value is greater than the specified threshold, then proceed otherwise skip this phase.

- **Mutation Operation:**

  For each solution, mutation operation is carried out based on a predefined condition, which we performed in the previous two phases.

- **Domination Criteria Check:**

  Each solution and its newly mutated form is subjected to a Domination criteria check.

- **Replacement:**

  If the newly mutated solution is more efficient than the original one, replace the original solution with the mutated one.

- **Random Replacement:**

  If no dominated solution is found that dominates the current solution, then a random solution is generated and replaced.

3.3 Non-Dominated Sorting Genetic Algorithm:

NSGA (Non-dominated Sorting Genetic Algorithm) is a type of multi-objective evolutionary algorithm designed to solve optimization problems involving multiple conflicting objectives [22]. It uses a population-based approach where a set of candidate solutions evolves over generations through processes like selection, crossover, and mutation.

A key feature of NSGA is its use of a non-dominated sorting technique to classify solutions into different levels or fronts based on their dominance relationships. Solutions in the first front are non-dominated, while those in subsequent fronts are dominated by one or more solutions from earlier fronts. NSGA also incorporates a crowding distance mechanism to maintain diversity among solutions, ensuring that the algorithm explores a wide range of the solution space. This makes NSGA suitable for complex optimization problems where trade-offs between different objectives must be considered. The below figure elucidates the steps in NSGA-II algorithm in detail.

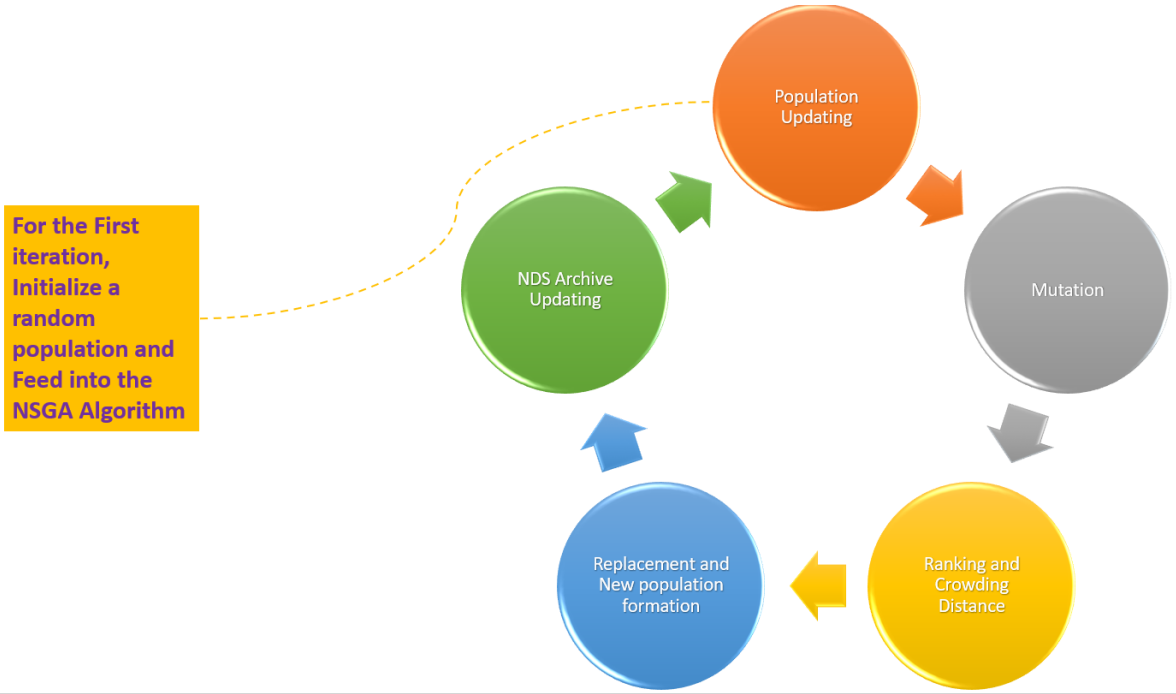


Fig 3.18 – Steps involved in NSGA-II algorithm

3.3.1 Mutation Operation:

For each solution, mutation operation is carried out based on Version-1 Mutation operation defined earlier.

3.3.2 Ranking and Crowding Distance:

Ranking and Crowding distances are calculated again for each solution as we did in MOABC.

3.3.3 Binary Crowded Tournament Selection Operator:

Basically, here we take any 2 members from the combination of parents and children (binary tournament). Then we compare their ranks and crowding distances. Among the 2 whoever wins, enters into new population. New population is returned.

Algorithm Binary Crowded Tournament Selection Operator (BCTSO)

```
Data: parent_members, child_members
Combine parent_members and child_members into list_of_members
Calculate front information for each member in list_of_members
Initialize an empty list new_population
for i from 1 to length of parent_members do
    Randomly select two members from list_of_members
    Compare their front ranks
    if Front ranks are different then
        Select the member with the lower rank
    else
        if Front ranks are the same then
            Compare their crowding distances
            Select the member with the greater crowding distance
        end if
    end if
    Add the selected member to new_population
end for
return new_population
```

3.4 Pareto Front and NDS Archive:

This step is common for both MOABC and NSGA algorithms. During each iteration optimal solutions are updated in a pareto front.

- **Pareto Front:**

  There are multiple conflicting objectives that one needs to optimize simultaneously. The Pareto front consists of solutions that achieve the best trade-offs between these conflicting objectives. A solution is considered Pareto optimal if there is no other solution that is better in all objectives. The Pareto front, then, represents the boundary of the objective space where no further improvement in one objective can be made without sacrificing another objective. A sample pareto front is shown in the below diagram.

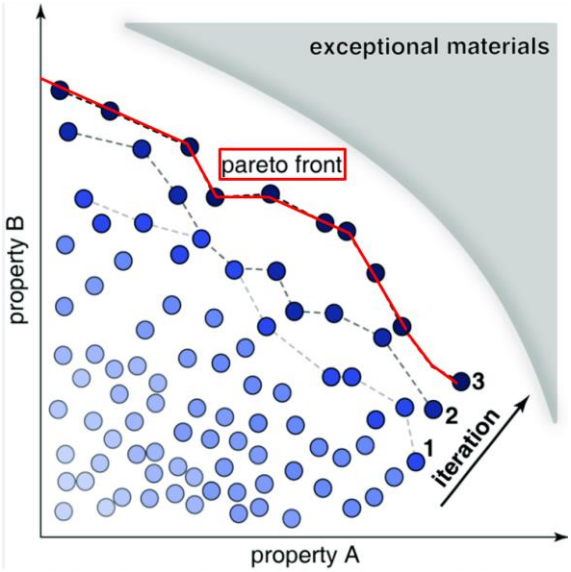


Fig 3.19 – Sample Pareto Front

- **NDS Archive:**

  Obtained sets of non-dominated solutions from the population are added to the NDS archive. The addition of a member (or a deletion) is done by the dominance criteria check. For each iteration, NDS archive remains persistent and is updated.

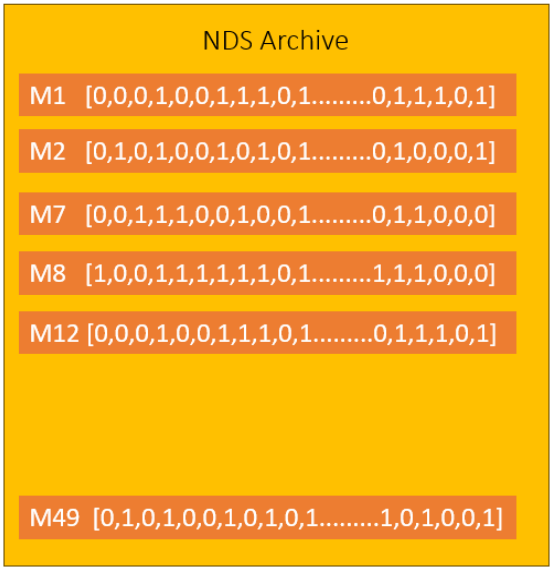


Fig 3.20 – NDS Archive

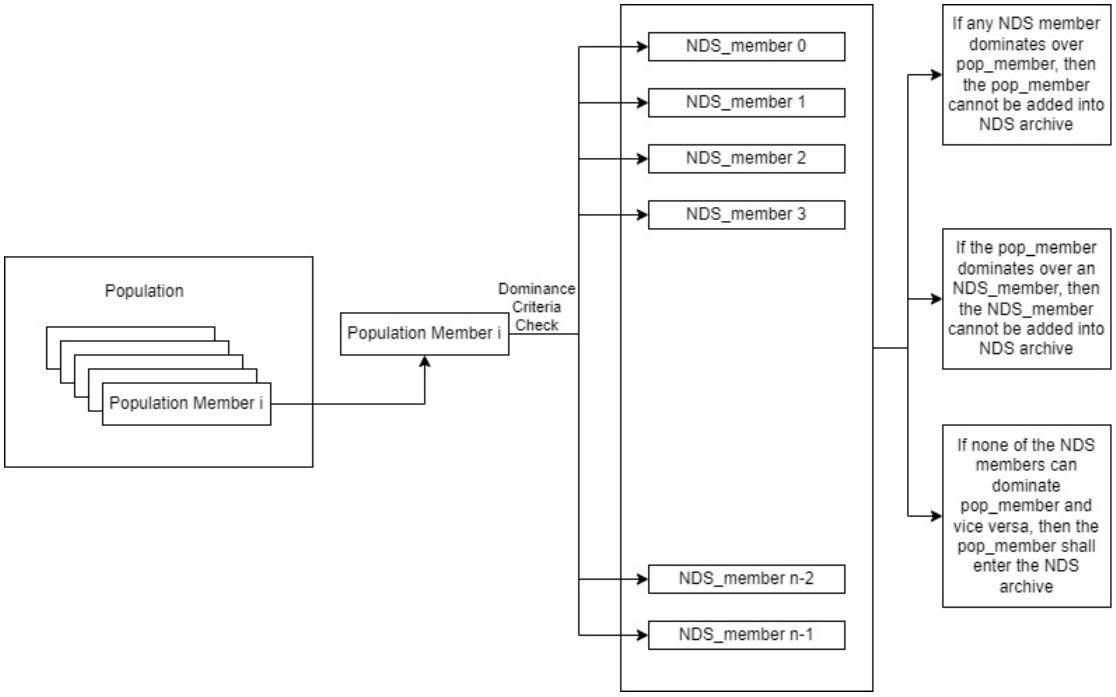


Fig 3.21– NDS Insert Function

3.5 Single Solution Extraction:

This step is common for both MOABC and NSGA algorithms. It involves extraction of a single best solution from all the optimal solutions present in the pareto front. There are 13 methods that can be used to extract a single solution from pareto front [21]. Each method is experimented and assessed for it's performance.

Fig 3.22 – Various Single Solution Extraction Methods Experimented

3.5.1 Consensus:

All the NDS archive members are taken. Each member would have certain sentences that would be included in its summary, and some that are not. A sentence rank dictionary is maintained, and each archive member s' vote is taken into account. This dictionary is then ranked in a descending order based on the votes. The sentences that are voted the most, are included one-by-one until the word limit is extinguished. The summary that is formed by these "top-most-voted" sentences is then returned. The below algorithm explains this in detail.

Algorithm Consensus Solution

```
sentences ← ∅
for point ← 1 to points_max do
    sentences ← sentences ∪ get_sentences(point)
end for
sentences ← sort_sentences_by_frequency(sentences)
consensusSolution ← ∅
consensusLength ← 0
n ← 1
while consensusLength + sentence_length(n) < L + ϵ do
    consensusSolution ← consensusSolution ∪ sentences(n)
    consensusLength ← consensusLength + sentence_length(n)
    n ← n + 1
end while
return consensusSolution
```

3.5.2 Largest Hypervolume:

The concept of hypervolume in this context refers to the space occupied by the objective function values of non-dominated solutions on the Pareto front. In a two-dimensional scenario, this measure represents the area covered by the points on the Pareto front. The algorithm selects the point on the Pareto front that has the largest hypervolume (referred to as LH).

The below algorithm outlines the pseudocode for this approach. Initially, the algorithm selects the first point on the Pareto front as the best, storing its corresponding hypervolume. Then, it checks all remaining points. If a point's hypervolume is greater than the current best, the best point and its stored hypervolume are replaced. At the end of this process, the summary for the selected best point is returned.

Algorithm Largest Hypervolume

```
bestPoint ← get_point(1)
largestHypervolume ← calculate_hypervolume(1)
for point ← 2 to points_max do
    if largestHypervolume ≤ calculate_hypervolume(point) then
      bestPoint ← get_point(point)
      largestHypervolume ← calculate_hypervolume(point)
   end if
end for
return get_summary(bestPoint)
```

3.5.3 Topmost Coverage:

The members are ranked according to their coverage values. The member that has the highest coverage value is considered as the summary and then returned.

Algorithm Summary with Highest Coverage Values

```
members ← sort_by_coverage(members)
summary ← members[0]
summaryLength ← member_length(members[0])
return summary
```

3.5.4 Topmost Redundancy Reduction:

The members are ranked according to their redundancy reduction values(objective 2). The member that has the highest redundancy reduction value is considered as the summary and then returned.

Algorithm Summary with Highest Redundancy Reduction Values

```
members ← sort_by_redundancy_reduction(members)
summary ← members[0]
summaryLength ← member_length(members[0])
return summary
```

3.5.5 Shortest Distance to Ideal Point:

Here, first all members' objective vectors are normalized in the (0,1) range. Then, a distance metric is chosen. The metrics that we have considered here are : Euclidean, Chebyshev, Manhattan and Mahalanobis. The distance between the normalized objective vector of the member and the ideal point (1,1) is measured. The ideal point is (1,1) since, both objectives are of maximization type. The member that has the shortest distance is taken as the summary and returned accordingly.

Algorithm Shortest Distance to the Ideal Point

```
bestPoint ← get_point(1)
shortestDistanceIdeal ← calculate_distance_to_ideal(1)
for point ← 2 to points_max do
    if shortestDistanceIdeal ≥ calculate_distance_to_ideal(point) then
        bestPoint ← get_point(point)
        shortestDistanceIdeal ← calculate_distance_to_ideal(point)
    end if
end for
return get_summary(bestPoint)
```

The distances are specified in the table 3.1. In table 3.1, S represents the covariance and P,Q represent the sets of sentences corresponding to their summaries.

3.5.6 Shortest Distance to All points:

Here, the distance b/w a point and all other points are measured. These distances are summed up. Whichever member(point) has the least accumulated distance is selected as the summary and is returned. The algorithm below explains the procedure in detail. The distance measures used here are : Euclidean, Chebyshev, Manhattan, Mahalanobis and Levenshtein.

Algorithm Shortest Distance to All Points

```
bestPoint ← get_point(1)
shortestDistanceAll ← calculate_distance_to_all(1)
for point ← 2 to points_max do
    if shortestDistanceAll ≥ calculate_distance_to_all(point) then
        bestPoint ← get_point(point)
        shortestDistanceAll ← calculate_distance_to_all(point)
    end if
end for
return get_summary(bestPoint)
```

Distance Measure	Expression
Euclidean	$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$
Mahalanobis	$\sqrt{(P_2 - P_1)^\top S^{-1}(P_2 - P_1)}$
Chebyshev	$\max\left(\lvert x_2 - x_1\rvert, \lvert y_2 - y_1\rvert\right)$
Manhattan	$\lvert x_2 - x_1\rvert + \lvert y_2 - y_1\rvert$
Levenshtein	$\lvert P\rvert + \lvert Q\rvert - 2 \cdot \lvert P \cap Q\rvert$

Table 3.1 – Table of distance measures and expressions

3.6 Objective Expression

Here, we choose the following 2 objectives:

- Coverage – This conveys numerically, how much the candidate summary has covered the various topics conveyed in the original full text.
- Redundancy – This conveys numerically, how many sentences conveying similar ideas/topics are present i.e detecting the presence of redundant sentences.

Coverage and redundancy are conflicting objectives [3, 5].

If one needs maximum coverage, one shall have many redundant sentences in their document. At the same time, if one needs to have the least set of redundant sentences present, it shall lead to certain amount of coverage loss.

The expressions are defined below.

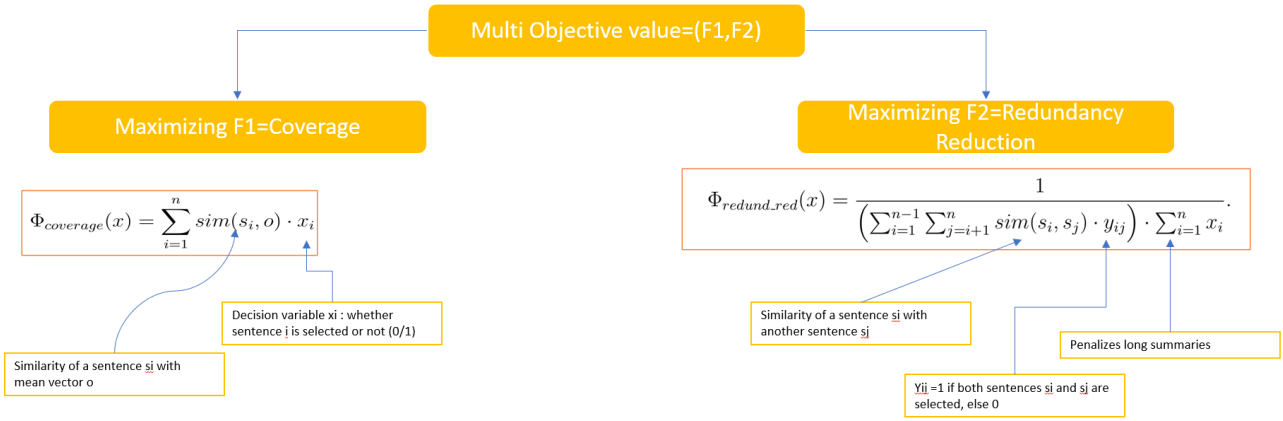


Fig 3.23 – Objectives Expression

3.7 Sentence Scoring Schemes:

A member if it is illegal, i.e it is does not satisfy the length constraints, it needs to be repaired. If a member is above the word limit, then certain sentences need to be removed and similarly if it is below the word limit, certain sentences need to be added. The sentences are ranked and added/removed based on a ranking/scoring scheme.

Here, we first explore the TF-ISF scoring scheme suggested in previous literature [14]. The TF-ISF based scoring scheme is explained in detail in the below algorithm.

Input: Sentences in document collection, $d = s_1, s_2, .., s_n$
Output: Vector of sentence score, $O = o_1, o_2, .., o_n$

```
Initialize Frequency Table from d, FreqTable
for i=0 to NoOfTerms do
    for j=0 to NoOfSentences in d do
    tfi← FreqTable or 0
    Oj+ ← tfi
  end for
end for
for i=0 to NoOfSentences in d do
  Oj← Oj/10
end for
return O
```

In order to improve the scoring scheme, we also incorporate LexRank, TextRank and Graph Based Extractive Text Summarizer (GETS) ranking.

- TextRank is a graph-based algorithm for text summarization and keyword extraction, leveraging sentence relationships to determine importance [13].
- LexRank, a variant of TextRank, further refines this approach by considering lexical similarity among sentences to improve summarization quality [12].
- Graph based extractive text summarizer (GETS), uses cosine similarities on a graph sentence network and takes those sentences that cross a threshold [11].

These rankings are used as they have textual extraction capabilities that can improve the summarization capability and quality of summaries. The rankings are used/selected in a random order to preserve randomness.

Algorithm Select and Return Scores

```
function SELECT_AND_RETURN_SCORES(doc_info)
    scores ← {}
    score_types ← ["textrank","tf","graph","lexrank"]
    score_type_chosen ← random.choice(score_types)
    if score_type_chosen = "tf" then
        scores ← doc_info.scores_details.tf_scores
    else if score_type_chosen = "textrank" then
        scores ← doc_info.scores_details.textrank_scores
    else if score_type_chosen = "graph" then
        scores ← doc_info.scores_details.graph_scores
    else if score_type_chosen = "lexrank" then
        scores ← doc_info.scores_details.lexrank_scores
    end if
    return scores
end function
```

3.8 Sentence Repair Scheme:

Based on the scores calculated in previous step, sentences are accordingly removed or added based on the candidate member s' calculated summary length. The process is explained in the below figure.

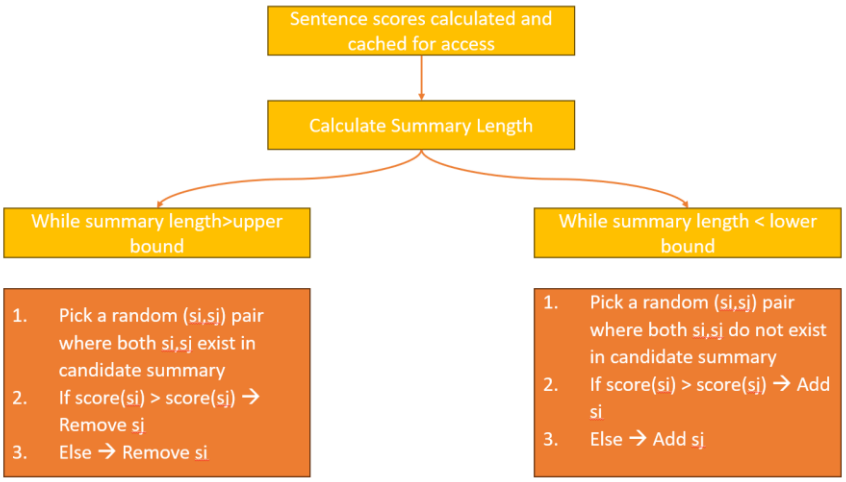
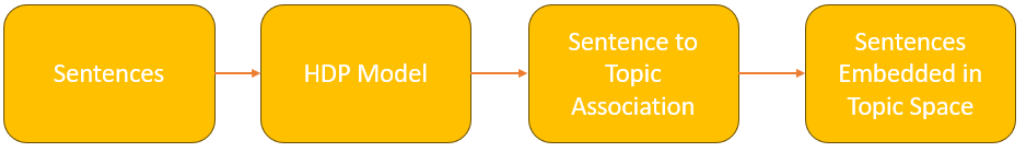


Fig 3.24 – Sentence Repair Scheme

3.9 Topic Modelling:

We have also integrated the knowledge of topics present in the document to bring about some improvements in MOABC algorithms [10]. Instead of representing the sentences in term space, we are representing it in topic sentence [17]. For this we are using Hierarchical Dirichlet Process (HDP). HDP is a non-parametric Bayesian method used for topic modeling. It extends the Dirichlet Process (DP) to allow for an infinite number of mixture components, making it suitable for tasks where the number of topics is unknown or could potentially be very large. In LDA, the no. of topics needed to be provided beforehand, whereas this is not needed in HDP [15, 16].



	Topic0	Topic1	Topic2	Topic3
S0	0.72	0.61	0.07	0.52
S1	0.44	0.12	0.29	0.06
S2	0.90	0.75	0.63	0.8
S3	0.64	0.46	0.13	0.44
S4	0.04	0.56	0.37	0.75

Note that now the sentences are embedded in topic space(instead of term space). Here, the no. of topics is returned by HDP model.
Each sentence vector dimension shall equal no. of topics

Fig 3.25 - Topic Modelling using Hierarchical Dirichlet Process (HDP)

3.10 Evaluation Metrics:

ROUGE scores are heavily used among researchers both implementing and assessing the text summarization approaches. ROUGE-N determines a single overall N-gram overlapping scores between a generated sentence and reference sentences, whereas ROUGE-L considers the longest common subsequence, which is the community of shared words between two compared texts with no repetitions. These parameters provide a rating how closely the machine-created description of the page and the human-written description of the page match each other.

ROUGE-1 measures the overlap of unigrams (single words) between a candidate text and a reference text. It is a simple measure of recall indicating how much of the reference text is contained within the candidate text.

ROUGE-2 calculates the overlap of bigrams (two consecutive words) between the candidate text and the reference text. This metric provides an indication of how well the candidate text retains the structure of the reference text, considering adjacent word combinations.

ROUGE-L focuses on the longest common subsequence (LCS) between the candidate and reference texts. It captures structural similarity and accounts for in-order word sequences, without the strict adjacency required by ROUGE-2.

The expressions for the fmeasure(f-score) is given below.

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.6}$$

3.11 Overall Process Flow:

To summarize, we perform the following steps that are explained in the diagram also.

1. Preprocess the text document.
2. Convert sentences to numerical vectors.
3. Initialize the population and run the Multi-Objective Evolutionary Computing algorithm.
4. Extract the final pareto front.
5. From the pareto front, choose a single solution extraction method.
6. Use the single solution, compare it with the gold-standard extract and compute the ROUGE score.

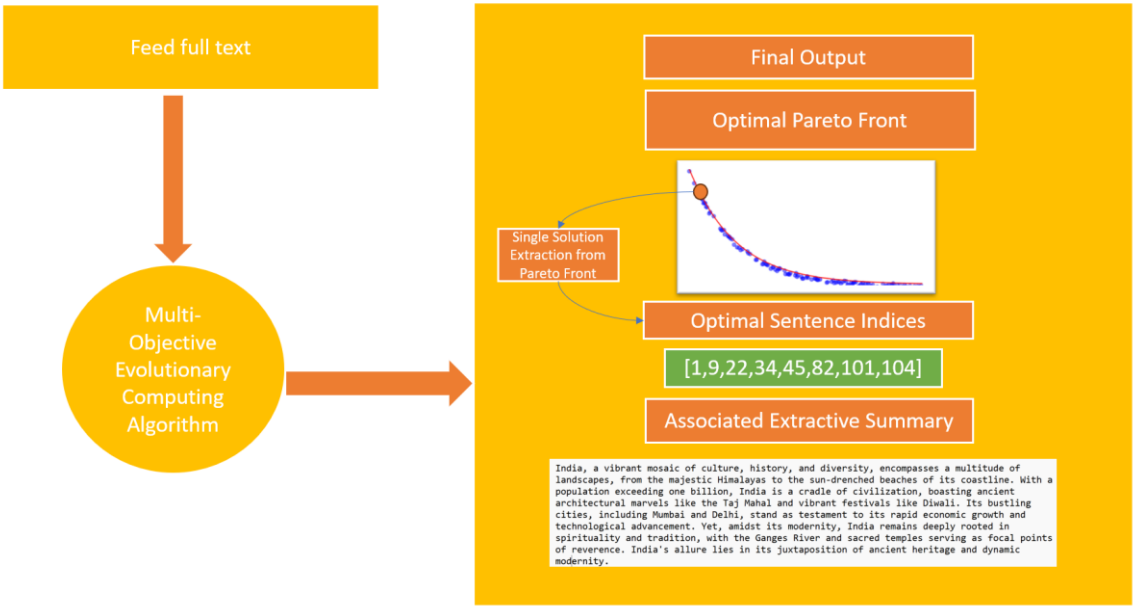


Fig 3.26 – Overall process flow in ETS

3.12 Parameters:

We use various parameters in our model. We summarize them below in the following table.

Parameter Name	Value
No. of generations	100
No. of independent runs	10
P_m (Mutation Parameter)	0.4
Population Size	50
Trial Cutoff Limit(Trial Check)	5
L	200
Epsilon	50

Table 3.2 - Parameters

CHAPTER – 4

RESULTS

4.1 Experiments:

4.1.1 Comparison of Performance of Single Solution Extraction methods:

This experiment is to find which extraction method is the best to extract the solution from the pareto front**.**

	v1			v2		
Single Solution Extraction Method	**Rouge-1**	**Rouge-2**	**Rouge-L**	**Rouge-1**	**Rouge-2**	**Rouge-L**
Consensus	**0.488003**	**0.207657**	**0.24613**	**0.490004**	**0.206773**	**0.242533**
Largest Hypervolume	0.433383	0.131932	0.201392	0.442169	0.144581	0.215188
Topmost Coverage	0.459804	0.178171	0.230284	0.462031	0.185829	0.235045
Topmost Redundancy Reduction	0.425854	0.121665	0.195965	0.436459	0.137917	0.20899
Ideal Euclidean	0.467883	0.172147	0.22317	0.477185	0.187835	0.230624
Ideal Manhattan	0.450335	0.151914	0.217248	0.450239	0.157928	0.220122
Ideal Chebyshev	0.475961	0.179695	0.230374	0.470444	0.178726	0.228513
Ideal Mahalanobis	0.457092	0.158584	0.211604	0.472495	0.175473	0.227154
All Chebyshev	0.468927	0.17297	0.225292	0.482124	0.189614	0.232476
All Manhattan	0.467964	0.171004	0.223779	0.486161	0.193111	0.239116
All Mahalanobis	0.46934	0.169296	0.227714	0.475477	0.177737	0.228766
All Euclidean	0.467513	0.167787	0.223637	0.477436	0.182022	0.230812
All Levenshtein	0.476481	0.185901	0.231284	0.488128	0.201269	0.239669

Table 4.1 – Comparison of different Mutation Operator Versions with Single Solution Extraction methods

	MOABC			NSGA		
Single Solution Extraction Method	**Rouge-1**	**Rouge-2**	**Rouge-L**	**Rouge-1**	**Rouge-2**	**Rouge-L**
Consensus	**0.488003**	**0.207657**	**0.24613**	**0.477775**	**0.191733**	0.238035
Largest Hypervolume	0.433383	0.131932	0.201392	0.444965	0.145234	0.213816
Topmost Coverage	0.459804	0.178171	0.230284	0.46325	0.178817	0.227706
Topmost Redundancy Reduction	0.425854	0.121665	0.195965	0.44206	0.141879	0.211273
Ideal Euclidean	0.467883	0.172147	0.22317	0.461458	0.162065	0.223547
Ideal Manhattan	0.450335	0.151914	0.217248	0.462307	0.168243	0.224203
Ideal Chebyshev	0.475961	0.179695	0.230374	0.475285	0.18886	**0.239015**
Ideal Mahalanobis	0.457092	0.158584	0.211604	0.460684	0.165726	0.225548
All Chebyshev	0.468927	0.17297	0.225292	0.464569	0.167418	0.216745
All Manhattan	0.467964	0.171004	0.223779	0.466226	0.170115	0.21807
All Mahalanobis	0.46934	0.169296	0.227714	0.465786	0.169212	0.219253
All Euclidean	0.467513	0.167787	0.223637	0.463335	0.166639	0.219111
All Levenshtein	0.476481	0.185901	0.231284	0.472801	0.18274	0.23488

Table 4.2 – Comparison between MOABC and NSGA with Single Solution Extraction methods

	Dominate			EpsilonDominate			RCR		
Single Solution Extraction Method	**Rouge-1**	**Rouge-2**	**Rouge-L**	**Rouge-1**	**Rouge-2**	**Rouge-L**	**Rouge-1**	**Rouge-2**	**Rouge-L**
Consensus	**0.487442**	**0.205303**	0.241226	**0.491977**	**0.217469**	**0.253821**	**0.492276**	**0.210495**	**0.247404**
Largest Hypervolume	0.432769	0.12633	0.195346	0.433308	0.13352	0.201359	0.437531	0.133266	0.20658
Topmost Coverage	0.456353	0.174335	0.227231	0.453223	0.170191	0.225285	0.466758	0.190847	0.232288
Topmost Redundancy Reduction	0.4276	0.123843	0.197399	0.429202	0.128493	0.199728	0.431921	0.12836	0.202126
Ideal Euclidean	0.460974	0.16348	0.218466	0.457247	0.155786	0.215846	0.461275	0.161244	0.224303
Ideal Manhattan	0.446017	0.149977	0.205254	0.444524	0.149592	0.208755	0.453208	0.161226	0.219934
Ideal Chebyshev	0.481413	0.191841	0.240614	0.474248	0.179686	0.235874	0.477146	0.186126	0.236002
Ideal Mahalanobis	0.455925	0.154982	0.214323	0.452878	0.146942	0.208262	0.45667	0.153117	0.216118
All Chebyshev	0.479972	0.190442	0.236587	0.474176	0.17514	0.22242	0.47804	0.185392	0.2343
All Manhattan	0.483602	0.197549	0.239999	0.475086	0.181089	0.228752	0.474207	0.179213	0.227666
All Mahalanobis	0.473976	0.180781	0.231238	0.475638	0.189074	0.231246	0.481706	0.183105	0.232318
All Euclidean	0.472279	0.177632	0.226525	0.472027	0.179416	0.226239	0.482911	0.188737	0.235976
All Levenshtein	0.486761	0.201122	**0.243859**	0.478519	0.188146	0.236702	0.483564	0.194652	0.236882

Table 4.3 – Comparison of Various Replacement Operators with Single Solution Extraction methods

	TFISF			SBERT		
Single Solution Extraction Method	**Rouge-1**	**Rouge-2**	**Rouge-L**	**Rouge-1**	**Rouge-2**	**Rouge-L**
Consensus	**0.488003**	**0.207657**	**0.24613**	**0.461448**	**0.187128**	**0.227898**
Largest Hypervolume	0.433383	0.131932	0.201392	0.432274	0.135638	0.199969
Topmost Coverage	0.459804	0.178171	0.230284	0.435896	0.152273	0.214098
Topmost Redundancy Reduction	0.425854	0.121665	0.195965	0.429873	0.130904	0.198952
Ideal Euclidean	0.467883	0.172147	0.22317	0.435168	0.135389	0.202526
Ideal Manhattan	0.450335	0.151914	0.217248	0.431005	0.136156	0.199472
Ideal Chebyshev	0.475961	0.179695	0.230374	0.445909	0.141701	0.207671
Ideal Mahalanobis	0.457092	0.158584	0.211604	0.438702	0.145521	0.204281
All Chebyshev	0.468927	0.17297	0.225292	0.451782	0.160861	0.218211
All Manhattan	0.467964	0.171004	0.223779	0.446752	0.155868	0.216721
All Mahalanobis	0.46934	0.169296	0.227714	0.448955	0.157877	0.220785
All Euclidean	0.467513	0.167787	0.223637	0.44911	0.157663	0.218263
All Levenshtein	0.476481	0.185901	0.231284	0.449796	0.157097	0.215789

Table 4.4 – Comparison of TFISF and SBERT with Single Solution Extraction methods

	v1			v2			v3			v4		
Single Solution Extraction Method	**Rouge-1**	**Rouge-2**	**Rouge-L**	**Rouge-1**	**Rouge-2**	**Rouge-L**	**Rouge-1**	**Rouge-2**	**Rouge-L**	**Rouge-1**	**Rouge-2**	**Rouge-L**
Consensus	0.456564	0.179032	**0.24228**	0.461272	0.172836	0.231789	**0.488003**	**0.207657**	**0.24613**	0.462249	0.17939	0.221719
Largest Hypervolume	0.455179	0.163846	0.221929	0.44691	0.146533	0.209237	0.433383	0.131932	0.201392	0.454155	0.160589	0.208676
Topmost Coverage	0.458444	**0.183214**	0.24029	0.455767	**0.173598**	0.227677	0.459804	0.178171	0.230284	0.459374	0.184019	0.234319
Topmost Redundancy Reduction	0.423967	0.128389	0.199834	0.42323	0.110758	0.189627	0.425854	0.121665	0.195965	0.435521	0.140511	0.209492
Ideal Euclidean	0.450201	0.16045	0.217946	0.447788	0.147693	0.211911	0.467883	0.172147	0.22317	0.46495	0.174729	0.224421
Ideal Manhattan	0.447103	0.153131	0.210197	0.433143	0.131261	0.200671	0.450335	0.151914	0.217248	0.46389	0.174318	0.2201
Ideal Chebyshev	0.456275	0.165886	0.228742	0.455947	0.159248	0.225121	0.475961	0.179695	0.230374	0.458064	0.162217	0.217375
Ideal Mahalanobis	0.457602	0.180554	0.235365	0.452946	0.160138	0.220601	0.457092	0.158584	0.211604	0.469698	0.183519	0.231863
All Chebyshev	0.45682	0.168102	0.229081	0.456618	0.157103	0.227654	0.468927	0.17297	0.225292	0.466811	0.182602	0.229629
All Manhattan	0.457582	0.169622	0.226216	0.458996	0.162972	0.230868	0.467964	0.171004	0.223779	0.471193	0.186826	0.23071
All Mahalanobis	**0.462706**	0.178862	0.232693	**0.462051**	0.170484	**0.233751**	0.46934	0.169296	0.227714	**0.476038**	**0.194277**	**0.234492**
All Euclidean	0.458757	0.169185	0.231581	0.457684	0.160296	0.228498	0.467513	0.167787	0.223637	0.474507	0.190195	0.231441
All Levenshtein	0.452001	0.160997	0.223304	0.456291	0.159638	0.220572	0.476481	0.185901	0.231284	0.454776	0.16342	0.217417

Table 4.5 – Comparison of Topic Modelling versions with Single Solution Extraction methods

It is observed that, “consensus” single solution extraction method from pareto front is the best method as it reflects at a higher ROUGE score in comparison to other methods.

4.1.2 Other Experiments:

1. Exp1: Mutation Operators: This experiment compares the performance of the various mutation operators (Version 1 and Version 2) .
2. Exp2: MOABC vs NSGA: This experiment compares the performance of MOABC and NSGA, 2 state-of-the-art multi-objective optimization algorithms.
3. Exp3: Replacement Operators: The replacement operator in any MOEA plays a crucial role. This experiment compares replacement operators on performance.
4. Exp4: Sentence Representation: This experiment tries to change the sentence representations and embedding to see if there is an improvement.
5. Exp5: Topic Modelling: This experiment compares the various versions of topic modelling proposed previously.

4.1.2.1 Comparison of Mutation Operators:

	v1			v2		
Document ID	**Rouge-1**	**Rouge2**	**Rouge-L**	**Rouge-1**	**Rouge2**	**Rouge-L**
d061j	0.551657	0.260877	0.272444	0.570783	0.298663	0.285493
d062j	0.431267	0.152089	0.199343	0.408765	0.093617	0.180944
d063j	0.451138	0.202795	0.216988	0.454979	0.21975	0.211262
d064j	0.425557	0.174111	0.245297	0.436432	0.170562	0.232912
d065j	0.519318	0.309747	0.359557	0.513404	0.299694	0.351334
d066j	0.4142	0.071273	0.17564	0.425344	0.070964	0.180189
d067f	0.523673	0.232305	0.267432	0.531485	0.233507	0.253942
d068f	0.619635	0.365537	0.309796	0.605184	0.344679	0.284687
d069f	0.453829	0.105145	0.186789	0.447611	0.102435	0.182829
d070f	0.489754	0.202696	0.228016	0.506052	0.233865	0.261737
	0.488003	**0.207657**	**0.24613**	**0.490004**	0.206773	0.242533

Table 4.6 – Mutation Operators Comparison across various documents

Here, we observe that Mutation Operator Version 2 performs slightly better than Version 1 in terms of ROUGE1 score, whereas it underperforms in case of Rouge2 and RougeL scores.

4.1.2.2 Comparison of Algorithms:

	MOABC			NSGA		
Document ID	Rouge-1	Rouge-2	Rouge-L	Rouge-1	Rouge-2	Rouge-L
d061j	0.551657	0.260877	0.272444	0.5327	0.260506	0.2702
d062j	0.431267	0.152089	0.199343	0.421115	0.122121	0.196113
d063j	0.451138	0.202795	0.216988	0.43643	0.17574	0.201363
d064j	0.425557	0.174111	0.245297	0.410522	0.139943	0.20208
d065j	0.519318	0.309747	0.359557	0.478116	0.247531	0.320793
d066j	0.4142	0.071273	0.17564	0.419623	0.077725	0.183548
d067f	0.523673	0.232305	0.267432	0.53563	0.264299	0.2798
d068f	0.619635	0.365537	0.309796	0.597453	0.328951	0.319857
d069f	0.453829	0.105145	0.186789	0.458077	0.102865	0.184673
d070f	0.489754	0.202696	0.228016	0.488088	0.197652	0.22192
	0.488003	**0.207657**	**0.24613**	0.477775	0.191733	0.238035

Table 4.7 – Comparison of both MOABC and NSGA algorithms over various documents

Here, we observe that MOABC outperforms NSGA in all 3 scoring methodologies.

4.1.2.3 Comparison of different replacement operators:

	Dominate			EpsilonDominate			RCR		
Document ID	Rouge-1	Rouge-2	Rouge-L	Rouge-1	Rouge-2	Rouge-L	Rouge-1	Rouge-2	Rouge-L
d061j	0.55272	0.29005	0.27781	0.55400	0.28680	0.28250	0.58071	0.34048	0.31546
d062j	0.44144	0.16579	0.21115	0.41913	0.12400	0.18274	0.46136	0.18850	0.22598
d063j	0.45703	0.20889	0.19440	0.46017	0.22777	0.21704	0.45607	0.22332	0.21999
d064j	0.44099	0.17967	0.24330	0.43176	0.15366	0.22258	0.44050	0.17420	0.25695
d065j	0.49562	0.28173	0.34960	0.51293	0.28782	0.34046	0.51438	0.30188	0.35057
d066j	0.41092	0.06859	0.17019	0.42033	0.07375	0.18205	0.42602	0.06847	0.17132
d067f	0.51880	0.21419	0.25537	0.53099	0.25073	0.26578	0.54257	0.24548	0.29174
d068f	0.60606	0.32829	0.28381	0.63537	0.38061	0.31271	0.59958	0.32943	0.28441
d069f	0.45432	0.10258	0.18393	0.45446	0.10191	0.18345	0.45306	0.10285	0.18650
d070f	0.49652	0.21324	0.24272	0.49231	0.20027	0.23285	0.48330	0.20165	0.22218
	0.48744	0.20530	0.24123	0.49115	0.20873	0.24222	**0.49575**	**0.21763**	**0.25251**

Table 4.8 – Comparison of different versions of Replacement Operators over various documents

Here, we observe that RCR based replacement performs better in Rouge1,2 and L than Dominate and EpsilonDominate methods.

4.1.2.4 Comparison of TFISF vs SBERT Sentence Representation:

	TFISF			SBERT		
Document ID	Rouge-1	Rouge-2	Rouge-L	Rouge-1	Rouge-2	Rouge-L
d061j	0.551657	0.260877	0.272444	0.540018	0.299693	0.356535
d062j	0.431267	0.152089	0.199343	0.385774	0.070138	0.168506
d063j	0.451138	0.202795	0.216988	0.425208	0.146583	0.170417
d064j	0.425557	0.174111	0.245297	0.459097	0.195037	0.265027
d065j	0.519318	0.309747	0.359557	0.464201	0.214963	0.251056
d066j	0.4142	0.071273	0.17564	0.482584	0.239276	0.306825
d067f	0.523673	0.232305	0.267432	0.476661	0.226256	0.191814
d068f	0.619635	0.365537	0.309796	0.451438	0.146679	0.179315
d069f	0.453829	0.105145	0.186789	0.458764	0.195129	0.192586
d070f	0.489754	0.202696	0.228016	0.470736	0.137523	0.196902
	0.488003	**0.207657**	**0.24613**	0.461448	0.187128	0.227898

Table 4.9 – Comparison of TFISF vs SBERT Sentence Representation over various documents

Here, we observe that using a more complex and representative transformer like SBERT does not yield any improvements in terms of ROUGE scores.

4.1.2.5 Comparison of Different Topic modelling Versions:

This experiment is to observe the effect of the various topic modelling algorithms on coverage score:

i. Coverage – Topic Knowledge **Enabled,** Redundancy reduction – Topic Knowledge **Enabled.**
ii. Coverage – Topic Knowledge **Enabled,** Redundancy reduction – Topic Knowledge **not Enabled.**
iii. Coverage – Topic Knowledge **not Enabled,** Redundancy reduction – Topic Knowledge **not Enabled.**

iv. Coverage – Topic Knowledge **not Enabled,** Redundancy reduction – Topic Knowledge **Enabled.**

Algorithm Topic Modelling Version Objective Vector Retrieval

Input: Version Number
Output: Objective Vector Definition
if Version == V1 **then**
  Return [*Coverage_Topic*, *Redundancy_Reduction_Topic*]
else if Version == V2 **then**
  Return [*Coverage_Topic*, *Redundancy_Reduction*]
else if Version == V3 **then**
  Return [*Coverage*, *Redundancy_Reduction*]
else if Version == V4 **then**
  Return [*Coverage*, *Redundancy_Reduction_Topic*]
end if

	v1			v2			v3			v4		
Document ID	v1 R1	v1 R2	v1 RL	v2 R1	v2 R2	v2 RL	v3 R1	v3 R2	v3 RL	v4 R1	v4 R2	v4 RL
d061j	0.490855	0.200422	0.256568	0.480962	0.177603	0.239983	0.551657	0.260877	0.272444	0.554496	0.327477	0.304413
d062j	0.405463	0.109369	0.198901	0.42615	0.121244	0.172391	0.431267	0.152089	0.199343	0.414817	0.111445	0.200867
d063j	0.422723	0.161335	0.209004	0.428155	0.142192	0.197412	0.451138	0.202795	0.216988	0.472572	0.220135	0.204916
d064j	0.404394	0.126168	0.181489	0.426592	0.146874	0.223509	0.425557	0.174111	0.245297	0.389721	0.121259	0.173326
d065j	0.453827	0.197573	0.266948	0.491439	0.227168	0.266563	0.519318	0.309747	0.359557	0.422357	0.157846	0.239523
d066j	0.409426	0.097689	0.208448	0.405764	0.124247	0.205008	0.4142	0.071273	0.17564	0.413225	0.076971	0.178291
d067f	0.495544	0.209356	0.25398	0.517685	0.255226	0.287039	0.523673	0.232305	0.267432	0.531238	0.281905	0.264046
d068f	0.582364	0.344546	0.371033	0.528717	0.258819	0.309875	0.619635	0.365537	0.309796	0.536354	0.229338	0.233252
d069f	0.427792	0.116045	0.192514	0.443475	0.101383	0.18917	0.453829	0.105145	0.186789	0.43255	0.10499	0.199008
d070f	0.473253	0.227822	0.283909	0.463782	0.173601	0.226942	0.489754	0.202696	0.228016	0.455156	0.162538	0.219546
	0.456564	0.179032	0.24228	0.461272	0.172836	0.231789	**0.488003**	**0.207657**	**0.24613**	0.462249	0.17939	0.221719

Table 4.10 – Comparison of Different Topic modelling Versions over various documents

Here, we observe that topic modelling does not yield to any improvement in ROUGE scores, since V3 represents the version that does not incorporate any topic modelling in the objective and still outperforms other methods using topic modelling.

4.2 Inferences and Results:

	Rouge-1	Rouge-2	Rouge-L
V1-Topic	0.456564	0.179032	0.24228
V2-Topic	0.461272	0.172836	0.231789
V3-Topic	0.488003	0.207657	0.24613
V4-Topic	0.462249	0.17939	0.221719
With-SBERT	0.461448	0.187128	0.227898
eDominate	0.491977	**0.217469**	**0.253821**
RCR	**0.492276**	0.210495	0.247404
NSGA	0.477775	0.191733	0.238035
V2-Mutation	0.490004	0.206773	0.242533

Table 4.11 – Consolidated Performance Measure of all Experiments performed

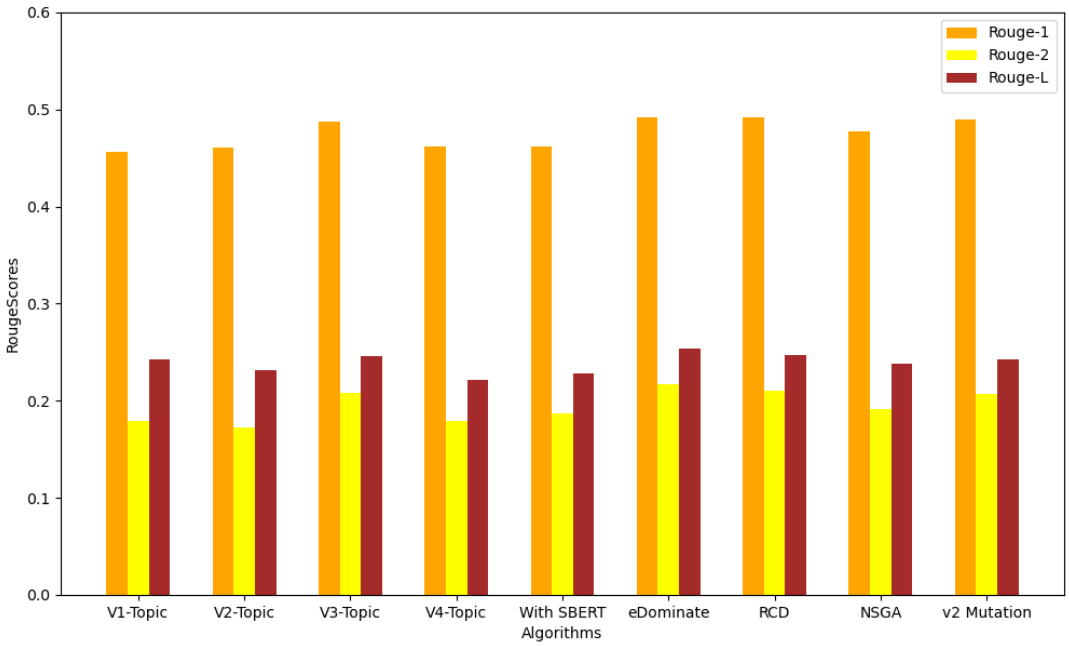


Fig 4.1 - Consolidated Performance Measure of all Experiments performed

We observe that RCR takes a high ROUGE-1 score, whereas epsilon-dominate takes a high ROUGE-2 and ROUGE-L score when compared to the other algorithms.

4.2.1 Contemporary Methods Comparison:

It is observed that our newly proposed methods have slightly better ROUGE-1 scores than the methods proposed previously in literature.

However, in terms of ROUGE-2, the algorithm's performance is poor when compared to SummaRunner.

	Rouge-1	Rouge-2
COSUM	0.49083	0.23092
ESDS-GHS-GLO	0.47903	0.22142
MA-SingleDocSum	0.48280	0.22840
DE	0.46694	0.12368
UnifiedRank	0.48487	0.21462
FEOM	0.46575	0.12490
NetSum	0.44963	0.11167
CRF	0.44006	0.10924
QSC	0.44865	0.18766
SVM	0.43235	0.10867
Manifold Ranking	0.43235	0.10677
NN-SE	0.47400	0.23200
SummaRuNNer	0.45400	**0.23900**
eDominate	**0.49198**	0.21747
v2-Mutation	**0.49000**	0.20677
v3-Topic	0.48800	0.20766
RCR	**0.49228**	0.21050

Table 4.12 – Comparison of Our proposed Method with Contemporary Methods

4.3 Trends in Pareto Front and Objective

We observe in the below figure that the pareto front moves towards the optimal pareto front as the iterations increase from 0 to 100.

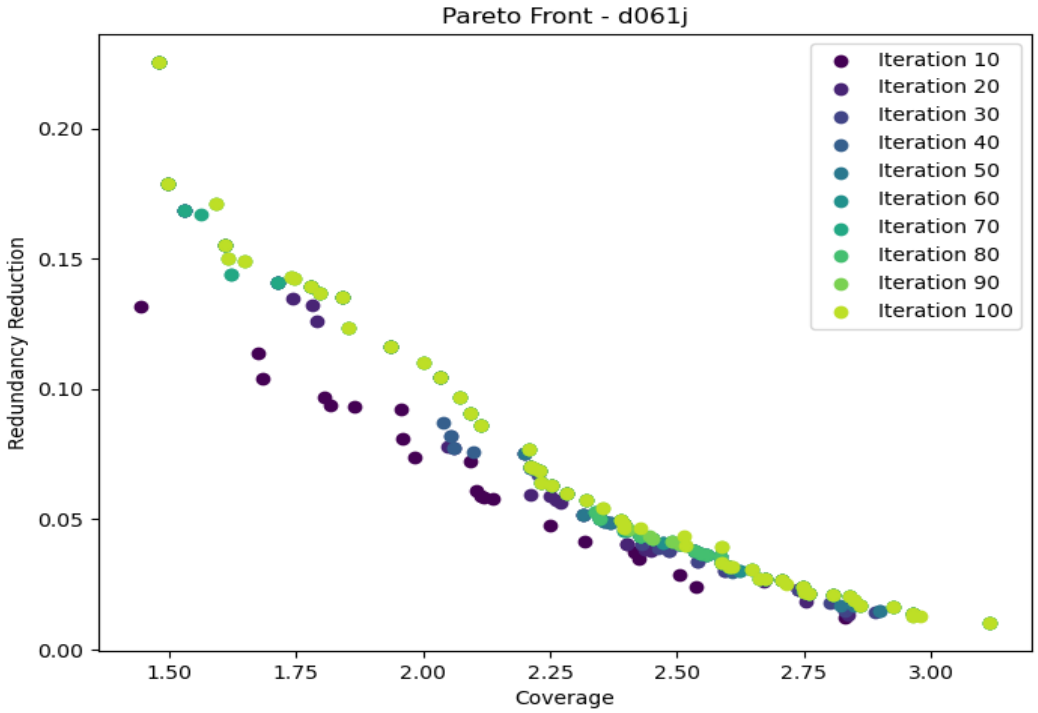


Fig 4.2 - Pareto Front Trend

From the below Fig 4.5, we see that as coverage increases, redundancy reduction decreases which confirms that the 2 objectives are indeed conflicting. It also reflects a trend wherein, as iterations proceed, coverage increases.

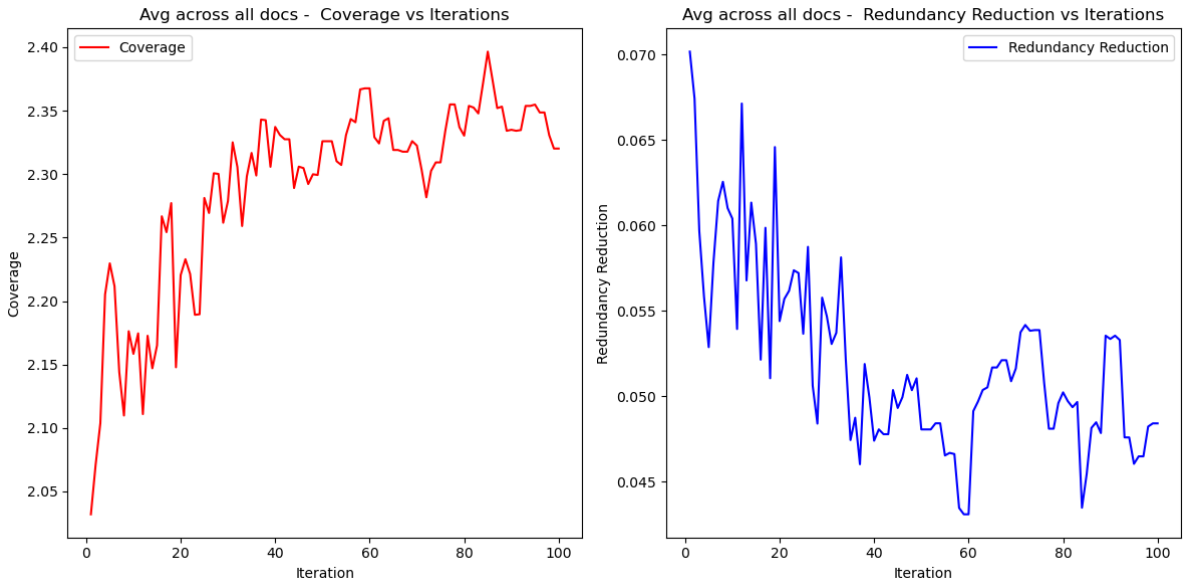


Fig 4.3 – Objective trend over iterations

CHAPTER - 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion:

We have applied multi-objective techniques to the problem of Extractive Text Summarization capabilities in an unsupervised learning framework. We obtain a ROUGE-1 and ROUGE-2 scores of 0.49 and 0.21 respectively. Parametrization is done to check the performance of the proposed algorithm against various settings. We also try to make changes to the sentence representations and the incorporation of topic modelling. We note that topic modelling has not much of a positive effect on ROUGE scores. We also note that V2-Mutation operator has a slightly better score when compared to other systems.

5.2 Future Work:

Topic modelling is a large area to be explored. We have used only the HDP model to encode sentences into the topic space. It could be that the sparsity of vectors have led to low ROUGE-scores. A more rigorous treatment and research in this area could lead to a good improvement. Also, we have plainly incorporated the use of S-BERT sentence transformer in our experiments. Its incorporation should be studied more deeply and we believe that there are more tunings that need to be performed in order for S-BERT to really showcase its contextual capabilities in a summarization settings. We also wish to use the existing model on larger and diverse datasets and observe its performance.

CHAPTER - 6

SOURCE CODE

```
# # Ram Ram
# import pickle
# with
open("/kaggle/input/duc2002dataandwheels/KaggleWheels/datasets/extracts_duc_filenam
es.pickle","rb") as file:
#     pass


#Ram Ram
if(False):
    !pip install keybert
    !pip install sentence_transformers
    !pip install rouge_score


    try:
        os.makedirs("/kaggle/working/results")
    except:
        pass

    try:
        os.makedirs("/kaggle/working/details")
    except:
        pass

    import os
    pack_list=[]
    for dirname, _, filenames in os.walk('/kaggle/input'):
        for filename in filenames:
            if filename.endswith('.whl'):
                pack_list.append(os.path.join(dirname,filename))

    for item in pack_list:
        !pip install item



import re
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer
nltk.download('stopwords')
nltk.download('punkt')
from nltk.stem import WordNetLemmatizer
```

```
nltk.download('wordnet')
nltk.download('omw-1.4')
import re
# import spacy
# nlp = spacy.load("en_core_web_sm")
import math
import numpy as np
from numpy import dot
from numpy.linalg import norm
import random
from itertools import combinations
import copy
import matplotlib.pyplot as plt

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.decomposition import LatentDirichletAllocation
from rouge_score import rouge_scorer
from datetime import datetime
from keybert import KeyBERT
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity


import pickle
from sentence_transformers import SentenceTransformer
model = SentenceTransformer("all-MiniLM-L6-v2")

import random
from sklearn.decomposition import LatentDirichletAllocation

import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from sklearn.feature_extraction.text import CountVectorizer

import networkx as nx
from csv import writer


import random
from itertools import combinations




# !pip install lexrank
# !pip install tomotopy
```

```
# from lexrank import LexRank
# from lexrank.mappings.stopwords import STOPWORDS
# from path import Path

# import time
# import timeit

# import tomotopy as tp
# from nltk.stem import *



def remove_specials(text):
    text=text.replace(',', ' ')
    text=text.replace(':', ' ')
    text=text.replace(';', ' ')
    text=text.replace('\n', ' ')
    text=text.replace('\t', ' ')
    return text

def remove_special_chars(input_string):
    # Exclude ? ! ' . from the pattern
    pattern = r'[^\w\s?!\'."]'
    result = re.sub(pattern, ' ', input_string)
    return result


def read_text_file(file_path):
    with open(file_path, 'r') as file:
        file_content = file.read()
    return file_content

def segmentation(input_stream):
    # Split based on . ? ! '
    delimiters = ['.', '?', '!']
    # Create a pattern for the regex to split on any of the delimiters
    pattern = '|'.join(map(re.escape, delimiters))
    list_of_sentences = re.split(pattern, str(input_stream))

    # Remove empty and whitespace-only strings from the list
    list_of_sentences = [sentence.strip() for sentence in list_of_sentences if sentence.strip()
!= ""]

    return list_of_sentences

def tokenization(sentence_list,token_check=True):
    sentence_word_list=[]
    for sentence in sentence_list:
        sentence_word_list.append(word_tokenize(sentence))
```

```
        if token_check==True:
            new_word_list=[]
            for i in range(len(sentence_word_list[-1])):
                if sentence_word_list[-1][i].isalnum():
                    new_word_list.append(sentence_word_list[-1][i])
            sentence_word_list[-1]=new_word_list
        for i in range(len(sentence_word_list[-1])):
            sentence_word_list[-1][i]=sentence_word_list[-1][i].strip().lower()

    return sentence_word_list

def remove_stop_words(word_list):
    stop_words = set(stopwords.words('english'))
    new_word_list=[]
    for word in word_list:
        if word not in stop_words:
            new_word_list.append(word)
    return new_word_list

def root_lemmatizer(word_list):
#     lemmatizer = WordNetLemmatizer()
    porter = PorterStemmer()
    stemmed_words = [porter.stem(word) for word in word_list]
#     lemmatized_words = [lemmatizer.lemmatize(word,pos='n') for word in word_list]
    #stemmed_words = [porter.stem(word) for word in lemmatized_words]
    return stemmed_words # lemmatized_words


def remove_stop_words_spacy(word_list):
    stop_words = set(nlp.Defaults.stop_words)
    new_word_list = [word for word in word_list if word not in stop_words]
    return new_word_list

def porter_stemmer_spacy(word_list):
    # Since SpaCy uses lemmatization instead of stemming,
    # we can use the SpaCy lemmatizer to get the base form of words.
    stemmed_words = [token.lemma_ for token in nlp(" ".join(word_list))]
    return stemmed_words

def tokenization_spacy(sentence_list,token_check=True):
    sentence_word_list = []
    for sentence in sentence_list:
        doc = nlp(sentence)
        new_word_list = [token.text.lower() for token in doc if token.is_alpha]
        if token_check==True:
            new_word_list_spacy=[]
            for i in range(len(new_word_list)):
                if new_word_list[i].isalnum():
                    new_word_list_spacy.append(new_word_list[i])
```

```
            sentence_word_list.append(new_word_list_spacy)
        else:
            sentence_word_list.append(new_word_list)

        for i in range(len(sentence_word_list[-1])):
            sentence_word_list[-1][i]=sentence_word_list[-1][i].strip().lower()
    return sentence_word_list


def
preprocess(input_stream,spacy_use=False,token_check=True,stop_check=True,stem_che
ck=True):
    processed_input_stream=segmentation(input_stream)
    NPIS=[]

    for i in range(len(processed_input_stream)):
        if len(processed_input_stream[i].split()) >=5:
            NPIS.append(processed_input_stream[i])
    processed_input_stream=NPIS.copy()


    if spacy_use==False:
        word_lists=tokenization(processed_input_stream,token_check)
        new_word_lists=[]
        negation=[]
        for i in range(len(word_lists)):
            l=word_lists[i]
            new_l=remove_stop_words(l)
            if stop_check==True:
                if len(new_l)>=1:
                    new_word_lists.append(new_l)
                else:
                    negation.append(i)
            else:
                new_word_lists.append(new_l)

        new_processed_input_stream=[]
        for i in range(len(processed_input_stream)):
            if i not in negation:
                new_processed_input_stream.append(processed_input_stream[i])
        processed_input_stream=new_processed_input_stream.copy()
        negation=[]
        stemmed_word_lists=[]
        for i in range(len(new_word_lists)):
            l=new_word_lists[i]
            new_l=root_lemmatizer(l)
            if stem_check==True:
                if len(new_l)>=1:
                    stemmed_word_lists.append(new_l)
```

```
            else:
                negation.append(i)
        else:
            stemmed_word_lists.append(new_l)

    new_processed_input_stream=[]
    for i in range(len(processed_input_stream)):
        if i not in negation:
            new_processed_input_stream.append(processed_input_stream[i])
    processed_input_stream=new_processed_input_stream.copy()



    NPIS=[]
    new_SWL=[]
    for i in range(len(stemmed_word_lists)):
        if len(stemmed_word_lists[i]) >=3:
            NPIS.append(processed_input_stream[i])
            new_SWL.append(stemmed_word_lists[i])
        else:
            pass
    processed_input_stream=NPIS.copy()
    stemmed_word_lists=new_SWL.copy()
    return stemmed_word_lists,processed_input_stream
else:
    word_lists=tokenization_spacy(processed_input_stream,token_check)
    new_word_lists=[]
    negation=[]
    for i in range(len(word_lists)):
        l=word_lists[i]
        new_l=remove_stop_words_spacy(l)
        if stop_check==True:
            if len(new_l)>=1:
                new_word_lists.append(new_l)
            else:
                negation.append(i)
        else:
            new_word_lists.append(new_l)


    new_processed_input_stream=[]
    for i in range(len(processed_input_stream)):
        if i not in negation:
            new_processed_input_stream.append(processed_input_stream[i])
    processed_input_stream=new_processed_input_stream.copy()
    negation=[]
    stemmed_word_lists=[]
    for i in range(len(new_word_lists)):
        l=new_word_lists[i]
```

```
            new_l=porter_stemmer_spacy(l)
            if stem_check==True:
                if len(new_l)>=1:
                    stemmed_word_lists.append(new_l)
                else:
                    negation.append(i)
            else:
                stemmed_word_lists.append(new_l)

        new_processed_input_stream=[]
        for i in range(len(processed_input_stream)):
            if i not in negation:
                new_processed_input_stream.append(processed_input_stream[i])
        processed_input_stream=new_processed_input_stream.copy()

        NPIS=[]
        new_SWL=[]
        for i in range(len(stemmed_word_lists)):
            if len(stemmed_word_lists[i]) >=3:
                NPIS.append(processed_input_stream[i])
                new_SWL.append(stemmed_word_lists[i])
            else:
                pass
        processed_input_stream=NPIS.copy()
        stemmed_word_lists=new_SWL.copy()


        return stemmed_word_lists,processed_input_stream

# def zeroprocess(input_stream):
#     processed_input_stream=segmentation(input_stream)
#     revised_input_stream=""
#     for item in processed_input_stream:
#         revised_input_stream+=remove_special_chars(item)+" \n "
#     revised_input_stream=revised_input_stream[:-1]
#     return revised_input_stream




    def tfik(word_sentence_list):
        tfik_dict={}
        for word in word_sentence_list:

            tfik_dict[word]=tfik_dict.get(word,0)+1
    #         print(tfik_dict,word,word_sentence_list)
        return tfik_dict
```

```
def distinct_terms(tfik_dict):
    terms_dict={}
    for k,v in tfik_dict.items():
        for word,ctr in v.items():
            terms_dict[word]=terms_dict.get(word,0)+1

    return terms_dict


def indices_terms_mapper(terms_dict):
    index_to_term_mapper={}
    term_to_index_mapper={}
    for i in range(len(list(terms_dict.keys()))):
        index_to_term_mapper[i]=list(terms_dict.keys())[i]
        term_to_index_mapper[list(terms_dict.keys())[i]]=i
    return  index_to_term_mapper, term_to_index_mapper



def
wik(tfik_dict,terms_dict,index_to_term_mapper,term_to_index_mapper,no_of_sentences,
no_of_terms):


    si_vec_dict={}
    for sentence_index in range(no_of_sentences):
        si_vec_dict[sentence_index]={}
        for term_index in range(no_of_terms):
            si_vec_dict[sentence_index][term_index]=0.0
    for sentence_index, occurrence_dict in tfik_dict.items():
        for word,count in occurrence_dict.items():
            si_vec_dict[sentence_index][term_to_index_mapper[word]]=count *
math.log(no_of_sentences/terms_dict[word],10)

    return si_vec_dict



def mean_o(wik_dict,no_of_sentences,no_of_terms):
    o_dict={}

    for term_index in range(no_of_terms):
        o_dict[term_index]=0

    for k,v in wik_dict.items():
        for term_index, wik_value in v.items():
```

```
            o_dict[term_index]+=wik_value
    for term_index in range(no_of_terms):
        o_dict[term_index]/=no_of_sentences

    return o_dict


def cos_sim(vec_a,vec_b):
    return dot(vec_a, vec_b)/(norm(vec_a)*norm(vec_b))

def coverage(x,doc_info):
    wik_dict=doc_info.wik_dict
    o_dict=doc_info.o_dict
    kw_topic_info=doc_info.kw_topic_info
    coverage_value=0
    for i in range(len(x)):
        coverage_value += doc_info.o_wik_sim[i] * x[i]
    return coverage_value


# def kw_multiplier(i,j,kw_topic_info):
#     return kw_topic_info.kw_multipliers_dict[(i,j)]
# def topic_multiplier(i,j,kw_topic_info):
#     return kw_topic_info.topic_multipliers_dict[(i,j)]

# def compare_vectors(vec1, vec2, val1=2, val2=1/2):
#     for i in range(len(vec1)):
#         if vec1[i] == 1 and vec2[i] == 1:
#             return val1
#     return val2

# def relevance(x,doc_info):
#     sum_vec={}
#     for k1,v1 in doc_info.o_dict.items():
#         sum_vec[k1]=0
#     for k1,v1 in doc_info.wik_dict.items():
#         for k2,v2 in v1.items():
#             sum_vec[k2]+=(v2 * x[k1])
#     for k1,v1 in doc_info.o_dict.items():
#         sum_vec[k1]/=doc_info.no_of_sentences

#     term_1=cos_sim(list(sum_vec.values()),doc_info.o_dict.values())
#     term_2=0
#     for i in range(len(doc_info.no_of_sentences)):
#         term_2+=(cos_sim(list(sum_vec.values()),list(wik_dict[i].values())) * x[i])

#     return term_1*term_2
```

```
def redundancy_reduction(x,doc_info):
    wik_dict=doc_info.wik_dict
    kw_topic_info=doc_info.kw_topic_info

#     time1=timeit.default_timer()
    redundancy_reduction_value=0
    for i in range(len(x)-1):
        for j in range(i+1,len(x)):
            redundancy_reduction_value += ((doc_info.wik_sim[(i,j)] *x[i]*x[j]))

    redundancy_reduction_value *= np.sum(x)
    redundancy_reduction_value=1/(1+redundancy_reduction_value)
#     time2=timeit.default_timer()
#     print("Inside Redundancy Reduction :",time2-time1)
    return redundancy_reduction_value
# print(redundancy_reduction(some_x_vec,wik_dict))

def redundancy_reduction_topic(x,doc_info):
    wik_dict=doc_info.wik_dict
    kw_topic_info=doc_info.kw_topic_info

#     time1=timeit.default_timer()
    redundancy_reduction_value=0
    for i in range(len(x)-1):
        for j in range(i+1,len(x)):
            redundancy_reduction_value += ((doc_info.t_wik_sim[(i,j)] *x[i]*x[j]))

    redundancy_reduction_value *= np.sum(x)
    redundancy_reduction_value=1/(1+redundancy_reduction_value)
#     time2=timeit.default_timer()
#     print("Inside Redundancy Reduction :",time2-time1)
    return redundancy_reduction_value
# print(redundancy_reduction(some_x_vec,wik_dict))


def get_NDS_consensus(NDS_archive):
    sentence_count_dict={}
    for i in range(len(NDS_archive[0].mem)):
        sentence_count_dict[i]=0
    for item in NDS_archive:
        for i in range(len(item.mem)):
            if item.mem[i]==1:
                sentence_count_dict[i]+=1
    return sentence_count_dict


def mutation_operator_v1(x, doc_info):
    revised_x = x.copy()
    wik_dict=doc_info.wik_dict
```

```
    o_dict=doc_info.o_dict
    pm=doc_info.pm

    RHS = np.mean([doc_info.o_wik_sim[j] for j in range(len(x))])

    mutation_probabilities = np.random.random(len(x))

    for i in range(len(x)):
        if mutation_probabilities[i] <= pm:

            LHS = doc_info.o_wik_sim[i]
            if LHS >= RHS:
                revised_x[i] = 1
            else:
                revised_x[i] = 0
    revised_x=fast_repair_operator(revised_x.copy(),doc_info)[0]
    return revised_x




def mutation_operator_v2(x, doc_info):
    revised_x = x.copy()
    wik_dict=doc_info.wik_dict
    o_dict=doc_info.o_dict
    pm=doc_info.pm

    RHS = np.mean([doc_info.o_wik_sim[j] for j in range(len(x))])

    mutation_probabilities = np.random.random(len(x))

    for i in range(len(x)):
        if mutation_probabilities[i] <= pm:

            LHS = doc_info.o_wik_sim[i]
            if LHS >= RHS:
                revised_x[i] = 1
            else:
                revised_x[i] = 0

    if(len(doc_info.NDS_archive)!=0):
        sentence_consensus = get_NDS_consensus(doc_info.NDS_archive)
        sentence_consensus_sorted = dict(sorted(sentence_consensus.items(), key=lambda
item: item[1],reverse=True))
        top=min(int(doc_info.no_of_sentences/10),doc_info.no_of_sentences)
        top_sentences=list(sentence_consensus_sorted.keys())[:top]
        for ind in top_sentences:
            if random.random()<pm:
                revised_x[ind]=1
```

```
    revised_x=fast_repair_operator(revised_x.copy(),doc_info)[0]
    return revised_x

def mutation_operator_select(x,doc_info):
    mutation_operator_version=doc_info.mutation_operator_version

    if mutation_operator_version==1:
        return mutation_operator_v1(x,doc_info)
    elif mutation_operator_version==2:
        return mutation_operator_v2(x,doc_info)




def term_indices_of_sentences(sentences,term_to_index_mapper):
    term_sentence_dict={}
    for i in range(len(sentences)):
        term_sentence_dict[i]=[]
        sentence=sentences[i]
        for term in sentence:
            term_sentence_dict[i].append(term_to_index_mapper[term])
        term_sentence_dict[i]=sorted(term_sentence_dict[i])
    return term_sentence_dict


def ovec_summary_modifier(o_dict,x,wik_dict):
    o_dict_temp={}
    for term_index in wik_dict[0].keys():
        o_dict_temp[term_index]=0
        for sentence_index in wik_dict.keys():

o_dict_temp[term_index]+=(wik_dict[sentence_index][term_index]*x[sentence_index])
    return o_dict_temp

def term_combined_list_former(set_of_sentences_indices,term_index_sentence_dict):
    combined=[]
    for sentence_index in set_of_sentences_indices:
        for term_index in term_index_sentence_dict[sentence_index]:
            if term_index not in combined:
                combined.append(term_index)
    return combined



def repair_operator(x,doc_info):  #term_index_sentence_dict is not there as of now
    preprocessed_inputs=doc_info.preprocessed_inputs
    list_of_sentences=doc_info.list_of_sentences
    wik_dict=doc_info.wik_dict
    o_dict=doc_info.o_dict
```

```
    epsilon=doc_info.epsilon
    delta=doc_info.delta
    L=doc_info.L

    cur_length_of_summary=0
    for i in range(len(x)):
        if x[i]==1:
            cur_length_of_summary+=doc_info.sentence_lengths[i]

    lower_bound=L-epsilon
    upper_bound=L+epsilon

    if cur_length_of_summary >=lower_bound and cur_length_of_summary<=
upper_bound:
        return (x,True)

    if cur_length_of_summary<lower_bound:
        return (x,True)


    if cur_length_of_summary >=lower_bound and cur_length_of_summary>
upper_bound:

        #Repair needed
        trial = 0
        max_trials=min((len(preprocessed_inputs)**2) ,300)
        print("Repair Trials : ",max_trials)
        while cur_length_of_summary> upper_bound and trial<=max_trials:
            summary_indices = [index for index, value in enumerate(x) if value == 1]
            try:
                chosen_combination= random.choice(list(combinations(summary_indices,2)))
            except:
                return (x,True)
            i=chosen_combination[0]
            j=chosen_combination[1]
            if i!=j and x[i]==1 and x[j]==1 and
cos_sim(list(wik_dict[i].values()),list(wik_dict[j].values())) >= delta:

                si=list(dict(sorted(wik_dict[i].items())).values())
                sj=list(dict(sorted(wik_dict[j].items())).values())
                o_vec=list(dict(sorted(o_dict.items())).values())

                x_without_i=x.copy()
                x_without_i[i]=0
                x_without_j=x.copy()
                x_without_j[j]=0
```

```
o_sum_vec=list(dict(sorted(ovec_summary_modifier(o_dict,x,wik_dict).items())).values(
))

o_sum_vec_without_i=list(dict(sorted(ovec_summary_modifier(o_dict,x_without_i,wik_
dict).items())).values())

o_sum_vec_without_j=list(dict(sorted(ovec_summary_modifier(o_dict,x_without_j,wik_
dict).items())).values())

            score_si=cos_sim(si,o_vec) + (cos_sim(o_sum_vec,o_vec)-
cos_sim(o_sum_vec_without_i,o_vec))*10
            score_sj=cos_sim(sj,o_vec) + (cos_sim(o_sum_vec,o_vec)-
cos_sim(o_sum_vec_without_j,o_vec))*10

            if score_si >= score_sj:
               #Discard sj, Keep si
               x[j]=0
               cur_length_of_summary-=doc_info.sentence_lengths[i]
            else:
               x[i]=0
               cur_length_of_summary-=doc_info.sentence_lengths[i]

         trial+=1

      return (x,True)


def select_and_return_scores(doc_info):
   scores={}
   score_types=["textrank","tf","graph","lexrank"]
   score_type_chosen=random.choice(score_types)
   if score_type_chosen=="tf":
      scores=doc_info.scores_details.tf_scores
   elif score_type_chosen=="textrank":
      scores=doc_info.scores_details.textrank_scores
   elif score_type_chosen=="graph":
      scores=doc_info.scores_details.graph_scores
   elif score_type_chosen=="lexrank":
      scores=doc_info.scores_details.lexrank_scores
   return scores
def fast_repair_operator(x,doc_info):
   preprocessed_inputs=doc_info.preprocessed_inputs
   list_of_sentences=doc_info.list_of_sentences
   wik_dict=doc_info.wik_dict
   o_dict=doc_info.o_dict
   epsilon=doc_info.epsilon
   delta=doc_info.delta
   L=doc_info.L
```

```
    cur_length_of_summary=0
    for i in range(len(x)):
        if x[i]==1:
            cur_length_of_summary+=doc_info.sentence_lengths[i]

    lower_bound = L - epsilon
    upper_bound = L + epsilon

    scores=select_and_return_scores(doc_info)

    present_scores={}
    absent_scores={}
    for i in range(len(x)):
        if x[i]==1:
            present_scores[i]=scores[i]
        else:
            absent_scores[i]=scores[i]

    if cur_length_of_summary >=lower_bound and cur_length_of_summary<=
upper_bound:
        return (x,True)


    elif cur_length_of_summary<lower_bound:


        while cur_length_of_summary < lower_bound:
            if len(list(absent_scores.keys())) <=0:
                return (x,True)
            else:

chosen_combination=random.choice(list(combinations(list(absent_scores.keys()),2)))
                si=chosen_combination[0]
                sj=chosen_combination[1]
                if scores[si]>scores[sj]:
                    sentence_to_be_added=si
                else:
                    sentence_to_be_added=sj

                x[sentence_to_be_added]=1
                absent_scores.pop(sentence_to_be_added)
                cur_length_of_summary += doc_info.sentence_lengths[sentence_to_be_added]
        return (x,True)

    elif cur_length_of_summary >=lower_bound and cur_length_of_summary>
upper_bound:
```

```
        while cur_length_of_summary >L+epsilon: #            present_scores =
dict(sorted(present_scores.items(), key=lambda item: item[1]))

            if len(list(present_scores.keys()))<=1:
                return (x,True)
            else:
                chosen_combination=
random.choice(list(combinations(list(present_scores.keys()),2)))
                si=chosen_combination[0]
                sj=chosen_combination[1]
                if scores[si]> scores[sj]:
                    sentence_to_be_removed=sj
                else:
                    sentence_to_be_removed=si
    #             sentence_to_be_removed=list(present_scores.keys())[0]
                x[sentence_to_be_removed]=0
                present_scores.pop(sentence_to_be_removed)
                cur_length_of_summary-
=doc_info.sentence_lengths[sentence_to_be_removed]

        return (x,True)



def words_per_sentence(list_of_sentences):
    words_per_sentence_dict={}
    for i in range(len(list_of_sentences)):
        words_per_sentence_dict[i]=len(list_of_sentences[i].split())



def dominate(vec0,vec1):
    if vec0==vec1:
        return "False"
    no_of_dimensions=len(vec0)
    count_vec=[0]*2

    for d in range(no_of_dimensions):
        if vec0[d]>vec1[d]:
            count_vec[0]+=1
        if vec1[d]>vec0[d]:
            count_vec[1]+=1

    if count_vec[0]==no_of_dimensions:
        #Vec 0 won
        return 0
    elif count_vec[1]==no_of_dimensions:
        #Vec 1 won
        return 1
```

```
    elif count_vec[0]==count_vec[1]:
        #no one won= NDS
        return "False"




class member:
    def __init__(self,mem_input,doc_info):
        self.mem=mem_input.copy()
        self.trials=0
        self.objective=self.calculate_objective(self.mem,doc_info)
        self.fitness=self.transform_objective(self.objective)
        self.length_of_summary=self.calculate_length_of_summary(self.mem,doc_info)

    def calculate_length_of_summary(self,x,doc_info):
        sum_len=0
        for i in range(len(x)):
            if x[i]==1:
                sum_len+= doc_info.sentence_lengths[i]
        return sum_len


    def transform_objective(self,objective):
        return objective


    def calculate_objective(self,mem,doc_info):
        if doc_info.topic_enabled=="v1":
            coverage_obj=coverage_topic(mem,doc_info)
            redundancy_obj=redundancy_reduction_topic(mem,doc_info)
        elif doc_info.topic_enabled=="v2":
            coverage_obj=coverage_topic(mem,doc_info)
            redundancy_obj=redundancy_reduction(mem,doc_info)
        elif doc_info.topic_enabled=="v3":
            coverage_obj=coverage(mem,doc_info)
            redundancy_obj=redundancy_reduction(mem,doc_info)
        elif doc_info.topic_enabled=="v4":
            coverage_obj=coverage(mem,doc_info)
            redundancy_obj=redundancy_reduction_topic(mem,doc_info)
        return [coverage_obj,redundancy_obj]


def coverage_topic(x,doc_info):
    coverage_value=0
    for i in range(len(x)):
        coverage_value += doc_info.t_o_wik_sim[i] * x[i]
    return coverage_value
```

```
def initialize_population(doc_info,less_ct=-1):
    list_of_members=[]
    if less_ct==-1:
        for i in range(doc_info.pop_size):
            new_mem=np.random.randint(0,2,doc_info.no_of_sentences)
            list_of_members.append(member(new_mem,doc_info))
    else:
        for i in range(less_ct):
            new_mem=np.random.randint(0,2,doc_info.no_of_sentences)
            list_of_members.append(member(new_mem,doc_info))
    return list_of_members


def naive_replace(list_of_members,current_mem,new_mem,i):
    new_mem.trials=0
    list_of_members[i]=new_mem
    return list_of_members
def replace(list_of_members,current_mem,new_mem,i):
    result=dominate(current_mem.fitness,new_mem.fitness)
    if result==0:
        print("Not",current_mem.fitness,new_mem.fitness)
        list_of_members[i].trials+=1
    elif result==1:
        print("WON",current_mem.fitness,new_mem.fitness)
        new_mem.trials=0
        list_of_members[i]=new_mem
    elif result=="False":
        #print("Non Dominating each other",current_mem.fitness,new_mem.fitness)
        list_of_members[i].trials+=1
    return list_of_members




def send_employed_bees(list_of_members, doc_info):
    start_time = time.time()  # Start timing the function
    no_of_members = len(list_of_members)
    l = []
    for i in range(no_of_members):
        current_mem = list_of_members[i]
#         time_before_mutation = time.time()
        new_mem_list = mutation_operator_select(current_mem.mem.copy(), doc_info)
#         time_after_mutation = time.time()
        new_mem = member(new_mem_list.copy(), doc_info)
#         time_after_member_creation = time.time()
```

```
list_of_members=replacement_operator_select(list_of_members,current_mem,new_mem,
i,doc_info)
#         time_after_replace = time.time()

#         print("Time for mutation:", time_after_mutation - time_before_mutation)
#         print("Time for member creation:", time_after_member_creation -
time_after_mutation)
#         print("Time for replace", time_after_replace - time_after_member_creation)

    return list_of_members
def roulette_wheel_selection(probabilities):
    cumulative_probabilities = [sum(probabilities[:i+1]) for i in range(len(probabilities))]
    total_probability = cumulative_probabilities[-1]
    random_number = random.uniform(0, total_probability)
    for i, cumulative_prob in enumerate(cumulative_probabilities):
        if random_number <= cumulative_prob:
            return i

def ranking(list_of_members):
    sp_dict={}
    np_dict={}
    f_dict={}
    f_dict[1]=[]
    for i in range(len(list_of_members)):
        p=list_of_members[i]
        sp_dict[i]=[]
        np_dict[i]=0
        for j in range(len(list_of_members)):
            if i!=j:
                q=list_of_members[j]
                result=dominate(p.objective,q.objective)
                if result==0:
                    sp_dict[i].append(j)
                elif result==1:
                    np_dict[i]+=1
        if np_dict[i]==0:
            f_dict[1].append(i)


    front_index=1
    while len(f_dict[front_index])!=0:
        Q=[]
        for p in f_dict[front_index]:
            for q in sp_dict[p]:
                np_dict[q]-=1
                if np_dict[q]==0:
                    Q.append(q)
```

```
        front_index+=1
        f_dict[front_index]=Q.copy()

    if len(f_dict[front_index])==0:
        r_v=f_dict.pop(front_index)

    return f_dict.copy()


def sort_dict(my_dict,objective_index):
    sorted_dict = dict(sorted(my_dict.items(), key=lambda item:
item[1][objective_index]))
    return sorted_dict

def crowding_distance(front_member_indices,list_of_members,no_of_objectives=2):
    current_member_objective_dict={}
    di_dict={}
    for member_index in front_member_indices:

current_member_objective_dict[member_index]=list_of_members[member_index].object
ive
        di_dict[member_index]=10**-8

    for objective_index in range(no_of_objectives):

sorted_dict_list=list(sort_dict(current_member_objective_dict.copy(),objective_index))
        if len(sorted_dict_list)<=2:
            for i in range(len(sorted_dict_list)):
                di_dict[sorted_dict_list[i]]=10**8
            return di_dict
        else:
            lower_extreme=sorted_dict_list[0]
            upper_extreme=sorted_dict_list[-1]
            di_dict[lower_extreme]=10**8
            di_dict[upper_extreme]=10**8
            for i in range(2,len(sorted_dict_list)-1):
                left_index=sorted_dict_list[i-1]
                right_index=sorted_dict_list[i+1]
                right_obj=list_of_members[right_index].objective[objective_index]
                left_obj=list_of_members[left_index].objective[objective_index]
                upper_obj=list_of_members[upper_extreme].objective[objective_index]
                lower_obj=list_of_members[lower_extreme].objective[objective_index]

#print(list_of_members[right_index].objective[objective_index],list_of_members[left_ind
ex].objective[objective_index],list_of_members[upper_extreme].objective[objective_inde
x],list_of_members[lower_extreme].objective[objective_index])
                val=abs(right_obj-left_obj)/abs(upper_obj-lower_obj)
                if val>=0 or val<0:
                    di_dict[sorted_dict_list[i]]+=val
```

```
            else:
#
print(list_of_members[right_index].objective[objective_index],list_of_members[left_inde
x].objective[objective_index],list_of_members[upper_extreme].objective[objective_index
],list_of_members[lower_extreme].objective[objective_index])
#                   print("OMG THIS HAS HAPPENED")
                di_dict[sorted_dict_list[i]]+=1
    return di_dict



class front_info:
    def __init__(self,member,member_index,rank,cd):
        self.member_index=member_index
        self.member=member
        self.rank=rank
        self.cd=cd

def probability_calculations(front_info_list):
    prob_list=[]
    for i in range(len(front_info_list)):
        value=1/(front_info_list[i].rank +(1/(1+front_info_list[i].cd)))
        prob_list.append(0.9*value +0.1)

    return prob_list



def front_info_former(list_of_members,doc_info):
    no_of_objectives=doc_info.no_of_objectives
    f_dict=ranking(list_of_members)
#     for k,v in f_dict.items():
#         print("!!!!! Rank = ",k)
#         for me in v:
#             print(list_of_members[me].objective)

    front_info_dict={}
    for i in range(len(list_of_members)):
        front_info_dict[i]=0
    for k,v in f_dict.items():
        cd_values=crowding_distance(v,list_of_members,no_of_objectives)
        for member_index in v:

front_info_dict[member_index]=front_info(list_of_members[member_index],member_in
dex,k,cd_values[member_index])
    return front_info_dict

def send_onlooker_bees(list_of_members,doc_info):
    no_of_objectives=doc_info.no_of_objectives
```

```
    front_info_list=list(front_info_former(list_of_members,doc_info).values())

    prob_list=probability_calculations(front_info_list)
    no_of_members=len(list_of_members)
    l=[]
    for i in range(no_of_members):
        roulette_index=roulette_wheel_selection(prob_list)
        current_mem=list_of_members[roulette_index]
        new_mem_list=mutation_operator_select(current_mem.mem.copy(),doc_info)
        new_mem=member(new_mem_list.copy(),doc_info)
        result=dominate(current_mem.objective, new_mem.objective)
        if result==0 or result=="False":
            list_of_members[roulette_index].trials+=1
        list_of_members.append(new_mem)
    return list_of_members


def rank_and_crowd_onlookers(list_of_members,doc_info):
    no_of_objectives=doc_info.no_of_objectives
    front_info_list=list(front_info_former(list_of_members,doc_info).values())
    prob_list=probability_calculations(front_info_list)
    prob_dict = {index: value for index, value in enumerate(prob_list)}
    sorted_dict = dict(sorted(prob_dict.items(), key=lambda item: item[1], reverse=True))
    ranked_list_indices=list(sorted_dict.keys())[0:int(len(list_of_members)/2)]
    ranked_list_of_members=[]
    for key in ranked_list_indices:
        ranked_list_of_members.append(list_of_members[key])
    return ranked_list_of_members




def send_scout_bees(list_of_members,doc_info):
    for i in range(len(list_of_members)):
        current_mem=list_of_members[i]
        if current_mem.trials<=doc_info.trial_cutoff_limit:
            pass
        else:
            soln_beaten=False
            for mutation_trial in range(10): #range(cycle)
                if soln_beaten==True:
                    break
                else:

new_mem_list=mutation_operator_select(current_mem.mem.copy(),doc_info)
                    new_mem=member(new_mem_list.copy(),doc_info)
                    result=dominate(current_mem.objective,new_mem.objective)
                    if result==1:
```

```
list_of_members=naive_replace(list_of_members,current_mem,new_mem,i)
                    soln_beaten=True

            if soln_beaten==False:
                #Soln still not replaced

random_selected_member=member(np.random.randint(0,2,len(list_of_members[0].mem)
),doc_info)
                random_selected_member.trials=0

list_of_members=naive_replace(list_of_members,current_mem,random_selected_membe
r,i)

    return list_of_members



def NDS_insert_ver_5(NDS_archive,list_of_members):
    for i in range(len(list_of_members)):
        member_to_insert=list_of_members[i]
        if len(NDS_archive)==0:
            NDS_archive.append(member_to_insert)
        else:
            to_be_removed=[]
            adding_flag=True
            for j in range(len(NDS_archive)):
                result=dominate(member_to_insert.objective,NDS_archive[j].objective)
                if result==0:
                    #This means ES has won!
                    to_be_removed.append(j)
                elif result==1:
                    #This means NDS archive item has won
                    adding_flag=False
            new_NDS_archive=[]
            for ind in range(len(NDS_archive)):
                #Here whatever items are present in to_be_removed list, are not added to new
NDS archive
                if ind not in to_be_removed:
                    new_NDS_archive.append(NDS_archive[ind])

            if adding_flag==True:
                # If none of the NDS archive items were able to dominate ES, then we insert it
                new_NDS_archive.append(member_to_insert)
            NDS_archive=new_NDS_archive.copy()

    return NDS_archive
```

```
def size_check(x,list_of_sentences):
    len_of_summary=0
    for i in range(len(x)):
        if x[i]==1:
            len_of_summary+=len(list_of_sentences[i].split())
    return len_of_summary


def NDS_insert_ver_6(NDS_archive,list_of_members,doc_info):
    list_of_sentences=doc_info.list_of_sentences
    L=doc_info.L
    epsilon=doc_info.L
    for i in range(len(list_of_members)):
        member_to_insert=list_of_members[i]
        if len(NDS_archive)==0:
            NDS_archive.append(member_to_insert)
        else:
            to_be_removed=[]
            adding_flag=True
            for j in range(len(NDS_archive)):
                result=dominate(member_to_insert.objective,NDS_archive[j].objective)
                if result==0:
                    #This means ES has won!
                    to_be_removed.append(j)
                elif result==1:
                    #This means NDS archive item has won
                    adding_flag=False
                elif member_to_insert.objective==NDS_archive[j].objective:
                    adding_flag=False
#                   print("Ram Ram")
            new_NDS_archive=[]
            for ind in range(len(NDS_archive)):
                #Here whatever items are present in to_be_removed list, are not added to new
NDS archive
                if ind not in to_be_removed:
                    len_of_summary=size_check(NDS_archive[ind].mem,list_of_sentences)
                    if len_of_summary >=L-epsilon and len_of_summary <= L+epsilon:
                        new_NDS_archive.append(NDS_archive[ind])

            if adding_flag==True:
                # If none of the NDS archive items were able to dominate ES, then we insert it
                len_of_summary=size_check(member_to_insert.mem,list_of_sentences)
                if len_of_summary <= L+epsilon:
                    new_NDS_archive.append(member_to_insert)
            NDS_archive=new_NDS_archive.copy()

    return NDS_archive

def normalize(NDS_archive):
```

```
    normalized_NDS_archive=copy.deepcopy(NDS_archive)

    min_obj0=float('inf')
    max_obj0=0

    min_obj1=float('inf')
    max_obj1=0

    for soln in NDS_archive:
        min_obj0=min(min_obj0,soln.objective[0])
        max_obj0=max(max_obj0,soln.objective[0])
        min_obj1=min(min_obj1,soln.objective[1])
        max_obj1=max(max_obj1,soln.objective[1])


    for i in range(len(NDS_archive)):
        soln=NDS_archive[i]
        normalized_obj0 = (soln.objective[0] - min_obj0) / (max_obj0 - min_obj0)
        normalized_obj1 = (soln.objective[1] - min_obj1) / (max_obj1 - min_obj1)
        normalized_NDS_archive[i].objective[0]=normalized_obj0
        normalized_NDS_archive[i].objective[1]=normalized_obj1
    return normalized_NDS_archive.copy()

def mahalakshmi(p1,p2,sigma_dict):
    value=0
    for i in range(len(sigma_dict.values())):
        cod=i #current objective dimension
        value += ((p1[cod] - p2[cod])/sigma_dict[cod])**2
    value=np.sqrt(value)
    return value

def standard_deviation(normalized_NDS_archive):
    no_of_objectives=len(normalized_NDS_archive[0].objective)
    obj_dict={}
    sigma_dict={}
    mean_dict={}
    for obj_index in range(no_of_objectives):
        obj_dict[obj_index]=[]
        sigma_dict[obj_index]=0
        mean_dict[obj_index]=0

    for i in range(len(normalized_NDS_archive)):
        for obj_index in range(no_of_objectives):
            obj_dict[obj_index].append(normalized_NDS_archive[i].objective[obj_index])

    for obj_index in range(no_of_objectives):
        mean_dict[obj_index]=np.mean(obj_dict[obj_index])
    for obj_index in range(no_of_objectives):
        for v in obj_dict[obj_index]:
```

```
            sigma_dict[obj_index]+=(v-mean_dict[obj_index])**2
        sigma_dict[obj_index]/=len(normalized_NDS_archive)

    return sigma_dict

def shortest_point_to_ideal_point_mahalanobis(NDS_archive,doc_info):
    NNA=normalize(NDS_archive)
    sigma_dict=standard_deviation(NNA)
    best_point=NNA[0]
    ideal_point=(1,1)
    shortest_distance=mahalakshmi(best_point.objective,ideal_point,sigma_dict)
    for soln in range(1,len(NNA)):
        calc_dist=mahalakshmi(NNA[soln].objective,ideal_point,sigma_dict)
        if shortest_distance > calc_dist :
            best_point=NNA[soln]
            shortest_distance=calc_dist
    return best_point



def euclidean(vector1, vector2):
    if len(vector1) != len(vector2):
        raise ValueError("Vectors must have the same dimensionality")

    sum_of_squares = sum((x - y) ** 2 for x, y in zip(vector1, vector2))
    return math.sqrt(sum_of_squares)


def shortest_point_to_ideal_point_euclidean(NDS_archive,doc_info):
    NNA=normalize(NDS_archive)
    best_point=NNA[0]
    ideal_point=(1,1)
    shortest_distance=euclidean(best_point.objective,ideal_point)
    for soln in range(1,len(NNA)):
        calc_dist=euclidean(NNA[soln].objective,ideal_point)
        if shortest_distance > calc_dist :
            best_point=NNA[soln]
            shortest_distance=calc_dist
    return best_point
def manhattan(vector1, vector2):
    if len(vector1) != len(vector2):
        raise ValueError("Vectors must have the same dimensionality")

    return sum(abs(x - y) for x, y in zip(vector1, vector2))

def shortest_point_to_ideal_point_manhattan(NDS_archive,doc_info):
    NNA=normalize(NDS_archive)
    best_point=NNA[0]
    ideal_point=(1,1)
```

```
    shortest_distance=manhattan(best_point.objective,ideal_point)
    for soln in range(1,len(NNA)):
        calc_dist=manhattan(NNA[soln].objective,ideal_point)
        if shortest_distance > calc_dist :
            best_point=NNA[soln]
            shortest_distance=calc_dist
    return best_point


def chebyshev(vector1, vector2):
    if len(vector1) != len(vector2):
        raise ValueError("Vectors must have the same dimensionality")

    return max(abs(x - y) for x, y in zip(vector1, vector2))

def shortest_point_to_ideal_point_chebyshev(NDS_archive,doc_info):
    NNA=normalize(NDS_archive)
    best_point=NNA[0]
    ideal_point=(1,1)
    shortest_distance=chebyshev(best_point.objective,ideal_point)
    for soln in range(1,len(NNA)):
        calc_dist=chebyshev(NNA[soln].objective,ideal_point)
        if shortest_distance > calc_dist :
            best_point=NNA[soln]
            shortest_distance=calc_dist
    return best_point

def calculate_all_points(NDS_archive,origin_vec,doc_info,distance_type):
    val=0
    sigma_dict=standard_deviation(NDS_archive)
    for item in NDS_archive:
        if distance_type=="chebyshev":
            val +=chebyshev(origin_vec,item.objective)
        elif distance_type=="manhattan":
            val +=manhattan(origin_vec,item.objective)
        elif distance_type=="euclidean":
            val +=euclidean(origin_vec,item.objective)
        elif distance_type=="mahalanobis":
            val +=mahalakshmi(origin_vec,item.objective,sigma_dict)
    return val

def shortest_point_to_all_points_chebyshev(NDS_archive,doc_info):
    best_point=NDS_archive[0]

shortest_distance=calculate_all_points(NDS_archive,best_point.objective,doc_info,"cheb
yshev")
    for soln in range(1,len(NDS_archive)):
```

```
calc_dist=calculate_all_points(NDS_archive,NDS_archive[soln].objective,doc_info,"che
byshev")
        if shortest_distance > calc_dist :
            best_point=NDS_archive[soln]
            shortest_distance=calc_dist
    return best_point

def shortest_point_to_all_points_manhattan(NDS_archive,doc_info):
    best_point=NDS_archive[0]

shortest_distance=calculate_all_points(NDS_archive,best_point.objective,doc_info,"man
hattan")
    for soln in range(1,len(NDS_archive)):

calc_dist=calculate_all_points(NDS_archive,NDS_archive[soln].objective,doc_info,"man
hattan")
        if shortest_distance > calc_dist :
            best_point=NDS_archive[soln]
            shortest_distance=calc_dist
    return best_point

def shortest_point_to_all_points_euclidean(NDS_archive,doc_info):
    best_point=NDS_archive[0]

shortest_distance=calculate_all_points(NDS_archive,best_point.objective,doc_info,"eucli
dean")
    for soln in range(1,len(NDS_archive)):

calc_dist=calculate_all_points(NDS_archive,NDS_archive[soln].objective,doc_info,"eucl
idean")
        if shortest_distance > calc_dist :
            best_point=NDS_archive[soln]
            shortest_distance=calc_dist
    return best_point

def shortest_point_to_all_points_mahalanobis(NDS_archive,doc_info):
    best_point=NDS_archive[0]

shortest_distance=calculate_all_points(NDS_archive,best_point.objective,doc_info,"mah
alanobis")
    for soln in range(1,len(NDS_archive)):

calc_dist=calculate_all_points(NDS_archive,NDS_archive[soln].objective,doc_info,"mah
alanobis")
        if shortest_distance > calc_dist :
            best_point=NDS_archive[soln]
            shortest_distance=calc_dist
    return best_point
```

```
def topmost_coverage(NDS_archive,doc_info):
    list_of_sentences=doc_info.list_of_sentences
    Lmax=doc_info.L + doc_info.epsilon
    best_mem=-1
    best_mem_score=-1
    for item in NDS_archive:
        if item.objective[0]>best_mem_score:
            best_mem=item
            best_mem_score=item.objective[0]
    return best_mem

def topmost_redundancy_reduction(NDS_archive,doc_info):
    list_of_sentences=doc_info.list_of_sentences
    Lmax=doc_info.L + doc_info.epsilon
    best_mem=-1
    best_mem_score=-1
    for item in NDS_archive:
        if item.objective[1]>best_mem_score:
            best_mem=item
            best_mem_score=item.objective[1]
    return best_mem

def consensus(NDS_archive,doc_info):
    list_of_sentences=doc_info.list_of_sentences
    Lmax=doc_info.L + doc_info.epsilon
    sentence_count_dict={}
    no_of_sentences=len(NDS_archive[0].mem)
    for i in range(no_of_sentences):
        sentence_count_dict[i]=0
    for i in range(len(NDS_archive)):
        for j in range(no_of_sentences):
            if NDS_archive[i].mem[j]==1:
                sentence_count_dict[j]+=1

    sentence_count_dict_sorted = dict(sorted(sentence_count_dict.items(), key=lambda
item: item[1],reverse=True))
    final_mem=[0]*no_of_sentences
    cur_length=0
    for k,v in sentence_count_dict_sorted.items():
        if cur_length + doc_info.sentence_lengths[k] >Lmax:
            break
        else:
            final_mem[k]=1
            cur_length+=doc_info.sentence_lengths[k]
#     print(sentence_count_dict_sorted)
    return member(final_mem,doc_info)
```

```
def largest_hypervolume(NDS_archive,doc_info):
    l_area=-1
    l_mem=0

    for archive_member in NDS_archive:
        cur_area=abs(archive_member.objective[0] * archive_member.objective[1])
        if cur_area > l_area:
            l_area=cur_area
            l_mem=archive_member
    return l_mem

def calculate_all_points_levenshtein(NDS_archive,sumi):
    sumi_set=set([i for i, val in enumerate(sumi) if val == 1])
    dist=0
    for item in NDS_archive:
        sumj_set=set([i for i,val in enumerate(item.mem) if val==1])
        dist+=len(sumi_set) + len(sumj_set) - 2*len(sumi_set.intersection(sumj_set))
    return dist


def shortest_point_to_all_points_levenshtein(NDS_archive,doc_info):
    best_point=NDS_archive[0]
    shortest_distance=calculate_all_points_levenshtein(NDS_archive,best_point.mem)
    for soln in range(1,len(NDS_archive)):
        calc_dist=calculate_all_points_levenshtein(NDS_archive,NDS_archive[soln].mem)
        if shortest_distance > calc_dist :
            best_point=NDS_archive[soln]
            shortest_distance=calc_dist
    return best_point

class rouge_score_details:
    def __init__(self,NDS_archive,doc_info):
        self.consensus_extract=consensus(NDS_archive,doc_info)
        self.largest_hypervolume_extract=largest_hypervolume(NDS_archive,doc_info)
        self.topmost_coverage_extract=topmost_coverage(NDS_archive,doc_info)

self.topmost_redundancy_reduction_extract=topmost_redundancy_reduction(NDS_archi
ve,doc_info)

self.shortest_point_to_ideal_point_euclidean_extract=shortest_point_to_ideal_point_eucl
idean(NDS_archive,doc_info)

self.shortest_point_to_ideal_point_manhattan_extract=shortest_point_to_ideal_point_ma
nhattan(NDS_archive,doc_info)

self.shortest_point_to_ideal_point_chebyshev_extract=shortest_point_to_ideal_point_che
byshev(NDS_archive,doc_info)
```

```
self.shortest_point_to_ideal_point_mahalanobis_extract=shortest_point_to_ideal_point_
mahalanobis(NDS_archive,doc_info)

self.shortest_point_to_all_points_chebyshev_extract=shortest_point_to_all_points_cheby
shev(NDS_archive,doc_info)

self.shortest_point_to_all_points_manhattan_extract=shortest_point_to_all_points_manha
ttan(NDS_archive,doc_info)

self.shortest_point_to_all_points_mahalanobis_extract=shortest_point_to_all_points_mah
alanobis(NDS_archive,doc_info)

self.shortest_point_to_all_points_euclidean_extract=shortest_point_to_all_points_euclide
an(NDS_archive,doc_info)

self.shortest_point_to_all_points_levenshtein_extract=shortest_point_to_all_points_leven
shtein(NDS_archive,doc_info)



def plot_objectives(NDS_archive,title_string):
    x_values=[]
    y_values=[]

    for soln in NDS_archive:
        x_values.append(soln.objective[0])
        y_values.append(soln.objective[1])

    plt.scatter(x_values, y_values, label='Multi Objective Optimization')
    plt.xlabel('Coverage Objective')
    plt.ylabel('Redundancy Reduction Objective')
    plt.title('Multi Objective : '+title_string)

def SBERT_embeddings(list_of_sentences):
    o_dict={}
    wik_dict={}
    model = SentenceTransformer("all-MiniLM-L6-v2")
    embeddings=model.encode(list_of_sentences[0])
    for i in range(len(embeddings)):
        o_dict[i]=0
#         o_dict[i]=embeddings[i]

    for s_index in range(len(list_of_sentences)):
        embeddings=model.encode(list_of_sentences[s_index])
        wik_dict[s_index]={}
        for j in range(len(embeddings)):
            wik_dict[s_index][j]=embeddings[j]
            o_dict[j]+=embeddings[j]
```

```
    for j in range(len(embeddings)):
        o_dict[j]/=len(list_of_sentences)

#     print(o_dict,temp)
    return (o_dict,wik_dict)



def preprocess_data(preprocessed_inputs):
    preprocessed_sentences = []
    for i in range(len(preprocessed_inputs)):
        preprocessed_sentences.append(" ".join(preprocessed_inputs[i]))

    return preprocessed_sentences



def retrieve_topic_assignment_matrix(list_of_sentences,preprocessed_inputs,
num_topics=10,n_top_words=20):

#     preprocessed_sentences = preprocess_data(preprocessed_inputs)
#     vectorizer = CountVectorizer()
#     document_term_matrix = vectorizer.fit_transform(preprocessed_sentences)
#     lda_model = LatentDirichletAllocation(n_components=num_topics,
random_state=42)
#     lda_output = lda_model.fit_transform(document_term_matrix)
#     feature_names = vectorizer.get_feature_names_out()
#     topic_to_words_mapper={}
#     for topic_idx, topic in enumerate(lda_model.components_):
#         top_words_idx = topic.argsort()[:-n_top_words - 1:-1]
#         top_words = [feature_names[i] for i in top_words_idx]
#         topic_to_words_mapper[topic_idx]=top_words.copy()


#     sentence_topic_probabilities = lda_model.transform(document_term_matrix)

    mdl = tp.HDPModel()
    for line in preprocessed_inputs:
        mdl.add_doc(line)
    for i in range(0, 100, 10):
        mdl.train(10)

    LDA_model=mdl.convert_to_lda()[0]

    sentence_topic_probabilities=[]
    topic_to_words_mapper={}
    for i in range(len(preprocessed_inputs)):
        sentence_topic_probabilities.append(LDA_model.docs[i].get_topic_dist())
```

```
    sentence_topic_probabilities=np.array(sentence_topic_probabilities)
    for i in range(mdl.k):
        topic_to_words_mapper[i]=[]
        for word in mdl.get_topic_words(i):
            topic_to_words_mapper[i].append(word)



    # Set the threshold probability
    threshold_probability = 0.5

    sentences=list_of_sentences
    topic_assignment_matrix = np.zeros((len(sentences), mdl.k), dtype=int)
    for sentence_idx, sentence_probabilities in enumerate(sentence_topic_probabilities):
        top_topic_idx = sentence_probabilities.argmax()
        top_topic_prob = sentence_probabilities[top_topic_idx]

        topic_assignment = 1 if top_topic_prob > threshold_probability else 0

        topic_assignment_matrix[sentence_idx][top_topic_idx] = topic_assignment
    return
topic_assignment_matrix.copy(),topic_to_words_mapper.copy(),sentence_topic_probabili
ties.copy(),mdl.k


def extract_keywords(text, num_keywords=20):
    model = KeyBERT('distilbert-base-nli-mean-tokens')
    keywords = model.extract_keywords(text, keyphrase_ngram_range=(1, 1),
stop_words='english', top_n=num_keywords)
    return [keyword[0] for keyword in keywords]
def create_sentence_keyword_matrix(sentences, keywords):
    vectorizer = CountVectorizer(vocabulary=keywords, binary=True)
    sentence_matrix = vectorizer.fit_transform(sentences)
    keyword_matrix = sentence_matrix
    return keyword_matrix.toarray()
def create_binary_matrix(sentences, keywords):
    sentence_matrix = create_sentence_keyword_matrix(sentences, keywords)
    binary_matrix = np.where(sentence_matrix > 0, 1, 0)
    return binary_matrix
def retrieve_kw_matrix(revised_text_file_content,list_of_sentences,num_keywords):
    document=revised_text_file_content
    extracted_keywords = extract_keywords(document, num_keywords)
    binary_matrix = create_binary_matrix(list_of_sentences, extracted_keywords)
    kw_matrix=binary_matrix.copy()
    return kw_matrix
def wik_sim_calculate(preprocessed_inputs, wik_dict):
    wik_sim={}
    for i in range(len(preprocessed_inputs)):
```

```
        for j in range(len(preprocessed_inputs)):
            if i!=j:
                wik_sim[(i,j)]=cos_sim(list(wik_dict[i].values()),list(wik_dict[j].values()))
            else:
                wik_sim[(i,j)]=0

    return wik_sim

def o_wik_sim_calculate(preprocessed_inputs,o_dict,wik_dict):
    o_wik_sim={}
    for i in range(len(preprocessed_inputs)):
        o_wik_sim[i]=cos_sim(list(wik_dict[i].values()),list(o_dict.values()))
    return o_wik_sim


class kw_topic_information_class:
    def
__init__(self,preprocessed_inputs,topic_to_words_mapper,topic_assignment_matrix,key
words,wik_dict,o_dict):
        self.preprocessed_inputs=preprocessed_inputs
        self.topic_to_words_mapper=topic_to_words_mapper
        self.topic_assignment_matrix=topic_assignment_matrix
        self.keywords=keywords

self.kw_multipliers_dict=self.kw_multipliers_init(self.preprocessed_inputs,self.keywords
)

self.topic_multipliers_dict=self.topic_multipliers_init(self.preprocessed_inputs,self.topic_
to_words_mapper,self.topic_assignment_matrix)
        self.wik_sim=self.wik_sim_calculate(preprocessed_inputs,wik_dict)
        self.o_wik_sim=self.o_wik_sim_calculate(preprocessed_inputs,o_dict,wik_dict)



    def wik_sim_calculate(self, preprocessed_inputs, wik_dict):
        wik_sim={}
        for i in range(len(preprocessed_inputs)):
            for j in range(len(preprocessed_inputs)):
                if i!=j:
                    wik_sim[(i,j)]=cos_sim(list(wik_dict[i].values()),list(wik_dict[j].values()))
                else:
                    wik_sim[(i,j)]=0

        return wik_sim

    def o_wik_sim_calculate(self,preprocessed_inputs,o_dict,wik_dict):
        o_wik_sim={}
        for i in range(len(preprocessed_inputs)):
            o_wik_sim[i]=cos_sim(list(wik_dict[i].values()),list(o_dict.values()))
```

```
        return o_wik_sim


    def
topic_multipliers_init(self,preprocessed_inputs,topic_to_words_mapper,topic_assignment
_matrix):
        topic_multipliers_dict={}
        for i in range(len(preprocessed_inputs)):
            for j in range(len(preprocessed_inputs)):
                if i!=j:
                    terms_i=preprocessed_inputs[i]
                    terms_i_set=set(terms_i)
                    terms_j=preprocessed_inputs[j]
                    terms_j_set=set(terms_j)


topic_i=topic_to_words_mapper[np.argmax(np.array(topic_assignment_matrix[i]))]
                    topic_i_set=set(topic_i)

topic_j=topic_to_words_mapper[np.argmax(np.array(topic_assignment_matrix[j]))]
                    topic_j_set=set(topic_j)
                    denom=len(topic_i_set.union(topic_j_set))

                    num=len(terms_i_set.intersection(terms_j_set,topic_i_set,topic_j_set))
                    topic_multipliers_dict[(i,j)]=((num+1)/(denom+1))
        return topic_multipliers_dict
    def kw_multipliers_init(self,preprocessed_inputs,keywords):
        kw_multipliers_dict={}
        for i in range(len(preprocessed_inputs)):
            for j in range(len(preprocessed_inputs)):
                if i!=j:
                    vec_i=keywords[i]
                    vec_j=keywords[j]
                    num_matches = np.sum(np.logical_and(vec_i == 1, vec_j == 1))
                    num_mismatches = np.sum(np.logical_xor(vec_i == 1, vec_j == 1))
                    kw_multipliers_dict[(i,j)]=((num_matches+1)/(num_mismatches+1))
        return kw_multipliers_dict

def rouge_score_calc(extract,list_of_sentences,all_rouge_scores):
    rouge_score_dict={}

rouge_score_dict["consensus"]={"candidate":all_rouge_scores.consensus_extract.mem,"o
bjective":all_rouge_scores.consensus_extract.objective}


rouge_score_dict["largest_hypervolume_extract"]={"candidate":all_rouge_scores.largest
_hypervolume_extract.mem,"objective":all_rouge_scores.largest_hypervolume_extract.o
bjective}
```

```
rouge_score_dict["topmost_coverage_extract"]={"candidate":all_rouge_scores.topmost_coverage_extract.mem,"objective":all_rouge_scores.topmost_coverage_extract.objective}

rouge_score_dict["topmost_redundancy_reduction"]={"candidate":all_rouge_scores.topmost_redundancy_reduction_extract.mem,"objective":all_rouge_scores.topmost_redundancy_reduction_extract.objective}

rouge_score_dict["shortest_point_to_ideal_point_euclidean"]={"candidate":all_rouge_scores.shortest_point_to_ideal_point_euclidean_extract.mem,"objective":all_rouge_scores.shortest_point_to_ideal_point_euclidean_extract.objective}

rouge_score_dict["shortest_point_to_ideal_point_manhattan"]={"candidate":all_rouge_scores.shortest_point_to_ideal_point_manhattan_extract.mem,"objective":all_rouge_scores.shortest_point_to_ideal_point_manhattan_extract.objective}

rouge_score_dict["shortest_point_to_ideal_point_chebyshev"]={"candidate":all_rouge_scores.shortest_point_to_ideal_point_chebyshev_extract.mem,"objective":all_rouge_scores.shortest_point_to_ideal_point_chebyshev_extract.objective}


rouge_score_dict["shortest_point_to_ideal_point_mahalanobis"]={"candidate":all_rouge_scores.shortest_point_to_ideal_point_mahalanobis_extract.mem,"objective":all_rouge_scores.shortest_point_to_ideal_point_mahalanobis_extract.objective}

rouge_score_dict["shortest_point_to_all_points_chebyshev"]={"candidate":all_rouge_scores.shortest_point_to_all_points_chebyshev_extract.mem,"objective":all_rouge_scores.shortest_point_to_all_points_chebyshev_extract.objective}


rouge_score_dict["shortest_point_to_all_points_manhattan"]={"candidate":all_rouge_scores.shortest_point_to_all_points_manhattan_extract.mem,"objective":all_rouge_scores.shortest_point_to_all_points_manhattan_extract.objective}

rouge_score_dict["shortest_point_to_all_points_mahalanobis"]={"candidate":all_rouge_scores.shortest_point_to_all_points_mahalanobis_extract.mem,"objective":all_rouge_scores.shortest_point_to_all_points_mahalanobis_extract.objective}

rouge_score_dict["shortest_point_to_all_points_euclidean"]={"candidate":all_rouge_scores.shortest_point_to_all_points_euclidean_extract.mem,"objective":all_rouge_scores.shortest_point_to_all_points_euclidean_extract.objective}

rouge_score_dict["shortest_point_to_all_points_levenshtein"]={"candidate":all_rouge_scores.shortest_point_to_all_points_levenshtein_extract.mem,"objective":all_rouge_scores.shortest_point_to_all_points_levenshtein_extract.objective}

    for k,v in rouge_score_dict.items():
        predicted_extract=""
        candidate=v["candidate"]
        for i in range(len(candidate)):
```

```
            if candidate[i]==1:
                predicted_extract+= list_of_sentences[i] + " . "

        scorer = rouge_scorer.RougeScorer(['rouge1', 'rouge2', 'rougeL'],
use_stemmer=True)
        scores = scorer.score(extract, predicted_extract)

        v["predicted_extract"]=predicted_extract

r1_d={"precision":scores["rouge1"][0],"recall":scores["rouge1"][1],"fmeasure":scores["r
ouge1"][2]}

r2_d={"precision":scores["rouge2"][0],"recall":scores["rouge2"][1],"fmeasure":scores["r
ouge2"][2]}

rl_d={"precision":scores["rougeL"][0],"recall":scores["rougeL"][1],"fmeasure":scores["r
ougeL"][2]}
        v["rouge_score_detail"]={"rouge1":r1_d,"rouge2":r2_d,"rougeL":rl_d}


    return rouge_score_dict


def extract_single_result(NDS_archive,doc_info):
    list_of_sentences=doc_info.list_of_sentences
    L=doc_info.L
    epsilon=doc_info.epsilon
    all_rouge_scores=rouge_score_details(NDS_archive,doc_info)
    return all_rouge_scores

def add_result_line(file_path, text):
    with open(file_path, 'a') as file:
        file.write(text + '\n')


def return_current_datetime_string():
    current_datetime = datetime.now()
    current_date_formatted = current_datetime.strftime('%d-%m-%y')
    current_time = datetime.now().time()
    hours_minutes = current_time.strftime('%H-%M')
    date_time_string=current_date_formatted+"_"+hours_minutes
    return date_time_string

def repair_all(bees_list,doc_info):
    new_bees_list=[]

    for bee in bees_list:
        time1=timeit.default_timer()
        repaired_mem=fast_repair_operator(bee.mem,doc_info)[0]
```

```
        new_bees_list.append(member(repaired_mem,doc_info))
        time2=timeit.default_timer()
#         print("Repair for this bee : ",time2-time1)
    return new_bees_list.copy()
def textrank_scores_calculate(wik_dict,list_of_sentences):
    similarity_matrix = np.zeros([len(list_of_sentences), len(list_of_sentences)])
    for i,row in wik_dict.items():
        for j,column in wik_dict.items():
            row_embedding=list(row.values())
            column_embedding=list(column.values())
            similarity_matrix[i][j]=cos_sim(row_embedding,column_embedding)
    nx_graph = nx.from_numpy_array(similarity_matrix)
    scores = nx.pagerank(nx_graph)
    textrank_scores=dict(sorted(scores.items(), key=lambda x:x[1],reverse=True))
    return textrank_scores

def
sentence_scores_calculation(tfik_dict,term_to_index_mapper,index_to_term_mapper):
    total_term_occ_dict={}
    for i in index_to_term_mapper.keys():
        total_term_occ_dict[i]=0

    for k,v in tfik_dict.items():
        for word,word_count in v.items():
            total_term_occ_dict[term_to_index_mapper[word]] += word_count


    scores_dict={}
    for i in tfik_dict.keys():
        scores_dict[i]=0
    for sentence_index,v in tfik_dict.items():
        for word,word_count in v.items():

scores_dict[sentence_index]+=total_term_occ_dict[term_to_index_mapper[word]]

    for sentence_index,v in scores_dict.items():
        scores_dict[sentence_index]/=10
    return scores_dict
def lexrank_model(dataset_access_path):

    documents = []
    documents_dir = Path(dataset_access_path)
    for file_path in documents_dir.files('*full_text.txt'):
        with file_path.open(mode='rt', encoding='utf-8') as fp:
            documents.append(fp.readlines())

    lxr = LexRank(documents, stopwords=STOPWORDS['en'])

    return lxr
```

```
class scores_class:
    def __init__(self,tf_scores,textrank_scores,graph_scores,lexrank_scores):
        self.tf_scores=tf_scores
        self.textrank_scores=textrank_scores
        self.graph_scores=graph_scores
        self.lexrank_scores=lexrank_scores



class tracker:
    def
__init__(self,population_members,time_taken,NDS_archive_ct,NDS_archive_growth,arc
hive_tracker):
        self.population_members=population_members
        self.time_taken=time_taken
        self.NDS_archive_ct=NDS_archive_ct
        self.NDS_archive_growth=NDS_archive_growth
        self.archive_tracker=archive_tracker
def run_MOABC(doc_info):
    NDS_archive = []
    archive_tracker = {}
    all_sols = []
    cycles=doc_info.cycles
    list_of_members = initialize_population(doc_info)
    population_tracker=[]
    for cycle in range(cycles):
        archive_format = {"employed": [], "onlooker": [], "scoutbeforerepair": [],
"scoutafterrepair": []}

        prev_ct = len(NDS_archive)
        start = timeit.default_timer()

        mutated_member_list = send_employed_bees(list_of_members.copy(), doc_info)
        NDS_archive = NDS_insert_ver_6(NDS_archive, mutated_member_list,
doc_info).copy()
        archive_format["employed"] = NDS_archive.copy()

        onlooker_bee_list = send_onlooker_bees(mutated_member_list, doc_info)
        NDS_archive = NDS_insert_ver_6(NDS_archive, onlooker_bee_list,
doc_info).copy()
        archive_format["onlooker"] = NDS_archive.copy()

        ranked_list_of_members = rank_and_crowd_onlookers(onlooker_bee_list, doc_info)
        scout_bees_list = send_scout_bees(ranked_list_of_members, doc_info)
        NDS_archive = NDS_insert_ver_6(NDS_archive, scout_bees_list, doc_info).copy()
        archive_format["scoutbeforerepair"] = NDS_archive.copy()
```

```
        scout_bees_list = repair_all(scout_bees_list.copy(), doc_info).copy()
        NDS_archive = NDS_insert_ver_6(NDS_archive, scout_bees_list, doc_info).copy()
        archive_format["scoutafterrepair"] = NDS_archive.copy()
        list_of_members = scout_bees_list.copy()

        end = timeit.default_timer()
        new_ct = len(NDS_archive)
        print("Cycle = ", cycle, "Completed, Len of Archive =", len(NDS_archive), ",  No of
solns added/removed = ", new_ct - prev_ct, "Time taken : ",round(end-start,2))

        archive_tracker[cycle] = archive_format
        doc_info.NDS_archive=NDS_archive
        population_tracker.append(tracker(list_of_members,round(end-
start,2),new_ct,new_ct-prev_ct,archive_format))

    return NDS_archive.copy(), archive_tracker,population_tracker




def NSGA_replace(list_of_members,current_mem,new_mem,i):
    result=dominate(current_mem.fitness,new_mem.fitness)
    if result==0:
        print("Not",current_mem.fitness,new_mem.fitness)
    elif result==1:
        print("WON",current_mem.fitness,new_mem.fitness)
    elif result=="False":
        #print("Non Dominating each other",current_mem.fitness,new_mem.fitness)
        pass

    list_of_members[i]=new_mem
    return list_of_members


def NSGA_mutation(list_of_members, doc_info):
    start_time = time.time()  # Start timing the function
    no_of_members = len(list_of_members)
    l = []
    for i in range(no_of_members):
        current_mem = list_of_members[i]
#         time_before_mutation = time.time()
        new_mem_list = mutation_operator_select(current_mem.mem.copy(), doc_info)
#         time_after_mutation = time.time()
        new_mem = member(new_mem_list.copy(), doc_info)
#         time_after_member_creation = time.time()
```

```
list_of_members=replacement_operator_select(list_of_members.copy(),current_mem,ne
w_mem,i,doc_info)
#         time_after_replace = time.time()

#         print("Time for mutation:", time_after_mutation - time_before_mutation)
#         print("Time for member creation:", time_after_member_creation -
time_after_mutation)
#         print("Time for replace", time_after_replace - time_after_member_creation)

    return list_of_members

def BCTSO(parent_members,child_members,doc_info):
    list_of_members=parent_members.copy()
    list_of_members.extend(child_members)
    no_of_objectives=doc_info.no_of_objectives
    front_info_list=list(front_info_former(list_of_members,doc_info).values())

    child_population=[]
    for i in range(len(parent_members)):
        tournament_pair=random.choice(list(combinations(range(len(list_of_members)),
2)))
        mi=tournament_pair[0]
        mj=tournament_pair[1]
        if front_info_list[mi].rank !=front_info_list[mj].rank:
            if front_info_list[mi].rank<front_info_list[mj].rank:
                #print(list_of_members[mi].objective,front_info_list[mi].rank,
list_of_members[mj].objective,front_info_list[mj].rank)
                child_population.append(list_of_members[mi])
            else:
                #print(list_of_members[mi].objective,front_info_list[mi].rank,
list_of_members[mj].objective,front_info_list[mj].rank)
                child_population.append(list_of_members[mj])
        elif front_info_list[mi].rank==front_info_list[mj].rank:
            if front_info_list[mi].cd==front_info_list[mj].cd:

child_population.append(random.choice([list_of_members[mi],list_of_members[mj]]))
            elif front_info_list[mi].cd<front_info_list[mj].cd:
                child_population.append(list_of_members[mj])
            else:
                child_population.append(list_of_members[mi])

    return child_population.copy()

def run_NSGA(doc_info):
    NDS_archive = []
    archive_tracker = {}
    all_sols = []
    cycles=doc_info.cycles
```

```
    list_of_members = initialize_population(doc_info)
    population_tracker=[]
    for cycle in range(cycles):

        prev_ct = len(NDS_archive)
        archive_format = {"final": []}

        start = timeit.default_timer()

        list_of_members=repair_all(list_of_members.copy(),doc_info).copy()
        LOF=list_of_members.copy()
        mutated_member_list = NSGA_mutation(LOF, doc_info)
        mutated_member_list = repair_all(mutated_member_list.copy(), doc_info)
        NDS_archive = NDS_insert_ver_6(NDS_archive, list_of_members, doc_info)
        NDS_archive = NDS_insert_ver_6(NDS_archive, mutated_member_list, doc_info)


list_of_members=BCTSO(list_of_members.copy(),mutated_member_list.copy(),doc_info
)
#         NDS_archive = NDS_insert_ver_6(NDS_archive, list_of_members,
doc_info).copy()
        archive_format["final"]=NDS_archive.copy()

        new_ct = len(NDS_archive)
        end=timeit.default_timer()
        print("Cycle = ", cycle, "Completed, Len of Archive =", len(NDS_archive), ",  No of
solns added/removed = ", new_ct - prev_ct, "Time taken : ",round(end-start,2))
        archive_tracker[cycle] = archive_format
        doc_info.NDS_archive=NDS_archive

        population_tracker.append(tracker(list_of_members,round(end-
start,2),new_ct,new_ct-prev_ct,archive_format))

    return NDS_archive.copy(), archive_tracker,population_tracker

def transform_tvectors(sentence_probabilities):
    wik_dict={}
    o_dict={}
    o_vec=np.mean(sentence_probabilities.T,axis=1)
    for i in range(len(sentence_probabilities)):
        wik_dict[i]={}
        for k in range(len(sentence_probabilities[i])):
            wik_dict[i][k]=sentence_probabilities[i][k]
    for i in range(len(sentence_probabilities[0])):
        o_dict[i]=o_vec[i]
    return wik_dict,o_dict
```

```
def rcd_based_replacement(list_of_members,current_mem,new_mem,i,doc_info):
    temp_list=list_of_members.copy()
    temp_list.extend([new_mem])
    no_of_objectives=doc_info.no_of_objectives
    front_info_list=list(front_info_former(temp_list,doc_info).values())

    mi=i
    mj=len(temp_list)-1
    if front_info_list[mi].rank !=front_info_list[mj].rank:
        if front_info_list[mi].rank<front_info_list[mj].rank:
            list_of_members[i].trials+=1
            pass
        else:
            list_of_members=naive_replace(list_of_members,current_mem,new_mem,i)
    elif front_info_list[mi].rank==front_info_list[mj].rank:
        if front_info_list[mi].cd==front_info_list[mj].cd:
            list_of_members[i].trials+=1
            pass
        elif front_info_list[mi].cd<front_info_list[mj].cd:
            list_of_members=naive_replace(list_of_members,current_mem,new_mem,i)
        else:
            list_of_members[i].trials+=1
            pass

    return list_of_members

def dominate_based_replacement(list_of_members,current_mem,new_mem,i,doc_info):
    result=dominate(current_mem.fitness,new_mem.fitness)
    if result==0:
        list_of_members[i].trials+=1
    elif result==1:
        list_of_members=naive_replace(list_of_members,current_mem,new_mem,i)
    elif result=="False":
        list_of_members[i].trials+=1
    return list_of_members

def
epsilon_dominate_based_replacement(list_of_members,current_mem,new_mem,i,doc_in
fo):
    result=dominate(current_mem.fitness,new_mem.fitness)
    if result==1:
        list_of_members=naive_replace(list_of_members,current_mem,new_mem,i)
    else:
        #in cases of Nondominating or 0 conditions
        percents={}
        validity={}
        percent_imp=1
        percent_dec=-0.25
```

```
        winners=0
        no_of_objectives=doc_info.no_of_objectives
        for index in range(no_of_objectives):
            percents[index]=(new_mem.fitness[index]-
current_mem.fitness[index])/current_mem.fitness[index]
            if new_mem.fitness[index] >= current_mem.fitness[index]:
                # %improvement case
                if percents[index] >= percent_imp:
                    validity[index]=True
                    winners+=1
                else:
                    validity[index]=False

            else :
                if percents[index] >= percent_dec:
                    validity[index]=True
                else:
                    validity[index]=False
        if list(validity.values()) == [True] *no_of_objectives and winners>=
int(no_of_objectives/2):
            print("Yeah!")
            list_of_members=naive_replace(list_of_members,current_mem,new_mem,i)
        else:
            list_of_members[i].trials+=1

    return list_of_members

def replacement_operator_select(list_of_members,current_mem,new_mem,i,doc_info):
    replacement_operator_version=doc_info.replacement_operator_version
#     if doc_info.model_version==
    if replacement_operator_version=="rcd":
        return rcd_based_replacement(list_of_members,current_mem,new_mem,i,doc_info)
    elif replacement_operator_version=="dominate":
        return
dominate_based_replacement(list_of_members,current_mem,new_mem,i,doc_info)
    elif replacement_operator_version=="epsilon_dominate":
        return
epsilon_dominate_based_replacement(list_of_members,current_mem,new_mem,i,doc_in
fo)


class saver_class:

#full_text_file_path,greedy_extract_file_path,original_text_content,L,revised_text_file_c
ontent,preprocessed_inputs,preprocessed_inputs_no_checks,list_of_sentences,index_to_s
entence_info_mapper,tfik_dict,no_of_sentences,terms_dict,no_of_terms,index_to_term_
mapper,term_to_index_mapper,wik_dict,o_dict,kw_topic_info,
```

```
    def
__init__(self,doc_info,NDS_archive,greedy_extract_text,rouge_score,NDS_archive_trac
ker,params,population_tracker):
        self.doc_info=doc_info
        self.NDS_archive=NDS_archive
        self.greedy_extract_text=greedy_extract_text
        self.rouge_score=rouge_score
        self.NDS_archive_tracker=NDS_archive_tracker
        self.params=params
        self.population_tracker=population_tracker

def save_details(saver_class,path,version,datetime_string,article_id):
    with open(path+version+"_"+article_id+".pickle","wb") as file:
        pickle.dump(saver_class,file)


class file_paths:
    def
__init__(self,base_path_input,base_path_output,version,current_datetime_string,kaggle):
        if kaggle==True:
            self.base_path_input=base_path_input
            self.dataset_path=self.base_path_input +"/datasets/"
            self.dataset_access_path=self.base_path_input +"/datasets/individual_1/"



            self.base_path_output=base_path_output
            self.result_path=self.base_path_output +"/results/"
            self.details_path=self.base_path_output+"/details/"

            self.version=version
            self.current_datetime_string=current_datetime_string
            self.result_file_name=self.version+"_"+current_datetime_string +".txt"
        else:
            self.base_path=base_path_input
            self.result_path=self.base_path +"\\results\\"
            self.dataset_path=self.base_path +"\\datasets\\"
            self.dataset_access_path=self.base_path +"\\datasets\\individual_1\\"
            self.details_path=self.base_path+"\\details\\"
            self.version=version
            self.current_datetime_string=current_datetime_string
            self.result_file_name=self.version+"_"+current_datetime_string +".txt"

class compressed_doc_info_class:
    def __init__(self,preprocessed_inputs,list_of_sentences,L,epsilon):
        self.preprocessed_inputs=preprocessed_inputs
        self.list_of_sentences=list_of_sentences
        self.L=L
        self.epsilon=epsilon
class doc_info_class:
```

```
    def __init__(self,
full_text_file_path,greedy_extract_file_path,original_text_content,L,revised_text_file_co
ntent,preprocessed_inputs,list_of_sentences,article_id,tfik_dict,no_of_sentences,terms_di
ct,no_of_terms,index_to_term_mapper,term_to_index_mapper,wik_dict,o_dict,kw_topic
_info,scores_details,sentence_lengths,params,sentence_probabilities,file_paths_class,HD
P_topics):
        self.full_text_file_path=full_text_file_path
        self.greedy_extract_file_path=greedy_extract_file_path
        self.original_text_content=original_text_content
        self.L=L
        self.revised_text_file_content=revised_text_file_content
        self.preprocessed_inputs=preprocessed_inputs
#         self.preprocessed_inputs_no_checks=preprocessed_inputs_no_checks
        self.list_of_sentences=list_of_sentences
        #self.index_to_sentence_info_mapper=index_to_sentence_info_mapper
        self.article_id=article_id
        self.tfik_dict=tfik_dict
        self.no_of_sentences=no_of_sentences
        self.terms_dict=terms_dict
        self.no_of_terms=no_of_terms
        self.index_to_term_mapper=index_to_term_mapper
        self.term_to_index_mapper=term_to_index_mapper
        self.wik_dict=wik_dict
        self.o_dict=o_dict
        self.kw_topic_info=kw_topic_info
        self.o_wik_sim=self.kw_topic_info.o_wik_sim
        self.wik_sim=self.kw_topic_info.wik_sim
        self.scores_details=scores_details
        self.no_of_objectives=params.no_of_objectives
        self.doc_threshold_value=params.doc_threshold_value
        self.sentence_threshold_value=params.sentence_threshold_value
        self.spacy_use=params.spacy_use
        self.restricted_docs_list=params.restricted_docs_list
        self.epsilon=params.epsilon
        self.L=params.L
        self.no_of_objectives=params.no_of_objectives
        self.pop_size=params.pop_size
        self.pm=params.pm
        self.trial_cutoff_limit=params.trial_cutoff_limit
        self.cycles=params.cycles
        self.sentence_lengths=sentence_lengths
        self.delta=params.delta
        self.NDS_archive=[]
        self.t_wik_dict,self.t_o_dict=transform_tvectors(sentence_probabilities)
        self.t_wik_sim=wik_sim_calculate(self.preprocessed_inputs,self.t_wik_dict)

self.t_o_wik_sim=o_wik_sim_calculate(self.preprocessed_inputs,self.t_o_dict,self.t_wik
_dict)
        self.mutation_operator_version=params.mutation_operator_version
```

```
        self.replacement_operator_version=params.replacement_operator_version
        self.algorithm=params.algorithm
        self.IP_population_initialization=params.IP_population_initialization
        self.topic_enabled=params.topic_enabled
        self.sbert_enabled=params.sbert_enabled
        self.kaggle=params.kaggle
        self.run=params.run
        self.kaggle=params.kaggle
        self.delim=params.delim

        self.result_path=file_paths_class.result_path
        self.dataset_path=file_paths_class.dataset_path
        self.result_file_name=file_paths_class.result_file_name
        self.version=file_paths_class.version
        self.details_path=file_paths_class.details_path
        self.dataset_access_path=file_paths_class.dataset_access_path

        self.HDP_topics=HDP_topics

        self.csv_name=params.csv_name


class GETS:
    def __init__(self,doc_info):
        self.doc_info=doc_info
        self.edge_weights=self.edge_weights_computation(doc_info)

self.sentence_weights=self.sentence_weights_computation(doc_info,self.edge_weights)

self.average_weight=self.average_weight_computation(doc_info,self.sentence_weights)

self.summary,self.summary_length=self.gets_summarizer(doc_info,self.sentence_weights
,self.average_weight)
        self.gets_sentence_scores=self.return_GETS_scores(self.sentence_weights)
    def edge_weights_computation(self,doc_info):
        edge_weights={}
        preprocessed_inputs=doc_info.preprocessed_inputs
        for i in range(len(preprocessed_inputs)):
            for j in range(len(preprocessed_inputs)):
                if i!=j :

edge_weights[(i,j)]=len(set(preprocessed_inputs[i]).intersection(preprocessed_inputs[j]))
                else:
                    edge_weights[(i,j)]=0

        return edge_weights

    def sentence_weights_computation(self,doc_info,edge_weights):
        sentence_weights={}
```

```
        preprocessed_inputs=doc_info.preprocessed_inputs
        for i in range(len(preprocessed_inputs)):
            sentence_weights[i]=0

        for i in range(len(preprocessed_inputs)):
            for j in range(i+1,len(preprocessed_inputs)):
                sentence_weights[i]+=edge_weights[(i,j)]
        return sentence_weights

    def average_weight_computation(self,doc_info,sentence_weights):
        preprocessed_inputs=doc_info.preprocessed_inputs
        weighted_sum=0
        for i in range(len(preprocessed_inputs)):
            weighted_sum+=sentence_weights[i]

        weighted_sum/=len(preprocessed_inputs)
        weighted_sum=math.ceil(weighted_sum)

        return weighted_sum

    def gets_summarizer(self,doc_info,sentence_weights,average_weight):
        Lmax=doc_info.L+doc_info.epsilon
        Lmin=doc_info.L-doc_info.epsilon
        new_revised_sentences={}
        for k,v in sentence_weights.items():
            if v >= average_weight:
                new_revised_sentences[k]=v
        sorted_sentences= dict(sorted(new_revised_sentences.items(), key=lambda item:
item[1],reverse=True))

        cur_summary=""
        cur_summary_length=0
        list_of_sentences=doc_info.list_of_sentences
        for k,v in sorted_sentences.items():
            if cur_summary_length+len(list_of_sentences[k].split()) >Lmax:
                break
            else:
                cur_summary_length+=len(list_of_sentences[k].split())
                cur_summary+=list_of_sentences[k]+" . "

        return cur_summary,cur_summary_length

    def return_GETS_scores(self,sentence_weights):
        sorted_sentence_weights= dict(sorted(sentence_weights.items(), key=lambda item:
item[1],reverse=True))
        return sorted_sentence_weights
```

```
special_characters = ['!', '@', '#', '$', '%', '^', '&', '*', '(', ')', '_', '+', '-', '=', '{', '}', '[', ']', '|', '\\',
':', ';', '<', '>', ',', '/', '?', "" ,"`","``","''" ]

def clean_text(input_string):
    translation_table = str.maketrans("", "", "".join(special_characters))
    cleaned_string = input_string.translate(translation_table)
    return cleaned_string

def print_pattern(text=""):
    print(text)
    print("**************************************************")

def calculate_sentence_lengths(list_of_sentences):
    sentence_lengths_dict={}
    for i in range(len(list_of_sentences)):
        sentence_lengths_dict[i]=len(list_of_sentences[i].split())
    return sentence_lengths_dict


def save_csv_details(doc_info,rouge_score):
    csv_row=[]
    csv_row.append(doc_info.mutation_operator_version)
    csv_row.append(doc_info.replacement_operator_version)
    csv_row.append(doc_info.algorithm)
    csv_row.append(doc_info.IP_population_initialization)
    csv_row.append(doc_info.topic_enabled)
    csv_row.append(doc_info.sbert_enabled)
    csv_row.append(doc_info.run)
    csv_row.append(doc_info.article_id)
    csv_row.append(doc_info.HDP_topics)


    for k,v in rouge_score.items():
        cur_row=csv_row.copy()
        cur_row.append(k)
        rouge_scores_row_dict=v["rouge_score_detail"]
        cur_row.append(rouge_scores_row_dict['rouge1']['precision'])
        cur_row.append(rouge_scores_row_dict['rouge1']['recall'])
        cur_row.append(rouge_scores_row_dict['rouge1']['fmeasure'])
        cur_row.append(rouge_scores_row_dict['rouge2']['precision'])
        cur_row.append(rouge_scores_row_dict['rouge2']['recall'])
        cur_row.append(rouge_scores_row_dict['rouge2']['fmeasure'])
        cur_row.append(rouge_scores_row_dict['rougeL']['precision'])
        cur_row.append(rouge_scores_row_dict['rougeL']['recall'])
        cur_row.append(rouge_scores_row_dict['rougeL']['fmeasure'])
```

```
        with
open(doc_info.result_path+doc_info.delim+doc_info.csv_name+'_final_results.csv', 'a')
as f_object:
            writer_object = writer(f_object)
            writer_object.writerow(cur_row)
            f_object.close()

def process_all_files(params,file_paths_class):

    result_path=file_paths_class.result_path
    dataset_path=file_paths_class.dataset_path
    result_file_name=file_paths_class.result_file_name
    version=file_paths_class.version
    details_path=file_paths_class.details_path
    dataset_access_path=file_paths_class.dataset_access_path


    fulltext_file_names={}
    extracts_file_names={}
    with open(dataset_path+"extracts_duc_filenames.pickle","rb") as file:
        extracts_file_names=pickle.load(file)
    with open(dataset_path+"fulltext_duc_filenames.pickle","rb") as file:
        fulltext_file_names=pickle.load(file)
    no_of_docs=10
    ctr=0
    docs_processed=0
    while docs_processed<no_of_docs:
        print_pattern()
        doc_name=list(fulltext_file_names.keys())[ctr]
        L=params.L
        full_text_file_path=dataset_access_path+fulltext_file_names[doc_name]

greedy_extract_file_path=dataset_access_path+extracts_file_names[doc_name][L][0]
        article_id=doc_name
        print
        print("Checking if doc : ",doc_name,"passes all preprocessing steps ::....")
        if article_id in params.restricted_docs_list:
#int(article_id)<=params.doc_threshold_value or
            print_pattern("Present in Restricted Docs List....Skipping...")
            ctr+=1
            continue
        if ctr <= params.doc_threshold_value :
            print_pattern("Present under Doc Restricted Value.....Skipping....")
            ctr+=1
            continue

        original_text_content=read_text_file(full_text_file_path)
        revised_text_file_content=remove_specials(original_text_content)
        cleaned_text=clean_text(revised_text_file_content)
```

```
preprocessed_inputs,list_of_sentences_via_preprocess=preprocess(cleaned_text,params.s
pacy_use)
      list_of_sentences=list_of_sentences_via_preprocess.copy()


      greedy_extract_text=read_text_file(greedy_extract_file_path)
      greedy_extract_text=remove_specials(greedy_extract_text)
      greedy_extract_text=clean_text(greedy_extract_text)
      greedy_list_of_sentences=segmentation(greedy_extract_text)
      greedy_extract_text=".".join(greedy_list_of_sentences)


      if len(list_of_sentences_via_preprocess)==len(preprocessed_inputs):
         print("Preprocessing Step Cleared Successfully")
      else:
         print_pattern("Cardinality of Sentences not matched, Please rectify doc structure,
Skipping...!!!")
         ctr+=1
         continue
         raise ValueError("Stop!")
      print("No of sentences = ",len(preprocessed_inputs))
      if len(preprocessed_inputs)>params.sentence_threshold_value:
         print_pattern("Too Long-Skipping.....")
         ctr+=1
         continue

      print("Document : ",article_id,"Processing !!!----------")

      tfik_dict={}
      for i in range(len(preprocessed_inputs)):
         tfik_dict[i]=tfik(preprocessed_inputs[i])

      no_of_sentences=len(preprocessed_inputs)
      terms_dict=distinct_terms(tfik_dict)
      no_of_terms=len(list(terms_dict.keys()))

      index_to_term_mapper, term_to_index_mapper=indices_terms_mapper(terms_dict)

term_index_sentence_dict=term_indices_of_sentences(preprocessed_inputs,term_to_inde
x_mapper)
      if params.sbert_enabled==True:
         o_dict,wik_dict=SBERT_embeddings(list_of_sentences)
      elif params.sbert_enabled==False:

wik_dict=wik(tfik_dict,terms_dict,index_to_term_mapper,term_to_index_mapper,no_of_
sentences,no_of_terms)
         o_dict=mean_o(wik_dict,no_of_sentences,no_of_terms)
```

```
topic_assignment_matrix,topic_to_words_mapper,sentence_probabilities,HDP_topics=ret
rieve_topic_assignment_matrix(list_of_sentences,preprocessed_inputs,
params.num_topics,params.n_top_words)

kw_matrix=retrieve_kw_matrix(revised_text_file_content,list_of_sentences,params.num_
keywords)

kw_topic_info=kw_topic_information_class(preprocessed_inputs,topic_to_words_mappe
r,topic_assignment_matrix,kw_matrix,wik_dict,o_dict)


tf_scores=sentence_scores_calculation(tfik_dict,term_to_index_mapper,index_to_term_
mapper)
    textrank_scores=textrank_scores_calculate(wik_dict,list_of_sentences)

compressed_doc_info=compressed_doc_info_class(preprocessed_inputs,list_of_sentence
s,params.L,params.epsilon)
    graph_scores=GETS(compressed_doc_info).gets_sentence_scores
    lxr=lexrank_model(dataset_access_path)
    lexrankscores_cont =
lxr.rank_sentences(list_of_sentences,threshold=None,fast_power_method=False)
    lexrankscores_dict={}
    for i in range(len(preprocessed_inputs)):
        lexrankscores_dict[i]=lexrankscores_cont[i]
    lexrankscores_dict = dict(sorted(lexrankscores_dict.items(), key=lambda item:
item[1],reverse=True))

scores_details=scores_class(tf_scores,textrank_scores,graph_scores,lexrankscores_dict)


    sentence_lengths=calculate_sentence_lengths(list_of_sentences)

doc_info=doc_info_class(full_text_file_path,greedy_extract_file_path,original_text_cont
ent,L,revised_text_file_content,preprocessed_inputs,list_of_sentences,article_id,tfik_dict,
no_of_sentences,terms_dict,no_of_terms,index_to_term_mapper,term_to_index_mapper,
wik_dict,o_dict,kw_topic_info,scores_details,sentence_lengths,params,sentence_probabil
ities,file_paths_class,HDP_topics)

    if doc_info.algorithm=="MOABC":
        NDS_archive,NDS_archive_tracker,population_tracker=run_MOABC(doc_info)
    elif doc_info.algorithm=="NSGA":
        NDS_archive,NDS_archive_tracker,population_tracker=run_NSGA(doc_info)

    all_rouge_scores=extract_single_result(NDS_archive,doc_info)


rouge_score=rouge_score_calc(greedy_extract_text,list_of_sentences,all_rouge_scores)

    add_result_line(result_path+result_file_name,article_id + ":     "+str(rouge_score))
```

```
save_details(saver_class(doc_info,NDS_archive,greedy_extract_text,rouge_score,NDS_a
rchive_tracker,params,population_tracker),details_path,version,current_datetime_string,a
rticle_id)

        save_csv_details(doc_info,rouge_score)

        docs_processed+=1
        ctr+=1
        print_pattern()




class params_class:
    def
__init__(self,doc_threshold_value,sentence_threshold_value,spacy_use,restricted_docs_li
st,epsilon,L,no_of_objectives,pop_size,pm,trial_cutoff_limit,cycles,delta,num_topics,n_t
op_words,mutation_operator_version,replacement_operator_version,algorithm,IP_popula
tion_initialization,topic_enabled,sbert_enabled,num_keywords,Lextract,run,delim,kaggle,
csv_name):
        self.doc_threshold_value=doc_threshold_value
        self.sentence_threshold_value=sentence_threshold_value
        self.spacy_use=spacy_use
        self.restricted_docs_list=restricted_docs_list
        self.epsilon=epsilon
        self.L=L
        self.no_of_objectives=no_of_objectives
        self.pop_size=pop_size
        self.pm=pm
        self.trial_cutoff_limit=trial_cutoff_limit
        self.cycles=cycles
        self.delta=delta
        self.num_topics=num_topics
        self.n_top_words=n_top_words
        self.mutation_operator_version=mutation_operator_version
        self.replacement_operator_version=replacement_operator_version
        self.algorithm=algorithm
        self.IP_population_initialization=IP_population_initialization
        self.topic_enabled=topic_enabled
        self.sbert_enabled=sbert_enabled
        self.num_keywords=num_keywords
        self.Lextract=Lextract
        self.run=run
        self.delim=delim
```

```
        self.kaggle=kaggle
        self.csv_name=csv_name



def dirs_execute():
    try:
        os.makedirs("/kaggle/working/results")
    except:
        pass

    try:
        os.makedirs("/kaggle/working/details")
    except:
        pass


kaggle=True
if kaggle==True:
    dirs_execute()
    base_path_input="/kaggle/input/duc2002dataandwheels/KaggleWheels"
    base_path_output="/kaggle/working"
    delim="/"
else:
    base_path_input=r"C:\Users\shari\OneDrive\Desktop\Text Sum"
    base_path_output=r"C:\Users\shari\OneDrive\Desktop\Text Sum"
    delim="\\"

dirs_execute()
current_datetime_string=return_current_datetime_string()

replacement_operators=["dominate","rcd","epsilon_dominate"]
mutation_operators=[1,2]
algos=["NSGA","MOABC"]
topic_enabled_versions=["v1","v2","v3","v4"]
IP_population_initialization=False
sbert_enabled=False

csv_name="TopicModellingExperiment_v1_8_10"
num_keywords=30

inde_runs=10

doc_threshold_value=-1
sentence_threshold_value=500
spacy_use=False
cycles=100
delta=0.1
num_topics=5
```

```
n_top_words=20
restricted_docs_list=[]
epsilon=50
L=200
Lextract=200
no_of_objectives=2
pop_size=50
pm=0.4
trial_cutoff_limit=5

mutation_operator_version=mutation_operators[0]
replacement_operator_version=replacement_operators[0]
algorithm=algos[1]
topic_enabled=topic_enabled_versions[0]


for run in range(8,10,1):

version=algorithm+"_"+replacement_operator_version+"_"+"MutationOperatorVersion=
"+str(mutation_operator_version)+"_"+"IndependentRun="+str(run)+"_"+"topic-
enabled="+str(topic_enabled)+"_"+"IP-population-
initialization="+str(IP_population_initialization)+"_"+"sbert-
enabled="+str(sbert_enabled)

file_paths_class=file_paths(base_path_input,base_path_output,version,current_datetime_
string,kaggle)
    print("Version In Progress :::",version)

params=params_class(doc_threshold_value,sentence_threshold_value,spacy_use,restricte
d_docs_list,epsilon,L,no_of_objectives,pop_size,pm,trial_cutoff_limit,cycles,delta,num_t
opics,n_top_words,mutation_operator_version,replacement_operator_version,algorithm,I
P_population_initialization,topic_enabled,sbert_enabled,num_keywords,Lextract,run,deli
m,kaggle,csv_name)
    d = process_all_files(params,file_paths_class)
```

CHAPTER - 7

OUTPUT

The following is the dataset information.

	DocName	No. of Sentences	No. of Words
0	d061j	187	3627
1	d062j	131	2620
2	d063j	247	4743
3	d064j	197	4080
4	d065j	285	5418
5	d066j	198	3828
6	d067f	126	2790
7	d068f	135	2511
8	d069f	333	7697
9	d070f	153	3077

The following image is a sample run from Jupyter Notebook while running MOABC algorithm.

```
**************************************************
Checking if doc :  d061j passes all preprocessing steps ::....
Preprocessing Step Cleared Successfully
No of sentences =  187
Document :  d061j Processing !!!--------------
Cycle =  0 Completed, Len of Archive = 21 ,  No of solns added/removed =  21 Time taken :  8.05
Cycle =  1 Completed, Len of Archive = 26 ,  No of solns added/removed =  5 Time taken :  6.33
Cycle =  2 Completed, Len of Archive = 31 ,  No of solns added/removed =  5 Time taken :  6.61
Cycle =  3 Completed, Len of Archive = 29 ,  No of solns added/removed =  -2 Time taken :  6.64
Cycle =  4 Completed, Len of Archive = 32 ,  No of solns added/removed =  3 Time taken :  7.17
Cycle =  5 Completed, Len of Archive = 33 ,  No of solns added/removed =  1 Time taken :  6.68
Cycle =  6 Completed, Len of Archive = 34 ,  No of solns added/removed =  1 Time taken :  6.59
Cycle =  7 Completed, Len of Archive = 36 ,  No of solns added/removed =  2 Time taken :  6.75
Cycle =  8 Completed, Len of Archive = 39 ,  No of solns added/removed =  3 Time taken :  6.81
Cycle =  9 Completed, Len of Archive = 41 ,  No of solns added/removed =  2 Time taken :  6.65
Cycle =  10 Completed, Len of Archive = 40 ,  No of solns added/removed =  -1 Time taken :  6.59
Cycle =  11 Completed, Len of Archive = 43 ,  No of solns added/removed =  3 Time taken :  6.64
Cycle =  12 Completed, Len of Archive = 45 ,  No of solns added/removed =  2 Time taken :  6.8
Cycle =  13 Completed, Len of Archive = 48 ,  No of solns added/removed =  3 Time taken :  7.17
Cycle =  14 Completed, Len of Archive = 47 ,  No of solns added/removed =  -1 Time taken :  6.93
Cycle =  15 Completed, Len of Archive = 45 ,  No of solns added/removed =  -2 Time taken :  6.75
Cycle =  16 Completed, Len of Archive = 47 ,  No of solns added/removed =  2 Time taken :  6.77
Cycle =  17 Completed, Len of Archive = 48 ,  No of solns added/removed =  1 Time taken :  6.87
Cycle =  18 Completed, Len of Archive = 54 ,  No of solns added/removed =  6 Time taken :  7.0
Cycle =  19 Completed, Len of Archive = 55 ,  No of solns added/removed =  1 Time taken :  7.08
Cycle =  20 Completed, Len of Archive = 57 ,  No of solns added/removed =  2 Time taken :  6.98
Cycle =  21 Completed, Len of Archive = 57 ,  No of solns added/removed =  0 Time taken :  7.53
Cycle =  22 Completed, Len of Archive = 58 ,  No of solns added/removed =  1 Time taken :  7.18
Cycle =  23 Completed, Len of Archive = 61 ,  No of solns added/removed =  3 Time taken :  7.71
Cycle =  24 Completed, Len of Archive = 61 ,  No of solns added/removed =  0 Time taken :  7.2
Cycle =  25 Completed, Len of Archive = 61 ,  No of solns added/removed =  0 Time taken :  7.08
Cycle =  26 Completed, Len of Archive = 63 ,  No of solns added/removed =  2 Time taken :  7.32
Cycle =  27 Completed, Len of Archive = 65 ,  No of solns added/removed =  2 Time taken :  8.2
Cycle =  28 Completed, Len of Archive = 65 ,  No of solns added/removed =  0 Time taken :  7.49
```

Sample Rouge scores are printed in the below figure.

```
Checking if doc :  d070f passes all preprocessing steps ::....
Preprocessing Step Cleared Successfully
No of sentences =  153
Document :  d070f Processing !!!-------------
Cycle =  0 Completed, Len of Archive = 15 ,  No of solns added/removed =  15 Time taken :  1.9
Cycle =  1 Completed, Len of Archive = 18 ,  No of solns added/removed =  3 Time taken :  1.88
Cycle =  2 Completed, Len of Archive = 18 ,  No of solns added/removed =  0 Time taken :  1.68
Cycle =  3 Completed, Len of Archive = 23 ,  No of solns added/removed =  5 Time taken :  1.51
Cycle =  4 Completed, Len of Archive = 22 ,  No of solns added/removed =  -1 Time taken :  1.73
rouge1: Score(precision=0.47368421052631576, recall=0.5, fmeasure=0.4864864864864865)
rouge2: Score(precision=0.21138211382113822, recall=0.22317596566523606, fmeasure=0.21711899791231734)
rougeL: Score(precision=0.2550607287449393, recall=0.2692307692307692, fmeasure=0.26195426195426197)
```

The following is a sample text file log that stores all the rouge scores, members, NDS archive and predicted extracts.

d061j: {'consensus': {'candidate': [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], 'objective': [2.5895746266575808, 0.041464417840380866], 'predicted_extract': 'Tropical Storm Gilbert formed in the eastern Caribbean and strengthened into a hurricane Saturday night . Hurricane Gilbert packing 110 mph winds and torrential rain moved over this capital city today after skirting Puerto Rico Haiti and the Dominican Republic . Right now its actually moving over Jamaica said Bob Sheets director of the National Hurricane Center in Miami . Its certainly one of the larger systems weve seen in the Caribbean for a long time said Hal Gerrish forecaster at the National Hurricane Center . Heavy rain and stiff winds downed power lines and caused flooding in the Dominican Republic on Sunday night as the hurricanes center passed just south of the Barahona peninsula then less than 100 miles from neighboring Haiti . The storm ripped the roofs off houses and flooded coastal areas of southwestern Puerto Rico after reaching hurricane strength off the islands southeast Saturday night . A National Weather Service report said the hurricane was moving west at 17 mph with maximum sustained winds of 115 mph . Havana Radio meanwhile reported Monday that 25 000 people were evacuated from coastal areas in Guantanamo Province on the nations southeastern coast as Gilberts winds and rain began to brush the island . Army officials in Mexico City said about 35 000 people were evacuated from Cancun but Cancun Mayor Jose Sanchez Zapata said about 11 000 fled . Hurricane Gilbert swept toward Jamaica yesterday with 100mileanhour winds and officials issued warnings to residents on the southern coasts of the Dominican Republic Haiti and Cuba . ', 'rouge_score_detail': {'rouge1': {'precision': 0.49193548387096775, 'recall': 0.5980392156862745, 'fmeasure': 0.5398230088495576}, 'rouge2': {'precision': 0.19838056680161945, 'recall': 0.2413793103448276, 'fmeasure': 0.2177777777777778}, 'rougeL': {'precision': 0.25806451612903225, 'recall': 0.3137254901960784, 'fmeasure': 0.2831858407079646}}}, 'largest_hypervolume_extract': {'candidate': array([0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]), 'objective': [1.4467947496788411, 0.22106855772624837], 'predicted_extract': 'The National Weather Service in San Juan Puerto Rico said Gilbert was moving westward at 15 mph with a broad area of cloudiness and heavy weather rotating around the center of the storm . The Associated Press Caribbean headquarters in San Juan Puerto Rico was unable to get phone calls through to Kingston where high winds and heavy rain preceding the storm drenched the capital overnight toppling trees causing local flooding and littering streets with branches . The north coast tourist region from Montego Bay on the west and Ocho Rios on the east far from the southern impact zone and separated by mountains was expected only to receive heavy rain . Its certainly one of the larger systems weve seen in the Caribbean for a long time said Hal Gerrish forecaster at the National Hurricane Center . Prime Minister Edward Seaga of Jamaica alerted all government agencies saying Sunday night Hurricane Gilbert appears to be a real threat and everyone should follow the instructions and hurricane precautions issued by the Office of Disaster Preparedness in order to minimize the danger . Forecasters said the hurricane had been gaining strength as it passed over the ocean after it dumped 5 to 10 inches of rain on the Dominican Republic and Haiti which share the island of Hispaniola . No serious injuries were immediately reported in the city of 750 000 people which was hit by the full force of the hurricane around noon . ', 'rouge_score_detail': {'rouge1': {'precision': 0.43037974683544306, 'recall': 0.5, 'fmeasure': 0.4625850340136054}, 'rouge2': {'precision': 0.11864406779661017, 'recall': 0.13793103448275862, 'fmeasure': 0.12756264236902048}, 'rougeL': {'precision': 0.17721518987341772, 'recall': 0.20588235294117646, 'fmeasure': 0.1904761904761905}}}, 'topmost_coverage_extract': {'candidate': array([0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0]), 'objective': [3.0928523481247243, 0.01149358521321743], 'predicted_extract': 'Tropical Storm Gilbert formed in the eastern Caribbean and strengthened into a hurricane Saturday night . The National Hurricane Center in Miami reported its position at 2 a . Hurricane Gilbert packing 110 mph winds and torrential rain moved over this capital city today after skirting Puerto Rico Haiti and the Dominican Republic . Right now its actually moving over Jamaica said Bob Sheets director of the National Hurricane Center in Miami . Hurricane warnings were posted for the Cayman Islands Cuba and Haiti . Its certainly one of the larger systems weve seen in the Caribbean for a long time said Hal Gerrish forecaster at the National Hurricane Center . The storm ripped the roofs off houses and flooded coastal areas of southwestern Puerto Rico after reaching hurricane strength off the islands southeast Saturday night . Gilbert reached Jamaica after skirting southern Puerto Rico Haiti and the Dominican Republic . Gilbert was moving westnorthwest at 15 mph and winds had decreased to 125 mph . Such storms have maximum sustained winds greater than 155 mph and can cause catastrophic damage . Hurricane Gilbert swept toward Jamaica yesterday with 100mileanhour winds and officials issued warnings to residents on the southern coasts of the Dominican Republic Haiti and Cuba . EDT the center of the hurricane was about 100 miles south of the Dominican Republic and 425 miles east of Kingston Jamaica . In Jamaica the government issued a hurricane watch for the entire island . Tropical Storm Gilbert formed in the eastern Caribbean and strengthened into a hurricane Saturday night . ', 'rouge_score_detail': {'rouge1': {'precision': 0.44129554655870445, 'recall': 0.5343137254901961, 'fmeasure': 0.48337028824833705}, 'rouge2': {'precision': 0.2073170731707317, 'recall': 0.2512315270935961, 'fmeasure': 0.22717149220489978}, 'rougeL': {'precision': 0.25101214574898784, 'recall': 0.30392156862745096, 'fmeasure': 0.2749445676274945}}}, 'topmost_redundancy_reduction': {'candidate': array([0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

The following full text is converted into the extractive summary given below.

Full Text:

Hurricane Gilbert swept toward the Dominican Republic Sunday, and the Civil Defense alerted its heavily populated south coast to prepare for high winds, heavy rains and high seas. The storm was approaching from the southeast with sustained winds of 75 mph gusting to 92 mph. "There is no need for alarm," Civil Defense Director Eugenio Cabral said in a television alert shortly before midnight Saturday. Cabral said residents of the province of Barahona should closely follow Gilbert's movement. An estimated 100,000 people live in the province, including 70,000 in the city of Barahona, about 125 miles west of Santo Domingo. Tropical Storm Gilbert formed in the eastern Caribbean and strengthened into a hurricane Saturday night. The National Hurricane Center in Miami reported its position at 2 a.m. Sunday at latitude 16.1 north, longitude 67.5 west, about 140 miles south of Ponce, Puerto Rico, and 200 miles southeast of Santo Domingo. The National Weather Service in San Juan, Puerto Rico, said Gilbert was moving westward at 15 mph with a "broad area of cloudiness and heavy weather" rotating around the center of the storm. The weather service issued a flash flood watch for Puerto Rico and the Virgin Islands until at least 6 p.m. Sunday. Strong winds associated with the Gilbert brought coastal flooding, strong southeast winds and up to 12 feet feet to Puerto Rico's south coast. There were no reports of casualties. San Juan, on the north coast, had heavy rains and gusts Saturday, but they subsided during the night. On Saturday, Hurricane Florence was downgraded to a tropical storm and its remnants pushed inland from the U.S. Gulf Coast. Residents returned home, happy to find little damage from 80 mph winds and sheets of rain. Florence, the sixth named storm of the 1988 Atlantic storm season, was the second hurricane. The first, Debby, reached minimal hurricane strength briefly before hitting the Mexican coast last month. Hurricane Gilbert, packing 110 mph winds and torrential rain, moved over this capital city today after skirting Puerto Rico, Haiti and the Dominican Republic. There were no immediate reports of casualties. Telephone communications were affected. "Right now it's actually moving over Jamaica," said Bob Sheets, director of the National Hurricane Center in Miami. "We've already had reports of 110 mph winds on the eastern tip. "It looks like the eye is going to move lengthwise across that island, and they're going to bear the full brunt of this powerful hurricane," Sheets said. Forecasters say Gilbert was expected to lash Jamaica throughout the day and was on track to later strike the Cayman Islands, a small British dependency northwest of Jamaica. Meanwhile, Havana Radio reported today that 25,000 people were evacuated from Guantanamo Province on Cuba's southeastern coast as strong winds fanning out from Gilbert began brushing the island. All Jamaica-bound flights were canceled at Miami International Airport, while flights from Grand Cayman, the main island of the three-island chain, arrived packed with frightened travelers. "People were running around in the main lobby of our hotel (on Grand Cayman) like chickens with their heads cut off," said one vacationer who was returning home to California through Miami. Hurricane warnings were posted for the Cayman Islands, Cuba and Haiti. Warnings were discontinued for the Dominican Republic. "All interests in the Western Caribbean should continue to monitor the progress of this dangerous hurricane," the service said, adding, "Little

change in strength is expected for the next several hours as the hurricane moves westward over Jamaica." The Associated Press' Caribbean headquarters in San Juan, Puerto Rico, was unable to get phone calls through to Kingston, where high winds and heavy rain preceding the storm drenched the capital overnight, toppling trees, causing local flooding and littering streets with branches. Most Jamaicans stayed home, boarding up windows in preparation for the hurricane. Some companies broadcast appeals for technicians and electricians to report to work. The weather bureau predicted Gilbert's center, 140 miles southeast of Kingston before dawn, would pass south of Kingston and hit the southern parish of Clarendon. Flash flood warnings were issued for the parishes of Portland on the northeast and St. Mary on the north. The north coast tourist region from Montego Bay on the west and Ocho Rios on the east, far from the southern impact zone and separated by mountains, was expected only to receive heavy rain. Officials urged residents in the higher risk areas along the south coast to seek higher ground. "It's certainly one of the larger systems we've seen in the Caribbean for a long time," said Hal Gerrish, forecaster at the National Hurricane Center. Forecasters at the center said the eye of Gilbert was 140 miles southeast of Kingston at dawn today. Maximum sustained winds were near 110 mph, with tropical-storm force winds extending up to 250 miles to the north and 100 miles to the south. Prime Minister Edward Seaga of Jamaica alerted all government agencies, saying Sunday night: "Hurricane Gilbert appears to be a real threat and everyone should follow the instructions and hurricane precautions issued by the Office of Disaster Preparedness in order to minimize the danger." Forecasters said the hurricane had been gaining strength as it passed over the ocean after it dumped 5 to 10 inches of rain on the Dominican Republic and Haiti, which share the island of Hispaniola. "We should know within about 72 hours whether it's going to be a major threat to the United States," said Martin Nelson, another meteorologist at the center. "It's moving at about 17 mph to the west and normally hurricanes take a northward turn after they pass central Cuba." Cuba's official Prensa Latina news agency said a state of alert was declared at midday in the Cuban provinces of Guantanamo, Holguin, Santiago de Cuba and Granma. In the report from Havana received in Mexico City, Prensa Latina said civil defense officials were broadcasting bulletins on national radio and television recommending emergency measures and providing information on the storm. Heavy rain and stiff winds downed power lines and caused flooding in the Dominican Republic on Sunday night as the hurricane's center passed just south of the Barahona peninsula, then less than 100 miles from neighboring Haiti. The storm ripped the roofs off houses and flooded coastal areas of southwestern Puerto Rico after reaching hurricane strength off the island's southeast Saturday night. Flights were canceled Sunday in the Dominican Republic, where civil defense director Eugenio Cabral reported some flooding in parts of the capital of Santo Domingo and power outages there and in other southern areas. Hurricane Gilbert slammed into… Kingston on Monday with torrential rains and 115 mph winds that ripped roofs off homes and buildings, uprooted trees and downed power lines. No serious injuries were immediately reported in the city of 750,000 people, which was hit by the full force of the hurricane around noon. For half an hour, the hurricane lashed the city, tearing branches from trees, blowing down fences and whipping paper through the air. The National Weather Service reported heavy damage to Kingston's airport and aircraft parked on its fields. The first shock let up as the eye of the storm moved across the city. Skies brightened, the winds died down and people waited for an hour before the second blow of the hurricane arrived. All Jamaica-bound flights were canceled at Miami International Airport. Flights from the Cayman Islands, reportedly next in the path of the hurricane, arrived in Miami packed with travelers cutting short their……

Extractive Summary:

Tropical Storm Gilbert formed in the eastern Caribbean and strengthened into a hurricane Saturday night. Hurricane Gilbert packing 110 mph winds and torrential rain moved over this capital city today after skirting Puerto Rico Haiti and the Dominican Republic. Right now, its actually moving over Jamaica said Bob Sheets director of the National Hurricane Center in Miami. “It’s certainly one of the larger systems we’ve seen in the Caribbean for a long time,” said Hal Gerrish forecaster at the National Hurricane Center. Heavy rain and stiff winds downed power lines and caused flooding in the Dominican Republic on Sunday night as the hurricanes center passed just south of the Barahona peninsula then less than 100 miles from neighboring Haiti. The storm ripped the roofs off houses and flooded coastal areas of southwestern Puerto Rico after reaching hurricane strength off the islands southeast Saturday night. A National Weather Service report said the hurricane was moving west at 17 mph with maximum sustained winds of 115 mph. Gilbert reached Jamaica after skirting southern Puerto Rico Haiti and the Dominican Republic. Havana Radio meanwhile reported Monday that 25 000 people were evacuated from coastal areas in Guantanamo Province on the nation’s southeastern coast as Gilberts winds and rain began to brush the island. Hurricane Gilbert swept toward Jamaica yesterday with 100mileanhour winds and officials issued warnings to residents on the southern coasts of the Dominican Republic Haiti and Cuba.

CHAPTER - 8

REFERENCES

[1] Sanchez-Gomez, J. M., Vega-Rodríguez, M. A., & Pérez, C. J. (2018). Extractive multi-document text summarization using a multi-objective artificial bee colony optimization approach. *Knowledge-Based Systems*, *159*, 1-8.

[2] Sanchez-Gomez, J. M., Vega-Rodríguez, M. A., & Pérez, C. J. (2020). A decomposition-based multi-objective optimization approach for extractive multi-document text summarization. *Applied Soft Computing*, *91*, 106231.

[3] Sanchez-Gomez, J. M., Vega-Rodríguez, M. A., & Pérez, C. J. (2020). Experimental analysis of multiple criteria for extractive multi-document text summarization. *Expert Systems with Applications*, *140*, 112904.

[4] Jung, C., Datta, R., & Segev, A. (2017, July). Multi-document summarization using evolutionary multi-objective optimization. In *Proceedings of the genetic and evolutionary computation conference companion* (pp. 31-32).

[5] Sanchez-Gomez, J. M., Vega-Rodríguez, M. A., & Pérez, C. J. (2022). A multi-objective memetic algorithm for query-oriented text summarization: Medicine texts as a case study. *Expert Systems with Applications*, *198*, 116769.

[6] Saini, N., Saha, S., Chakraborty, D., & Bhattacharyya, P. (2019). Extractive single document summarization using binary differential evolution: Optimization of different sentence quality measures. *PloS one*, *14*(11), e0223477.

[7] Moratanch, N., & Chitrakala, S. (2017, January). A survey on extractive text summarization. In *2017 international conference on computer, communication and signal processing (ICCCSP)* (pp. 1-6). IEEE.

[8] Widyassari, A. P., Rustad, S., Shidik, G. F., Noersasongko, E., Syukur, A., & Affandy, A. (2022). Review of automatic text summarization techniques & methods. *Journal of King Saud University-Computer and Information Sciences*, *34*(4), 1029-1046.

[9] Yadav, A. K., Ranvijay, Yadav, R. S., & Maurya, A. K. (2023). State-of-the-art approach to extractive text summarization: a comprehensive review. *Multimedia Tools and Applications*, *82*(19), 29135-29197.

[10] Belwal, R. C., Rai, S., & Gupta, A. (2021). A new graph-based extractive text summarization using keywords or topic modeling. *Journal of Ambient Intelligence and Humanized Computing*, *12*(10), 8975-8990.

[11] Yadav, A. K., Ranvijay, Yadav, R. S., & Maurya, A. K. (2024). Graph-based extractive text summarization based on single document. *Multimedia Tools and Applications*, *83*(7), 18987-19013.

[12] Erkan, G., & Radev, D. R. (2004). Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of artificial intelligence research*, *22*, 457-479.

[13] Mihalcea, R., & Tarau, P. (2004, July). Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing* (pp. 404-411).

[14] Wahab, M. H. H., Hamid, N. A. W. A., Subramaniam, S., Latip, R., & Othman, M. (2024). Decomposition–based multi-objective differential evolution for extractive multi-document automatic text summarization. Applied Soft Computing, 151, 110994.

[15] Gialitsis, N., Pittaras, N., & Stamatopoulos, P. (2019, September). A topic-based sentence representation for extractive text summarization. In *Proceedings of the Workshop MultiLing 2019: Summarization Across Languages, Genres and Sources* (pp. 26-34).

[16] Wu, Z., Lei, L., Li, G., Huang, H., Zheng, C., Chen, E., & Xu, G. (2017). A topic modeling based approach to novel document automatic summarization. *Expert Systems with Applications*, *84*, 12-23.

[17] Belwal, R. C., Rai, S., & Gupta, A. (2021). Text summarization using topic-based vector space model and semantic measure. *Information Processing & Management*, *58*(3), 102536.

[18] Jung, C., Yoon, W. C., Datta, R., & Jung, S. (2021). Knowledge base driven automatic text summarization using multi-objective optimization. *International journal of advanced computer science and applications*, *12*(8).

[19] Debnath, D., Das, R., & Pakray, P. (2021). Extractive single document summarization using multi-objective modified cat swarm optimization approach: ESDS-MCSO. *Neural Computing and Applications*, 1-16.

[20] Mishra, S. K., Saini, N., Saha, S., & Bhattacharyya, P. (2022). Scientific document summarization in multi-objective clustering framework. *Applied Intelligence*, *52*(2), 1520-1543.

[21] Sanchez-Gomez, J. M., Vega-Rodríguez, M. A., & Pérez, C. J. (2019). Comparison of automatic methods for reducing the Pareto front to a single solution applied to multi-document text summarization. *Knowledge-Based Systems*, *174*, 123-136.

[22] Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. A. M. T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE transactions on evolutionary computation, 6(2), 182-197

CHAPTER 9

APPENDIX

9.1 BASE PAPER

Engineering Applications of Artificial Intelligence 119 (2023) 105757

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

A new multi-objective evolutionary algorithm for citation-based summarization: Comprehensive analysis of the generated summaries

Jesus M. Sanchez-Gomez [a,*], Miguel A. Vega-Rodríguez [a], Carlos J. Pérez [b]

[a] Universidad de Extremadura, Departamento de Tecnología de Computadores y Comunicaciones, Campus Universitario s/n, 10003 Cáceres, Spain[1]
[b] Universidad de Extremadura, Departamento de Matemáticas, Campus Universitario s/n, 10003 Cáceres, Spain[1]

ARTICLE INFO

Keywords:
Evolutionary algorithm
Multi-objective optimization
MOEA/D
Scientific summarization
Citation-based summarization

ABSTRACT

The number of scientific publications in different knowledge fields has considerably grown in recent times. This makes difficult for researchers to synthesize all the scientific-technical advances, so automatic summarization methods of scientific papers would be helpful. These methods generate a summary from a reference paper with its most relevant contributions. More specifically, citation-based summarization considers the citation contexts to the reference paper in subsequent publications. For the first time, this problem has been formulated as a multi-objective optimization problem, optimizing the content coverage and the redundancy reduction in a simultaneous way. A Decomposition-based Multi-Objective optimization algorithm for Citation-based Summarization (DMOCS) has been designed, developed, and applied for solving this problem. The results obtained by the proposed approach have improved the existing ones in the scientific literature between 17.47% and 133.50%, increasing the ROUGE percentage improvements when the N-gram is larger. Besides, an exhaustive analysis of the different parts of a scientific paper has been performed, showing that the citations from the citing papers with their corresponding spans in the reference paper impact in the quality of a citation-based summary.

1. Introduction

Nowadays, the amount of publications in every area of the scientific literature grows considerably, since a lot of articles, books, and proceedings are constantly being published. This makes it difficult to carry out one of the researchers' main tasks, which is to collect and analyze a set of papers from a certain topic, which entails a great time consumption. The natural solution for this matter is to make the information contained in the papers more tractable through automatic text summarization techniques. Specifically, scientific summarization methods can be used to address this issue.

Scientific summarization aims to provide researchers a brief and informative representation of the most relevant aspects, impacts, or contributions of a paper. Another characteristic of scientific summarization methods is that the extension of the documents involved is generally greater than other kind of documents, in addition to the fact that scientific papers follows a template-like structure: introduction, hypotheses, methods, results, discussion, and conclusions (El-Kassas et al., 2021). A scientific paper summary is classified in two ways (Altmami and Menai, 2022). In the first place, the abstract itself contained in the paper. This summary provides a general overview of the publication. However, sometimes, it shows the authors' viewpoint in a possibly inadequate, incomplete, or biased way. In addition, the abstract cannot include all the contributions and impacts of the paper due to its restricted length. Even the contributions included in the abstract could be of low relevance to the scientific community (Iqbal et al., 2021). All these issues led to the development of other kind of scientific paper summary, the citation-based summary. A citation-based summary considers, in addition to the text of the reference paper, the subsequent citations referring to it (Iqbal et al., 2021). Besides, the contributions expressed in the citations are generally more focused than the ones in the abstract, and they include supplementary information missing in the abstract.

Therefore, citation-based summarization methods also use the citations to a reference paper. In the citing papers (the subsequent publications that cite the reference paper), the part of text that develops the citation is defined as the citation context (Jebari et al., 2021). Likewise, in the reference paper, the part of text that is referenced by the citation is referred as the citing span. In this way, the set of citation contexts may be seen as a small community-created summary of the reference paper (Wang et al., 2020). Therefore, the sentences for a citation-based summary can be extracted from the following texts: the abstract, the entire body, the citing spans from the reference paper, and the citation contexts from the set of citing papers.

* Corresponding author.
E-mail addresses: jmsanchezgomez@unex.es (J.M. Sanchez-Gomez), mavega@unex.es (M.A. Vega-Rodríguez), carper@unex.es (C.J. Pérez).
[1] https://ror.org/0174shg90

https://doi.org/10.1016/j.engappai.2022.105757
Received 12 July 2022; Received in revised form 2 December 2022; Accepted 19 December 2022
Available online 29 December 2022
0952-1976/© 2022 Elsevier Ltd. All rights reserved.

In general, a summary has a superior quality when several objectives are optimized, such as its content coverage and its redundancy reduction. For this reason, multi-objective optimization approaches could be used to address the citation-based summarization problem. Unlike single-objective optimization approaches, multi-objective ones can optimize several objective functions at the same time. In the multi-objective optimization area, a decomposition-based optimization strategy is within the most successfully used, which divides the multi-objective optimization problem into a determined number of scalar subproblems (Zhang and Li, 2007).

Therefore, two research gaps were found in this field. On the one hand, none of the existing proposals addresses the citation-based summarization problem from a multi-objective optimization perspective, despite of its advantages compared to single-objective one. On the other hand, there is a lack of an exhaustive analysis of the candidate texts for citation-based summarization (abstract, body, citing spans, and citation contexts) in order to find out which one has more influence on the quality of a citation-based summary.

In this paper, a Decomposition-based Multi-Objective optimization algorithm for Citation-based Summarization (DMOCS) has been designed, developed, and applied for solving the citation-based summarization problem. Content coverage and redundancy reduction have been considered as the objective functions to be simultaneously optimized. The different parts that may compose a citation-based summary have been comprehensively analyzed: abstract, body, citing spans, and citation contexts. For the experimentation, the datasets provided by the Text Analysis Conference (TAC) have been used, specifically the dataset from the TAC 2014 Biomedical Summarization Track (2017). Regarding the evaluation, the Recall-Oriented Understudy for Gisting Evaluation (ROUGE) metrics have been considered (Lin, 2004). The main contributions of this work are the following ones:

- For the first time, the citation-based summarization problem has been solved by using a multi-objective optimization approach.
- A Decomposition-based Multi-Objective optimization algorithm for Citation-based Summarization (DMOCS) has been specifically designed, developed, and applied for this problem.
- An extensive analysis of the candidate texts (abstract, body, citing spans, and citation contexts) for citation-based summarization has been performed.
- Experimentation has shown that the citing spans are more important than the paper body and that the citation contexts are especially useful to provide high-quality citation-based summaries.
- The results of the DMOCS approach have outperformed the ones from the approaches in the scientific literature in terms of ROUGE scores.

The rest of the paper is organized as follows. Section 2 presents the related works. In Section 3, the citation-based summarization problem is defined. Section 4 describes the proposed decomposition-based multi-objective optimization approach. In Section 5, the datasets used and the evaluation metrics considered are detailed. Section 6 includes the experimental settings, the results of the extensive analysis, and the comparison with other approaches. Finally, Section 7 contains the conclusions.

2. Related work

The scientific literature contains a large amount of works for automatic text summarization. Some reviews have been recently published on this topic, such as Kumar et al. (2021) and Widyassari et al. (2022). More specifically, some surveys have addressed the citation-based summarization problem (Iqbal et al., 2021; El-Kassas et al., 2021). In this section, only comparable citation-based summarization approaches are reviewed.

Mollá et al. (2014) showed that the information from the citing papers aids to perform an extractive summary of a scientific publication. They reported several methods for finding the sentences in the reference paper that best match the citation context: Oracle, $tf-idf$ with singular value decomposition, maximal marginal relevance, and unified medical language system with WordNet. By means of these methods, they analyzed the use of the citation contexts for selecting the most relevant sentences from the reference paper. In this way, the generated summaries were comparable with the ones produced by standard extractive summarization methods since the first ones improve them in terms of readability, balance, and coherence.

Cohan and Goharian (2015) proposed a summarization approach for scientific papers that leveraged the citation contexts and the document discourse model. This approach addressed the problem of inconsistency between the content of the paper and the citation-based summary by providing the appropriate context for every citation. Firstly, the citing spans of the reference paper corresponding to every citation were extracted. Secondly, the candidate sentences for the final summary were pulled out by means of the previous citing spans and the discourse facets of the citations. Finally, the summary was generated by maximizing the informativeness and the novelty of the candidate sentences.

Conroy and Davis (2015) analyzed different approaches for approximating the latent weights of terms for the scientific summarization problem. The first one was a term-frequency vector space model that used a non-negative matrix factorization for dimensionality reduction. The other two approaches were based on language modeling to predict the distributions of the terms in the human-generated summaries. This language model takes advantage of the main sections of the reference paper and the set of citation contexts. The contemplated system was made of the following steps: data preprocessing and segmentation, term selection, latent term weight estimation, and sentence selection.

In a later work, Conroy and Davis (2018) presented a summarization system for scientific and structured documents, consisting of three elements: a section mixture model that was used for the estimation of the weights of terms, a hypothesis test for choosing a subset of these terms, and a combinatorial optimization method for the extraction of the sentences. The section mixture model was a transformation of the bigram mixture model based on the principal sections of a scientific paper and a set of citation contexts. Specifically, this language model was a modification of the one defined in their previous work (Conroy and Davis, 2015). Regarding the extraction of the sentences, an optimal combinatorial covering algorithm that maximized the content coverage score and minimized the redundancy was used.

Finally, Ronzano and Saggion (2016) studied how to exploit the citations of a paper with the aim of identifying its most relevant parts. For this task, they evaluated the differences in the performance of an extractive summarization approach when the different parts of a paper are considered or not for summarization. In this way, the contents of the paper (separated in abstract and body), the citation contexts, and their corresponding citing spans were analyzed in order to assess their contribution to the quality of the summary in terms of ROUGE scores.

All the reviewed works considered the dataset from the TAC2014 Biomedical Summarization Track in their experimentation. Moreover, all of them evaluated their experimentation with ROUGE metrics.

3. Problem definition

This section presents the definition of the citation-based summarization problem. The problems based on extractive summarization technique have been commonly addressed by using vector-based word approaches, where each sentence is defined as a vector of words. In addition, the presented problem has been formulated as a multi-objective optimization problem.

The problem definition is the following. Firstly, let the document collection $D = \{d_1, d_2, \ldots, d_N\}$ be a set with N documents. For simplicity purpose, D is also symbolized as a set containing the n sentences in the document collection, i.e., $D = \{s_1, s_2, \ldots, s_n\}$. Secondly, let $T = \{t_1, t_2, \ldots, t_m\}$ be a set of terms that contains the m different terms

in D. Now, every sentence $s_i \in D$ is represented as a vector of m dimensions as $s_i = (w_{i1}, w_{i2}, \dots, w_{im}), i = 1, 2, \dots, n$, so every component is characterized by the weight of the term t_k in the sentence s_i. The value of the weight w_{ik} is determined by using the *tf-isf* scheme (Salton and Buckley, 1988). This scheme integrates the definitions of the term frequency and inverse sentence frequency to generate a weight for each term in each sentence:

$$w_{ik} = tf_{ik} \cdot \log \frac{n}{n_k}, \tag{1}$$

where the first multiplier, the term frequency, counts the number of times that the kth term appears in the ith sentence (tf_{ik}), and the second multiplier, the inverse sentence frequency, counts the number of sentences in which the kth term occurs (n_k) regarding the total of sentences (n).

Besides, the main content or center of the document collection D can be represented with the mean vector $O = (o_1, o_2, \dots, o_m)$ (Radev et al., 2004). Each component o_k is computed as follows:

$$o_k = \frac{1}{n} \sum_{i=1}^{n} w_{ik}, \quad k = 1, 2, \dots, m. \tag{2}$$

Finally, the cosine similarity measures the likeness between two sentences s_i and s_j, and it is calculated as:

$$sim(s_i, s_j) = \frac{\sum_{k=1}^{m} (w_{ik} \cdot w_{jk})}{\sqrt{\sum_{k=1}^{m} w_{ik}^2 \cdot \sum_{k=1}^{m} w_{jk}^2}}, \quad i, j = 1, 2, \dots, n. \tag{3}$$

Now, after defining the notation and main concepts, the citation-based summarization problem is described. The aim is to produce a summary, $S \subset D$, considering the following aspects:

- Content coverage: the main content of the document collection should be covered by the generated summary.
- Redundancy reduction: the sentences included in the generated summary should not be similar among them to avoid redundancy.
- Length constraint: the generated summary should have a determined length L.

The citation-based summarization problem entails the simultaneous optimization of the two criteria (content coverage and redundancy reduction) formulated as two objective functions, also considering the length constraint. Moreover, multi-objective optimization is a good strategy to address this problem, since the objective functions are in conflict with each other. For example, the more sentences from the document collection are included in the summary, the more its content coverage increases, but the redundancy reduction decreases at the same time. Conversely, the fewer sentences the summary contains, the more its redundancy reduction will be, but the content coverage will decrease. Definitely, multi-objective optimization is able to deal with this issue.

Before determining the objective functions, the representation of a solution is symbolized as a binary vector, i.e. $X = (x_1, x_2, \dots, x_n)$, where each component x_i is a binary variable that contemplates the presence ($x_i = 1$) or absence ($x_i = 0$) of the sentence s_i in the summary S.

The first objective function corresponds to the content coverage criterion as $\Phi_{CC}(X)$. It is formulated as the sum of the cosine similarities between every sentence in the summary, $s_i \in S$, and the center of the document collection D, represented with the mean vector O. Therefore, the following function should be maximized:

$$\Phi_{CC}(X) = \sum_{i=1}^{n} sim(s_i, O) \cdot x_i. \tag{4}$$

The second objective function refers to the redundancy reduction criterion, $\Phi_{RR}(X)$. The redundancy is defined as the sum of the cosine similarities between each pair of sentences in the summary, $s_i, s_j \in S$. As the redundancy reduction is the opposite concept to redundancy, then the following function should be maximized:

$$\Phi_{RR}(X) = -\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} sim(s_i, s_j) \cdot x_i x_j. \tag{5}$$

Finally, as the proposed approach is based on the Pareto optimality, the multi-objective citation-based summarization problem can be formulated in the following way:

$$\max \ \phi(X) = \left\{ \Phi_{CC}(X), \Phi_{RR}(X) \right\}, \tag{6}$$

$$\text{subject to } L - \varepsilon \leq \sum_{i=1}^{n} l_i \cdot x_i \leq L + \varepsilon, \tag{7}$$

where l_i is the length of the sentence s_i and ε the length tolerance, which is calculated as the difference between the longest and the shortest sentence in the document collection.

4. The proposed multi-objective optimization approach

Firstly, the features of the decomposition-based optimization approach and its pseudocode are presented. Then, in the following subsections, the main operators of the proposed approach are described. And finally, the procedure for selecting the final solution is reported.

In the problem presented here, the objective functions to be optimized (maximized in this case) are conflicting between them. This means that there is no single point in the objective space that maximizes all the objective functions in a simultaneous way, but rather a set of them with different trade-offs, named as Pareto optimal points. These points compose the Pareto front. A more detailed explanation about multi-objective optimization aspects can be found in Deb (2015).

In this work, a Decomposition-based Multi-Objective optimization for Citation-based Summarization (DMOCS) approach has been specifically designed and developed for addressing this problem. The DMOCS approach is based on the Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D) (Zhang and Li, 2007). MOEA/D decomposes the multi-objective problem into scalar subproblems, and then it optimizes them, also considering the information from their neighboring subproblems. Moreover, MOEA/D has been widely and successfully used in many research fields. Some of the applications in real-world problems are: reservoir flood control problem (Ma et al., 2016), job shop scheduling problem (Zhao et al., 2017), structural optimization problems (Ho-Huu et al., 2018), wastewater treatment process (Zhou and Qiao, 2019), or crowdsensing problem with social awareness (Ji et al., 2020), among others.

In DMOCS the following concepts and notation are used. Let $\lambda^i = (\lambda_1^i, \lambda_2^i)$ be a vector of weights with two elements, one for each objective j, with $\sum_{j=1}^{2} \lambda_j^i = 1$. Additionally, let $\Lambda = \{\lambda^1, \lambda^2, \dots, \lambda^{pop_s}\}$ be a set with pop_s vectors of weights, where pop_s is the population size. In this way, the multi-objective problem is split into pop_s scalar optimization subproblems, and each individual i in the population is concerned with its corresponding weight vector λ^i. Now, for each vector of weights λ^i, its neighborhood is determined as a set of its nearest vectors of weights as $\{\lambda^{i,1}, \lambda^{i,2}, \dots, \lambda^{i,niche}\}$, being *niche* the neighborhood size. Finally, let $z^* = (z_1^*, z_2^*)$ be the reference point, where z_j^* is the best value found for the objective j. Therefore, the objective space is separated into pop_s evenly-distributed subspaces, and in each of them the solution is optimized as a scalar subproblem. Moreover, every subproblem i considers its corresponding weight vector λ^i and the information from its neighboring subproblems $\{\lambda^{i,1}, \lambda^{i,2}, \dots, \lambda^{i,niche}\}$.

Regarding the scalarizing functions, there are different approaches that can be used, such as Tchebycheff, normalized Tchebycheff, and boundary intersection. The Tchebycheff approach considers the longest distance, from all the objectives, between the solutions and the reference point (Jaszkiewicz, 2002). The normalized Tchebycheff approach operates in a similar way as the previous one, except that it normalizes the different objectives, avoiding biases (Tang et al., 2015). Finally,

the boundary intersection approach takes into account the distance between the solutions and the boundary of the attainable objective set, being also independent of the scales of the objectives (Das and Dennis, 1998).

Algorithm 1 displays the pseudocode of the DMOCS approach. Firstly, the initialization steps are performed from lines 1 to 5. In line 1, the final set of solutions FSS is initialized to an empty set. The set with the pop_s evenly-distributed vectors of weights is initialized in line 2. In line 3, the neighborhoods of the subproblems are initialized, based on the distances of the vectors of weights. Then, the initial population is generated in a random way in line 4. And lastly, in line 5, the reference point is initialized according to the initial population by selecting the best value found in all the individuals for each objective.

Algorithm 1 Pseudocode for the DMOCS approach

Input: pop_s: population size, gen_m: generations, $niche_s$: neighborhood size
Output: FSS: Final set of solutions

1: $FSS \leftarrow \emptyset$
2: $Weights \leftarrow initialize\_weights(pop_s)$
3: $Neighborhoods \leftarrow initialize\_neighborhoods(Weights, pop_s, niche_s)$
4: $Population \leftarrow initialize\_population(pop_s)$
5: $Reference \leftarrow initialize\_reference(Population)$
6: **for** $gen = 1$ **to** gen_m **do**
7: **for** $i = 1$ **to** pop_s **do**
8: $Parent1, Parent2 \leftarrow get\_randomly(Population, Neighborhoods, i)$
9: $Offspring \leftarrow crossover\_operation(Parent1, Parent2)$
10: $Offspring \leftarrow mutation\_operation(Offspring)$
11: $Reference \leftarrow update\_reference(Reference, Offspring)$
12: $Population \leftarrow update\_problem(Population, Neighborhoods, Weights, Reference, Offspring, i)$
13: **end for**
14: $FSS \leftarrow FSS \cup save\_population(Population)$
15: **end for**
16: **return** FSS

The steps from lines 6 to 15 are repeated during a maximum number of generations gen_m. Within this first loop, the steps from lines 7 to 13 are carried out for every individual i of the population. In this second loop, first, the selection operation is performed in line 8. This operation consists of selecting two parents in a random way, $Parent1$ and $Parent2$, from the corresponding neighborhood of the individual i. Then, in line 9, the crossover operation is carried out with the chosen parents. This operation is detailed in Section 4.1, and an offspring solution from the two parents is obtained. In line 10, the mutation operation is performed over the offspring solution. With this operation, that is detailed in Section 4.2, an improvement heuristic is carried out on the offspring solution. Later, the reference point is updated with the offspring solution in line 11. And at the end of this second loop, in line 12, the best solutions of the neighboring subproblems are updated with the offspring solution. In addition, the scalarizing function used here depends on the considered decomposition approach. After the execution of a generation, the entire population is saved in the final set of solutions. In this step, the repair operation (described in Section 4.3) is performed over every individual of the population before being stored. Furthermore, it is checked that only non-dominated and non-repeated solutions are saved in the final set.

Finally, after the execution of the two loops, the final set of solutions is returned in line 16. Fig. 1 displays the flowchart of the proposed DMOCS approach.

4.1. Crossover operator

The crossover operator has been specifically designed for the DMOCS approach. This operation produces an offspring solution with the best sentences from the two parents. To this end, a set of candidate sentences is made through joining all their sentences, and a crossover score is calculated for each candidate sentence. The crossover score considers the cosine similarity between the ith sentence s_i and the mean vector O, so it is computed as:

$$score^{cro}_{s_i} = \frac{sim(s_i, O)}{\frac{1}{n}\sum_{j=1}^{n} sim(s_j, O)}. \tag{8}$$

Then, the set of candidate sentences is ordered from the largest to the lowest score. Finally, the offspring solution is generated by incorporating the first sentences till attaining the maximum length constraint established in Eq. (7).

4.2. Mutation operator

The mutation operator developed here consists of the addition, removal, or exchange of a unique sentence from the summary with the aim of improving its quality. In every mutation, only one of these three options will be carried out, being randomly selected, as long as the condition of the length constraint in Eq. (7) is met. Thus, the mutation probability established is $p_m = 1/n$ since only one sentence is always mutated. The feasible options are described next:

- Addition of a sentence. A sentence not included in the summary, $s_i \notin S$, is randomly selected to be added to the summary. The condition to be added is that it must improve the quality of the summary regarding the average cosine similarity with the mean vector O:

$$sim(s_i, O) > \frac{1}{n}\sum_{j=1}^{n} sim(s_j, O). \tag{9}$$

  If the sentence fulfills this condition, then it is added to the summary. Otherwise, this condition is checked for the rest of sentences not included in the summary, until one of them does it. Then, this sentence will be added to the summary. However, if no sentence fulfills this condition, the sentence with the largest value of cosine similarity with the mean vector will be included.
- Removal of a sentence. A sentence that is contained in the summary, $s_i \in S$, is randomly chosen to be removed from the summary. In this case, the condition to be removed is that the chosen sentence must not degenerate the quality of the summary in terms of average cosine similarity with the mean vector O:

$$sim(s_i, O) < \frac{1}{n}\sum_{j=1}^{n} sim(s_j, O). \tag{10}$$

  If the sentence accomplishes the condition, then it is removed from the summary. On the contrary, this condition is examined for the remaining sentences in the summary until one of them meets the condition, and this sentence will be removed. Finally, if no sentence accomplishes this condition, then the one with the lowest value of cosine similarity with the mean vector will be removed from the summary.
- Exchange a sentence. This option consists of exchanging a sentence included in the summary with another of the document collection. This operation involves the previous ones: firstly, the removal of a sentence, and secondly, the addition of a different one.

4.3. Repair operator

The repair operator specifically designed for this problem lies in repairing the summaries generated by worsening as little as possible their quality regarding cosine similarity of their sentences with the mean vector O. Only the summaries that do not fulfill the constraint in Eq. (7) have to be repaired. After checking the length of the summary, if it is larger than the maximum allowed length constraint, the summary is

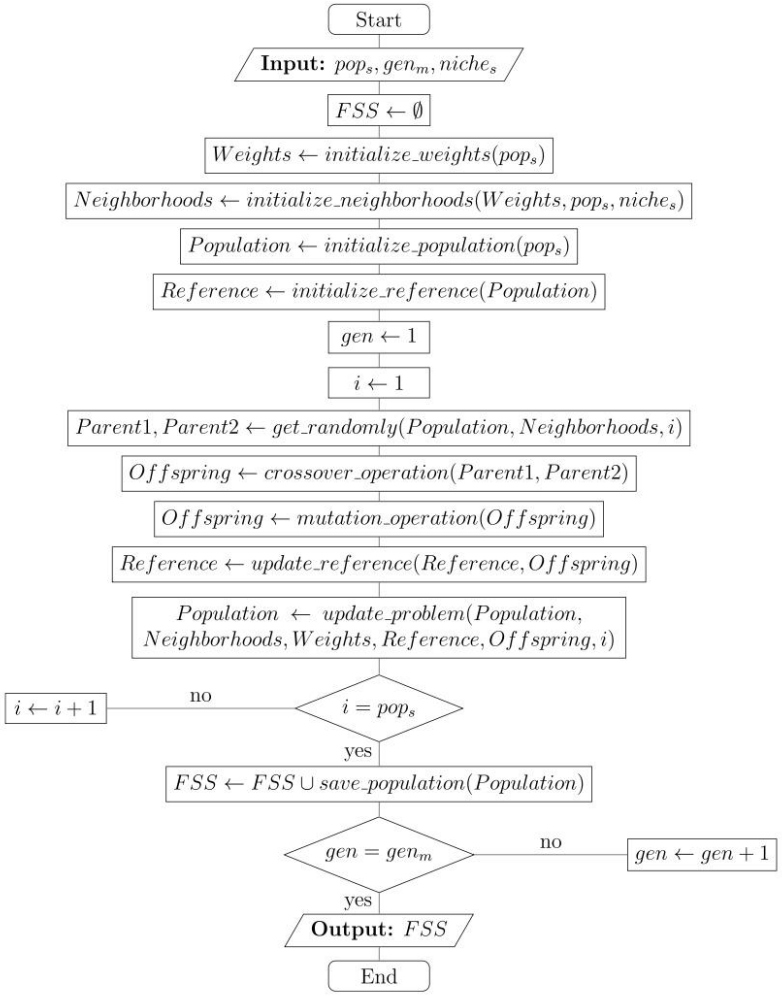


Fig. 1. Flowchart of the DMOCS approach.

repaired; and if it is lower than the minimum one, then the summary is discarded (this last situation is very infrequent). Hence, the reparation of a summary consists of removing the necessary sentences until the maximum allowed length constraint is accomplished. For each sentence in the summary, a repair score is calculated:

$$score_{s_i}^{rep} = \frac{sim(s_i, O)}{\frac{1}{n}\sum_{j=1}^{n} sim(s_j, O)}. \tag{11}$$

In this way, the sentence with the lowest repair score is removed from the summary. This procedure is repeated until the condition in Eq. (7) is fulfilled.

4.4. Selecting the final solution

All the solutions from the Pareto set may be selected as the single final solution. In order to avoid subjectivity, an automatic process is applied to select only one solution from the final Pareto set. Eleven methods have been applied and tested for the DMOCS approach: the hypervolume; the consensus solution; the shortest distance to the ideal point, in which the Euclidean, Manhattan, Chebyshev, and Mahalanobis distances have been studied; and the shortest distance to all points, in which the four previous distances in addition to the Levenshtein distance have been considered. The detailed explanation of all these methods can be found in Sanchez-Gomez et al. (2019).

Table 1
Characteristics of the used collections from the TAC2014 Biomedical Summarization Track. RP refers to Reference Paper, and CP to Citing Paper.

Description	Value
Number of collections	15
Average number of documents	11 (1 RP and 10 CPs)
Average number of words in RP	9,420.00
Average number of sentences in RP	557.33
Average number of citations per RP	16.13
Summary length constraint (words)	250

5. Datasets and evaluation metrics

The datasets used, the preprocessing of the document collections, and the evaluation metrics considered are described here.

5.1. Datasets

The datasets used for the experimentation have been provided by the Text Analysis Conference (TAC). TAC is a series of workshops organized by the National Institute of Standards and Technology (NIST, USA) to support research in natural language processing, and it provides datasets for some applications named as tracks. In specific, the dataset from the TAC 2014 Biomedical Summarization Track (2017)[2] has been considered.

This dataset contains twenty collections of biomedical documents. In every collection, there is one reference paper, ten citing papers, and a set of annotations made by four experts that contain the citing spans and the citation contexts. Specifically, these annotations consist of a set of pairs composed by a citing span and a citation context. That is, every citation context in a citing paper corresponds to a citing span in the reference paper.

The goal of this track is the generation of a summary of the reference paper that takes into account the community discussion represented by the citation contexts. Fig. 2 shows the scheme of a collection from this dataset.

In this work, fifteen collections have been used since the reference papers contained in five collections are not separated into sections, so they cannot be considered. Table 1 shows some characteristics of the collections used in the experimentation.

In addition to the characteristics shown in Table 1, each collection also includes the citation-based summaries made by the experts from TAC. These summaries (human-generated summaries) will be used for the quality assessment of the summaries generated by the proposed approach (system-generated summaries).

5.2. Preprocessing

This subsection describes the preprocessing steps needed for the document collections. Before the experimentation, the following steps have to be carried out over each collection:

- Clean-up of the reference paper. In the first place, the text of the reference paper is processed in order to separate its main parts: abstract and body. Therefore, all the other texts are removed from the paper, such as the title, authors, and the common sections of accession numbers, supplementary material, supporting information, acknowledgments, author contributions, and references. In addition, the body is also divided into the sections to be studied: introduction, results, and discussion. After this step, the texts of the abstract, body, citing spans, and citation contexts are gathered to be uniformly processed.
- Segmentation. The sentences are pulled out by demarcating their beginning and ending.
- Tokenization. The words are separated with a token, such as a blank space. In this step, punctuations, exclamations, interrogations, and other marks are erased.
- Stop words removal. The words with no meaning, such as articles, prepositions, or conjunctions, are removed from the sentences. The list of stop words from ROUGE package has been used (Li, 2020).
- Stemming. For the remaining words, their root is extracted by using the Porter stemming algorithm (Porter, 2021). In this way, the words that have the same lexical root will be taken into account as an only term.

[2] https://tac.nist.gov/2014/BiomedSumm/data.html

Table 2
Counts of terms and sentences after preprocessing the used collections from the TAC2014 Biomedical Summarization Track.

Part	Section	Average number of terms	Average number of sentences
Abstract		153.87	7.60
Body	Introduction	659.60	23.73
	Results	2,954.67	107.13
	Discussion	1,197.80	43.73
Total		4,965.94	182.19
Citing spans	Introduction	237.20	8.27
	Results	1,046.73	37.53
	Discussion	502.13	17.87
Total		1,786.06	63.67
Citation contexts	Introduction	228.73	7.33
	Results	208.67	7.07
	Discussion	193.13	6.33
Total		630.53	20.73

Table 2 reports the counts of terms and sentences of the considered collections after the preprocessing steps.

5.3. Evaluation metrics

To assess the performance of the generated summaries, the Recall-Oriented Understudy for Gisting Evaluation (ROUGE) metrics have been considered (Lin, 2004). These metrics are the most widely used for the evaluation of automatic summarization. They measure the similarity between a system-generated summary and a human-generated summary through computing the number of overlapping units.

ROUGE-N and ROUGE-L scores have been used. On the one hand, ROUGE-N scores measure the N-gram recall between a system-generated summary and a set of human-generated summaries. Four ROUGE-N scores have been considered: ROUGE-1, ROUGE-2, ROUGE-3, and ROUGE-4. On the other hand, ROUGE-L score computes a ratio based on the length of the summaries' longest common subsequence.

6. Experimental results

This section presents the experimental settings, the results of the comprehensive analysis, and the comparison with other approaches.

6.1. Experimental settings

The parameters of the DMOCS approach are: the population size, pop_s; the maximum number of generations, gen_m; the mutation probability, p_m; the neighborhood size or niche, $niche_s$; and the decomposition approach (or scalarizing function), $approach_d$. The values of the population size, the maximum number of generations, and the mutation probability were fixed to $pop_s = 64$, $gen_m = 500$, and $p_m = 1/n$ (as reported in Section 4.2). Furthermore, a parametric study was conducted for the specific parameters of the algorithm: the neighborhood size or

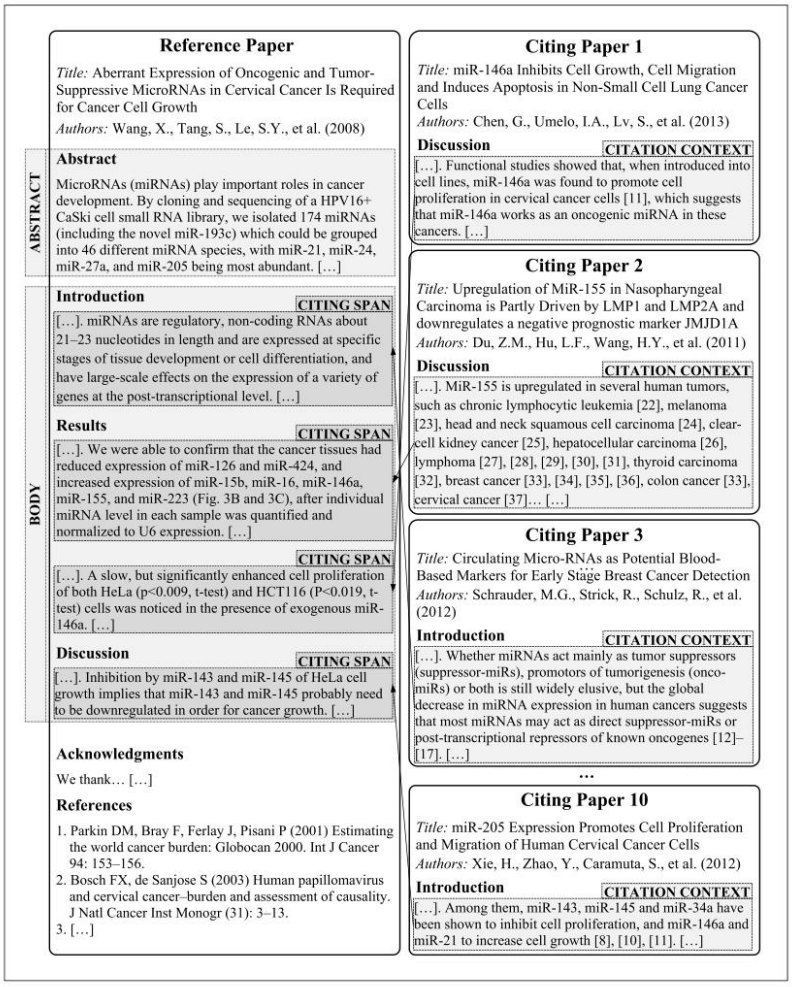


Fig. 2. Scheme of a collection from the TAC2014 Biomedical Summarization Track.

Table 3
Tested values and selected value for the parameters of the neighborhood size or niche ($niche_s$) and the decomposition approach ($approach_d$).

Parameter	Tested values	Selected value
$niche_s$	1, 3, 5, 7, 9	3
$approach_d$	Tchebycheff, normalized Tchebycheff, boundary intersection	Boundary intersection

niche and the decomposition approach. Table 3 shows the tested values and the selected value for each parameter.

Regarding the method considered for selecting the final solution from the Pareto front (described in Section 4.4), the results of the experimental study have reported that the method of the consensus solution has obtained the best results, both in average ROUGE scores and in number of collections. Thus, the consensus solution has been used in the experimentation.

Finally, in order to provide statistically reliable results, 31 independent runs have been executed per experiment, showing the average ROUGE score. A compute node with 4 processors AMD Opteron Abu Dhabi 6376 2.3 GHz with 96-GB RAM and C/C++ programming language with the Eclipse IDE on Ubuntu 20.04.2.0 LTS (Focal Fossa) operating system have been used.

6.2. Analysis of the candidate parts for citation-based summarization

In this subsection, the most important parts of a collection are analyzed. As it has been defined, a collection can be separated into four main parts: abstract, body, and citing spans from the reference paper, and citation contexts from the citing papers. Henceforth, these parts will be identified as single letters: A for abstract, B for body, S for

Table 4
Average ROUGE scores obtained for the combinations formed with abstract (A), body (B), citing spans (S), and citation contexts (C). The best values are highlighted in bold.

Combination	ROUGE-1	ROUGE-2	ROUGE-3	ROUGE-4	ROUGE-L
A–––	0.42804	0.19249	0.09249	0.05413	0.38742
–B––	0.59949	0.19856	0.06918	0.02825	0.53628
––S–	0.62236	0.23953	0.10042	0.05305	0.55904
–––C	0.57593	0.20793	0.08636	0.04661	0.52139
AB––	0.59939	0.20302	0.07450	0.03351	0.53324
A–S–	0.63671	0.25383	0.10887	0.05621	0.57268
A––C	0.62153	0.25031	0.11073	**0.06138**	0.56443
–B–C	0.58915	0.19685	0.07305	0.03402	0.52739
––SC	0.63498	0.26050	0.10842	0.05619	0.56873
AB–C	0.62448	0.24684	0.10249	0.05139	0.55508
A–SC	**0.63685**	**0.26387**	**0.11244**	0.05860	**0.57444**

citing spans, and C for citation contexts. A total of eleven different combinations can be formed with these four parts, taking into account that the citing spans are a subset of the body, so they are not simultaneously present in any combination.

Table 4 reports the ROUGE scores obtained after applying the proposed approach to each combination.

The results reported in Table 4 show that the combination formed with the abstract, the citing spans, and the citation contexts (A–SC) has obtained the best results in ROUGE-1, ROUGE-2, ROUGE-3, and ROUGE-L scores, and the second best one in ROUGE-4 score. This combination has considered the citing spans instead of the body. The citing spans, by definition, represent the texts with the greatest impact and repercussion for the scientific community, since they substantiate the subsequent citations made to the reference paper in the citing papers of the scientific literature. Therefore, they are highly relevant for the quality of a citation-based summary, not being necessary to use the entire body of the reference paper. This idea is reinforced when the combinations with the body are compared with the corresponding combinations that change the body by the citing spans: –B–– vs ––S–, AB–– vs A–S–, –B–C vs ––SC, and AB–C vs A–SC; where the ROUGE scores are always better when the citing spans are used instead of the body.

The combination composed of the abstract and the citing spans (A–S–) has obtained the second best result in ROUGE-1 and ROUGE-L scores, and the third one in ROUGE-2, ROUGE-3, and ROUGE-4 scores. In addition, the combination formed with the citing spans and the citation contexts (––SC) has achieved the second best result in ROUGE-2 score, and the third one in ROUGE-1 and ROUGE-L scores. Comparing with the best combination, these two combinations have also considered the citing spans instead of the body, which strengthens the previous conclusion. The combination A–S– does not include the citation contexts, whereas the combination ––SC does not contain the abstract. Since the ROUGE scores obtained by them are very similar, it can be concluded that both parts are important for the generation of a citation-based summary. On the one hand, the abstract usually contains key phrases such as the paper's contributions, and on the other hand, the citation contexts may include relevant text written by other authors.

Therefore, the best combinations, A–SC, A–S–, and ––SC, have in common the presence of the citing spans and the absence of the body. Moreover, the combination with the best ROUGE scores also considers the abstract and the citation contexts, so both have provided a positive impact on the performance. Specifically, the combination A–SC has obtained an average execution time of 10.00 s, being much shorter than the average execution time of all combinations (23.61 s). Now, since the citing spans can also be divided into sections, the following experimentation focuses on the analysis of them, also keeping the parts of the abstract and the citation contexts.

Table 5
Average ROUGE scores obtained for the combinations formed with abstract (A), introduction from citing spans (S^I), results from citing spans (S^R), discussion from citing spans (S^D), and citation contexts (C). The best values are highlighted in bold.

Combination	ROUGE-1	ROUGE-2	ROUGE-3	ROUGE-4	ROUGE-L
AS^I – – C	0.63452	0.26386	0.11696	0.06386	0.57502
A – S^R – C	0.62047	0.24256	0.09900	0.04818	0.55947
A – – S^DC	0.64724	0.27221	0.12567	0.07020	0.58073
AS^IS^R – C	0.63602	0.25286	0.10589	0.05213	0.56923
AS^I – S^DC	**0.66010**	**0.28437**	**0.13233**	**0.07164**	**0.59519**
A – S^RS^DC	0.64505	0.26322	0.10884	0.05288	0.58030
$AS^IS^RS^D$C	0.63685	0.26387	0.11244	0.05860	0.57444

6.3. Analysis of the citing spans

This subsection analyzes the most important sections of the citing spans, i.e., S^I for Introduction, S^R for Results, and S^D for Discussion. With these sections, a total of seven different combinations can be made, in addition to the parts of the abstract and the citation contexts (which are fixed).

Table 5 presents the ROUGE scores obtained for all the combinations. As can be observed, the combination formed with the abstract, the introduction and discussion from the citing spans, and the citation contexts (AS^I–S^DC) has achieved the best results in all ROUGE scores. This combination has not included the results section, which indicates that this section is of low interest for producing a citation-based summary. The combination composed by the abstract, the discussion from the citing spans, and the citation contexts (A—S^DC) has obtained the second best results in the five ROUGE scores. In the same way, this combination has not considered neither the results nor the introduction sections.

Hence, the best combinations, AS^I–S^DC and A—S^DC, have in common the presence of the discussion and the absence of the results, and the use of the introduction provides a better performance. The reason is that the texts in the introduction and discussion supply the most relevant information about the reference paper, such as the motivation, contributions, qualitative analyses, or conclusions reported. Besides, the average execution time of the combination AS^I–S^DC has been 2.86 s, being the average execution time of all combinations 3.76 s. Therefore, in addition to achieve the best performance in ROUGE scores, it is one of the fastest.

In the same way as in this analysis, the following subsection focuses on the study of the sections of the citation contexts, since they can also be separated.

6.4. Analysis of the citation contexts

The main sections of the citation contexts (introduction, results, and discussion) are analyzed here. These sections will be named as: C^I for Introduction, C^R for Results, and C^D for Discussion. A total of seven different possible combinations can be formed with these sections, also taking into account the part of the abstract and the introduction and discussion from the citing spans (which are kept). Table 6 presents the ROUGE scores obtained for these combinations.

The results reported in Table 6 show that, in the case of the citation contexts, the discussion section is the most relevant one, followed by the results section. More specifically, the combination formed with the abstract, the introduction and discussion from the citing spans, and the results and discussion from the citation contexts (AS^IS^D–C^RC^D) has achieved the best results in ROUGE-2, ROUGE-3, and ROUGE-4 scores, and the second best ones in ROUGE-1 and ROUGE-L scores. Besides, the combination including the abstract, the introduction and discussion from the citing spans, and all sections from the citation contexts ($AS^IS^DC^IC^RC^D$) has achieved the best results in ROUGE-1 and ROUGE-L scores, and the second best result in ROUGE-3 score. These two combinations have in common the presence of the results

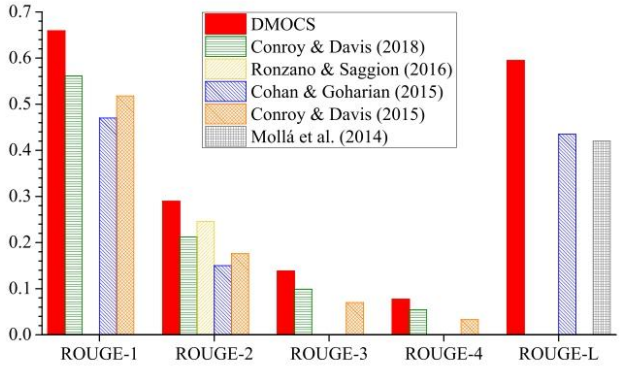


Fig. 3. Comparison of the average ROUGE scores obtained by the DMOCS approach and by the other approaches.

Table 6
Average ROUGE scores obtained for the combinations formed with abstract (A), introduction from citing spans (S^I), discussion from citing spans (S^D), introduction from citation contexts (C^I), results from citation contexts (C^R), and discussion from citation contexts (C^D). The best values are highlighted in bold.

Combination	ROUGE-1	ROUGE-2	ROUGE-3	ROUGE-4	ROUGE-L
$AS^IS^DC^I$ – –	0.64730	0.27178	0.12159	0.06576	0.58352
AS^IS^D – C^R –	0.64330	0.27333	0.12810	0.07345	0.58272
AS^IS^D – – C^D	0.65271	0.27993	0.13000	0.07185	0.58909
$AS^IS^DC^IC^R$ –	0.65587	0.27144	0.12174	0.06514	0.59083
$AS^IS^DC^I$ – C^D	0.65750	0.28623	0.13012	0.07152	0.59161
AS^IS^D – C^RC^D	0.65899	**0.28943**	**0.13835**	**0.07705**	0.59457
$AS^IS^DC^IC^RC^D$	**0.66010**	0.28437	0.13233	0.07164	**0.59519**

Table 7
Comparison of the average ROUGE scores obtained by the DMOCS approach and by the other approaches. The best values are highlighted in bold.

Approach	ROUGE-1	ROUGE-2	ROUGE-3	ROUGE-4	ROUGE-L
DMOCS	**0.65899**	**0.28943**	**0.13835**	**0.07705**	**0.59457**
Conroy and Davis (2018)	0.56100	0.21200	0.09900	0.05400	–
Ronzano and Saggion (2016)	–	0.24543	–	–	–
Cohan and Goharian (2015)	0.47000	0.15000	–	–	0.43500
Conroy and Davis (2015)	0.51800	0.17600	0.07000	0.03300	–
Mollá et al. (2014)	–	–	–	–	0.42000

and discussion of the citation contexts, so they are the most relevant sections for a citation-based summary. The introduction section is less relevant because the citations from the introduction section of the citing papers, sometimes, are less tightly related to the important content of the reference paper.

Finally, the combination AS^IS^D–C^RC^D has achieved the best results in the ROUGE scores among all combinations studied in the comprehensive analysis. Its average execution time is 2.62 s, whereas the average execution time obtained by all combinations has been 2.72 s. Moreover, it can be observed that the execution time of the best combinations has been progressively reduced, starting from 10.00 s to 2.62 s.

6.5. Comparison with other approaches

In this last subsection, the results reported by the DMOCS approach are compared with the existing ones in the scientific literature. All the works reviewed in Section 2 are included in this comparison.

Table 7 shows the results of the best combination of the DMOCS approach and the ones of the other approaches. The symbol "-" is shown for those approaches that do not present the result for the corresponding ROUGE score.

Table 8
Percentage improvement obtained by the DMOCS approach regarding the other approaches.

Approach	ROUGE-1	ROUGE-2	ROUGE-3	ROUGE-4	ROUGE-L
Conroy and Davis (2018)	+17.47%	+36.52%	+39.75%	+42.69%	–
Ronzano and Saggion (2016)	–	+19.93%	–	–	–
Cohan and Goharian (2015)	+40.21%	+92.95%	–	–	+36.68%
Conroy and Davis (2015)	+27.22%	+64.45%	+97.65%	+133.50%	–
Mollá et al. (2014)	–	–	–	–	+41.56%

As can be appreciated in Table 7, the DMOCS approach improves the results obtained by all the other approaches in all ROUGE scores. Fig. 3 shows the average ROUGE scores obtained by the DMOCS approach and all the compared approaches in a visual way.

Now, Table 8 reports the percentage improvements achieved by the DMOCS approach. These percentage improvements range between 17.47% and 133.50% for the ROUGE scores evaluated. Note that the percentage improvements are larger as the N-gram of ROUGE scores increases. This illustrates that the improvements of the produced summaries with respect to the ones from the other approaches are still larger when longer N-grams are used in the evaluation.

Finally, a visual example is presented with the aim of illustrating the performance of the proposed approach. Fig. 4 displays the abstract of a reference paper, the summary made by an expert, and the summary generated by the DMOCS approach for a collection from the TAC2014 Biomedical Summarization Track. First, the abstract of the reference paper, written by the authors, contains the principal ideas of their work according to their point of view. The second summary is made by an expert following the instructions given by TAC. This summary highlights the impact and contributions of the reference paper and subsequent discussions about it in the citing papers. Finally, the third summary is the one generated by the best combination of the DMOCS approach. This combination has considered the abstract of the reference paper, the introduction and discussion from the citing spans, and the results and discussion from the citation contexts (AS^IS^D–C^RC^D). As can be appreciated, the text of the generated summary has an important overlap with the text of the summary made by the expert. Furthermore, some of the sentences contained in the abstract are present in the generated summary. Hence, the summary produced by the DMOCS approach is a high-quality citation-based summary.

7. Conclusions

Scientific paper summarization is one of the most challenging summarization problems in the scientific literature. The involved methods try to supply a brief representation of the most important contents in

Abstract for Reference Paper "Aberrant Expression of Oncogenic and Tumor-Suppressive MicroRNAs in Cervical Cancer Is Required for Cancer Cell Growth" (Wang, X., Tang, S., Le, S.Y., et al., 2008)

MicroRNAs (miRNAs) play important roles in cancer development. By cloning and sequencing of a HPV16+ CaSki cell small RNA library, we isolated 174 miRNAs (including the novel miR-193c) which could be grouped into 46 different miRNA species, with miR-21, miR-24, miR-27a, and miR-205 being most abundant. We chose for further study 10 miRNAs according to their cloning frequency and associated their levels in 10 cervical cancer- or cervical intraepithelial neoplasia-derived cell lines. No correlation was observed between their expression with the presence or absence of an integrated or episomal HPV genome. All cell lines examined contained no detectable miR-143 and miR-145. HPV-infected cell lines expressed a different set of miRNAs when grown in organotypic raft cultured as compared to monolayer cell culture, including expression of miR-143 and miR-145. This suggests a correlation between miRNA expression and tissue differentiation. Using miRNA array analyses for age-matched normal cervix and cervical cancer tissues, in combination with northern blot verification, we identified significantly deregulated miRNAs in cervical cancer tissues, with miR-126, miR-143, and miR-145 downregulation and miR-15b, miR-16, miR-146a, and miR-155 upregulation. Functional studies showed that both miR-143 and miR-145 are suppressive to cell growth. When introduced into cell lines, miR-146a was found to promote cell proliferation. Collectively, our data indicate that downregulation of miR-143 and miR-145 and upregulation of miR-146a play a role in cervical carcinogenesis.

Summary made by an expert

Wang et al. identified a subset of miRNAs and showed their significant role in downregulation and upregulation in cervical cancer and pre-neoplastic lesions. miRNAs are regulatory, non-coding RNAs with 21-23 nucleotide in length that are found in cell differentiation and are responsible for post-transcriptional process. Authors used miRNA expression from cervical cancer-derived CaSki C-2 cell lines and from HPV-infected vaginal keratinocytes (CaSki cells). They reported the miR-24, miR-27a, and mir-205 as most abundant miRNAs. Using a clustering analysis, authors found 16 miRNAs that were upregulated and 25 miRNAs that were downregulated. miR-143 and miR-145 were found suppressive to HeLa cell growth and need to be downregulated in cervical cells. miR-146a has been found to function as a growth factor in upregulating the cancer process. The introduction of mir146a into cervical cancer cells increased cell doubling time and promoted cell proliferation which suggest that miR-146a works as an "oncogenic" miRNA. High level downregulated mir-146a in both cervical and normal tissues suggests that this miRNA is cervical cancer-specific. miR-15b, miR-16, miR24, and miR-155 are other miRNAs that have been found to be upregulated in cervical cancer. miR-126 was identified as a metastasis suppressing miRNA that is downregulated in cervical cancer. Authors found that more than 68% of the miR-21 isolated from CaSki cells had an additional C at the 3' end and 33% of the miR-205 isolated contained an extra 5' ends of either miRNA. The finding of this work could be beneficial in research study for cancer and drug target design.

Summary generated by DMOCS approach

MicroRNAs (miRNAs) play important roles in cancer development. By cloning and sequencing of a HPV16+ CaSki cell small RNA library, we isolated 174 miRNAs (including the novel miR 193c) which could be grouped into 46 different miRNA species, with miR 21, miR 24, miR 27a, and miR 205 being most abundant. This suggests a correlation between miRNA expression and tissue differentiation. Functional studies showed that both miR 143 and miR 145 are suppressive to cell growth. When introduced into cell lines, miR 146a was found to promote cell proliferation. Using miRNA array analyses for age matched normal cervix and cervical cancer tissues, in combination with northern blot verification, we identified significantly deregulated miRNAs in cervical cancer tissues, with miR 126, miR 143, and miR 145 downregulation and miR 15b, miR 16, miR 146a, and miR 155 upregulation. Upregulation of miR 146a in cervical cancer tissues, but not in HPV induced pre neoplastic lesions or in cancer derived cell lines indicates that miR 146 expression is cervical cancer specific. The expression of miR 146a has been found to be up regulated in papillary thyroid carcinoma, anaplastic thyroid cancer and cervical cancer, which suggests miR 146a could work as an "oncogenic" miRNA in these cancers. In addition, up regulation of miR 24 has also been observed in gastric and cervical cancers. One study reports upregulation of this miR as revealed by qRT PCR whereas a sequencing approach and microarray analysis point to a repression of miR 133b in tumor tissue.

Fig. 4. Abstract of a reference paper, summary made by an expert, and summary generated by the DMOCS approach for a collection of the dataset from the TAC2014 Biomedical Summarization Track.

a scientific paper, also taking into account the citations in papers published later. More specifically, citation-based summarization requires the use of the citations and their context both in the paper referred and in the subsequent papers. According to these methods, in addition to the reference paper itself, the citation contexts in the citing papers and their corresponding spans in the reference paper should be considered as the texts that best provide the main ideas.

A Decomposition-based Multi-Objective optimization algorithm for Citation-based Summarization (DMOCS) has been designed, developed, and applied for solving the citation-based summarization problem. The objective functions of the content coverage and the redundancy reduction have been formulated to be optimized in a simultaneous way, taking advantage of one of the main benefits of multi-objective approaches. Specifically, the decomposition-based optimization strategy has been successfully applied. In addition, the specifically designed problem-aware operators of crossover, mutation, and repair have definitely contributed to the good performance.

In addition, an extensive analysis of the most important parts and sections for citation-based summarization has been carried out. In this way, the parts that have the greatest impact in the quality of the summary have been identified. The citing spans from the reference paper, instead of the body, have shown to be especially important in generating a high-quality citation-based summary. Regarding the sections of the citing spans, the introduction and discussion have

provided a better performance. On the other side, as for the citation contexts from the citing papers, the results and discussion sections have generated the best scores, also showing that they provide quality to the summary. Besides, the results reported by the DMOCS approach have outperformed all those existing in the scientific literature. Specifically, the percentage improvements in the ROUGE scores are between 17.47% and 133.50%. These improvements are larger when the N-gram of ROUGE scores increases, which shows that the improvement of the DMOCS approach is highly relevant in this context.

As a future research line, developing new methods based on techniques such as word embeddings is of interest. These techniques can be applied in text representation by means of replacing the weight vectors generated with *tf-isf* scheme by those of word embeddings. This would take advantage of the inclusion of semantic information provided from text corpora. The *word2vec* model (Mikolov et al., 2013) is one of the most commonly used for word embeddings in the natural language processing field, which could improve the performance for citation-based summarization.

CRediT authorship contribution statement

Jesus M. Sanchez-Gomez: Software, Validation, Formal Analysis, Investigation, Data curation, Writing – original draft, Visualization. **Miguel A. Vega-Rodríguez:** Conceptualization, Methodology, Formal Analysis, Resources, Writing – review & editing, Supervision, Project administration, Funding acquisition. **Carlos J. Pérez:** Conceptualization, Methodology, Formal Analysis, Resources, Writing – review & editing, Supervision, Project administration, Funding acquisition.

Data availability

The authors do not have permission to share data.

Acknowledgments

This research has been supported by Ministry of Science, Innovation and Universities - Spain and State Research Agency - Spain (Projects PID2019-107299GB-I00 and PID2021-122209OB-C32 funded by MCIN/AEI/10.13039/501100011033), Junta de Extremadura - Spain (Projects GR21017 and GR21057), and European Union (European Regional Development Fund). Jesus M. Sanchez-Gomez is supported by Junta de Extremadura, Spain and European Union (European Social Fund) under the doctoral fellowship PD18057.

References

Altmami, N.I., Menai, M.E.B., 2022. Automatic summarization of scientific articles: A survey. J. King Saud Univ. - Comput. Inform. Sci. 34 (4), 1011–1028. http://dx.doi.org/10.1016/j.jksuci.2020.04.020.

Cohan, A., Goharian, N., 2015. Scientific article summarization using citation-context and article's discourse structure. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, pp. 390–400. http://dx.doi.org/10.18653/v1/D15-1045.

Conroy, J.M., Davis, S.T., 2015. Vector space models for scientific document summarization. In: Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing. Association for Computational Linguistics, pp. 186–191. http://dx.doi.org/10.3115/v1/W15-1525.

Conroy, J.M., Davis, S.T., 2018. Section mixture models for scientific document summarization. Int. J. Digit. Libraries 19 (2), 305–322. http://dx.doi.org/10.1007/s00799-017-0218-6.

Das, I., Dennis, J.E., 1998. Normal-boundary intersection: A new method for generating the Pareto surface in nonlinear multicriteria optimization problems. SIAM J. Optim. 8 (3), 631–657. http://dx.doi.org/10.1137/S1052623496307510.

Deb, K., 2015. Multi-objective evolutionary algorithms. In: Springer Handbook of Computational Intelligence. Springer, pp. 995–1015. http://dx.doi.org/10.1007/978-3-662-43505-2_49.

El-Kassas, W.S., Salama, C.R., Rafea, A.A., Mohamed, H.K., 2021. Automatic text summarization: A comprehensive survey. Expert Syst. Appl. 165, 113679. http://dx.doi.org/10.1016/j.eswa.2020.113679.

Ho-Huu, V., Hartjes, S., Visser, H.G., Curran, R., 2018. An improved MOEA/D algorithm for bi-objective optimization problems with complex Pareto fronts and its application to structural optimization. Expert Syst. Appl. 92, 430–446. http://dx.doi.org/10.1016/j.eswa.2017.09.051.

Iqbal, S., Hassan, S.-U., Aljohani, N.R., Alelyani, S., Nawaz, R., Bornmann, L., 2021. A decade of in-text citation analysis based on natural language processing and machine learning techniques: An overview of empirical studies. Scientometrics 126 (8), 6551–6599. http://dx.doi.org/10.1007/s11192-021-04055-1.

Jaszkiewicz, A., 2002. On the performance of multiple-objective genetic local search on the 0/1 knapsack problem - A comparative experiment. IEEE Trans. Evol. Comput. 6 (4), 402–412. http://dx.doi.org/10.1109/TEVC.2002.802873.

Jebari, C., Herrera-Viedma, E., Cobo, M.J., 2021. The use of citation context to detect the evolution of research topics: A large-scale analysis. Scientometrics 126 (4), 2971–2989. http://dx.doi.org/10.1007/s11192-020-03858-y.

Ji, J., Guo, Y., Gong, D., Tang, W., 2020. MOEA/D-based participant selection method for crowdsensing with social awareness. Appl. Soft Comput. 87, 105981. http://dx.doi.org/10.1016/j.asoc.2019.105981.

Kumar, Y., Kaur, K., Kaur, S., 2021. Study of automatic text summarization approaches in different languages. Artif. Intell. Rev. 54 (8), 5897–5929. http://dx.doi.org/10.1007/s10462-021-09964-4.

Li, J., 2020. ROUGE metric. URL: https://pypi.org/project/rouge-metric/. (Last Accessed: 2 December 2022).

Lin, C.-Y., 2004. ROUGE: A package for automatic evaluation of summaries. In: Text Summarization Branches Out: Proceedings of the ACL-04 Workshop, Vol. 8. Association for Computational Linguistics, pp. 74–81.

Ma, X., Liu, F., Qi, Y., Li, L., Jiao, L., Deng, X., Wang, X., Dong, B., Hou, Z., Zhang, Y., Wu, J., 2016. MOEA/D with biased weight adjustment inspired by user preference and its application on multi-objective reservoir flood control problem. Soft Comput. 20 (12), 4999–5023. http://dx.doi.org/10.1007/s00500-015-1789-z.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J., 2013. Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems, Vol. 26. Curran Associates, Inc., pp. 3136–3144.

Mollá, D., Jones, C., Sarker, A., 2014. Impact of citing papers for summarisation of clinical documents. In: Proceedings of the Australasian Language Technology Association Workshop, Vol. 2014. Association for Computational Linguistics, pp. 79–87. http://dx.doi.org/10.13140/2.1.2366.1126.

Porter, M., 2021. The porter stemming algorithm. URL: http://www.tartarus.org/martin/PorterStemmer/. (Last Accessed 2 December 2022).

Radev, D.R., Jing, H., Styś, M., Tam, D., 2004. Centroid-based summarization of multiple documents. Inf. Process. Manage. 40 (6), 919–938. http://dx.doi.org/10.1016/j.ipm.2003.10.006.

Ronzano, F., Saggion, H., 2016. An empirical assessment of citation information in scientific summarization. In: Natural Language Processing and Information Systems. Springer, pp. 318–325. http://dx.doi.org/10.1007/978-3-319-41754-7_30.

Salton, G., Buckley, C., 1988. Term-weighting approaches in automatic text retrieval. Inf. Process. Manage. 24 (5), 513–523. http://dx.doi.org/10.1016/0306-4573(88)90021-0.

Sanchez-Gomez, J.M., Vega-Rodríguez, M.A., Pérez, C.J., 2019. Comparison of automatic methods for reducing the Pareto front to a single solution applied to multi-document text summarization. Knowl.-Based Syst. 174, 123–136. http://dx.doi.org/10.1016/j.knosys.2019.03.002.

TAC 2014 Biomedical Summarization Track, 2017. Text analysis conference (TAC) 2014 biomedical summarization track. https://tac.nist.gov/2014/BiomedSumm/index.html. (Last Accessed 2 December 2022).

Tang, L., Zuo, X., Wang, C., Zhao, X., 2015. A MOEA/D based approach for solving robust double row layout problem. In: Evolutionary Computation (CEC), 2015 IEEE Congress on. IEEE, pp. 1966–1973. http://dx.doi.org/10.1109/CEC.2015.7257126.

Wang, M., Zhang, J., Jiao, S., Zhang, X., Zhu, N., Chen, G., 2020. Important citation identification by exploiting the syntactic and contextual information of citations. Scientometrics 125 (3), 2109–2129. http://dx.doi.org/10.1007/s11192-020-03677-1.

Widyassari, A.P., Rustad, S., Shidik, G.F., Noersasongko, E., Syukur, A., Affandy, A., Setiadi, D.R.I.M., 2022. Review of automatic text summarization techniques & methods. J. King Saud Univ. - Comput. Inform. Sci. 34 (4), 1029–1046. http://dx.doi.org/10.1016/j.jksuci.2020.05.006.

Zhang, Q., Li, H., 2007. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. IEEE Trans. Evol. Comput. 11 (6), 712–731. http://dx.doi.org/10.1109/TEVC.2007.892759.

Zhao, F., Chen, Z., Wang, J., Zhang, C., 2017. An improved MOEA/D for multi-objective job shop scheduling problem. Int. J. Comput. Integr. Manuf. 30 (6), 616–640. http://dx.doi.org/10.1080/0951192X.2016.1187301.

Zhou, H., Qiao, J., 2019. Multiobjective optimal control for wastewater treatment process using adaptive MOEA/D. Appl. Intell. 49 (3), 1098–1126. http://dx.doi.org/10.1007/s10489-018-1319-7.