JADAVPUR UNIVERSITY

MASTER DEGREE THESIS

SOME STUDIES ON ABSTRACTIVE MULTI-DOCUMENT SUMMARIZATION

A thesis submitted in partial fulfilment of the requirement for the degree of

Master of Technology in Computer Technology

in the

Department of Computer Science and Engineering

By

Bijay Kar

University Roll No : **001910504020**

EXAMINATION ROLL NO : **M6TCT22021**

Registration No : **149854 of 2019-20**

Under the Guidance Of

Prof. (Dr.) Kamal Sarkar

Department of Computer Science & Engineering

JADAVPUR UNIVERSITY

KOLKATA- 700032

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

FACULTY OF ENGINEERING AND TECHNOLOGY
JADAVPUR UNIVERSITY

CERTIFICATE OF RECOMMENDATION

This is to certify that the thesis entitled **"Some Studies on Abstractive Multi-Document Summarization"** has been carried out by BIJAY KAR (University Roll No – **001910504020** Examination Roll No **- M6TCT22021,** Registration number- **149854 of 2019-20** and) under the guidance and supervision of Prof. (Dr.) **Kamal Sarkar**, Department of Computer Science and Engineering, JADAVPUR UNIVERSITY, KOLKATA, is being presented for partial fulfilment of the degree of Master of Technology during the academic year 2021-22. The research results presented in the thesis have not been included in any other paper submitted for the award of any degree in any other University or Institute.

Prof. (Dr.) **Kamal Sarkar** (Thesis Supervisor)
Department of Computer Science & Engineering
Jadavpur University, Kolkata - 700032

Countersigned,

.....
Prof. (Dr.) Anupam Saha
Head of the Department
Department of Computer Science & Engineering
Jadavpur University, Kolkata - 700032

.....
Prof. Chandan Mazumder,
Dean,
Department of Computer Science & Engineering
Jadavpur University, Kolkata - 700032

FACULTY OF ENGINEERING AND TECHNOLOGY
JADAVPUR UNIVERSITY

Declaration of Originality and Compliance of Academic Ethics

I now declare that this thesis entitled **"Some Studies on Abstractive Multi-Document Summarization"** contains a literature survey and original research work by the undersigned candidate, as a part of her M. Tech degree in Computer Technology.

All information in this document has been obtained and presented by academic rules and ethical conduct.

I have fully cited and referenced all materials and results that are not original to this work as these rules and conduct requirements.

Name (in block): **BIJAY KAR**
University Roll No. **001910504020**
Examination Roll Number: **M6TCT22021**.
Registration Number: **149854 of 2019-20**

Thesis Title: **"Some Studies on Abstractive Multi-Document Summarization"**
Signature with date:

FACULTY OF ENGINEERING AND TECHNOLOGY
JADAVPUR UNIVERSITY

CERTIFICATE OF APPROVAL

(Only in case the thesis is approved)

The foregoing project is hereby accepted as a credible study of an engineering subject carried out and presented in a manner satisfactory to warrant its acceptance as a prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed, or conclusion drawn therein, but approve the project only for the purpose for which it is submitted.

_____

Signature of Examiner 1

Date:

_____

Signature of Examiner 2

Date:

Acknowledgement

I would like to express my honest and sincere thanks and humble gratitude to my honourable teacher and guide Prof. Kamal Sarkar, Professor of the Department of Computer Science & Engineering, Jadavpur University, for his exclusive guidance and full support in completing and producing this project successfully. I am very much obliged to him for the constant encouragement and continuous inspiration that he has given to me. These words are only a token of my deep respect towards him for all he has done to take my project to the present shape.

I would like to thank Mrs. Sohini Roychowdhury Di for all her support and valuable suggestions to the activities of the project in all phases of the project.

Finally, I convey my real sense of gratitude and thankfulness to my family members, especially my Wife and Elder Brother, for being my constant source of inspiration and positive thoughts ; and to my Parents for their unconditional love and support throughout my life without which I would hardly be capable of completing this huge work.

……..
Name (in block): **BIJAY KAR**
University Roll No. **001910504020**
Examination Roll Number: **M6TCT22021**.
Registration Number: **149854 of 2019-20**

TABLE OF CONTENT

Chapter 1: ABSTRACT

The main objective is to propose a framework for abstractive summarization of multi-documents, which aims to select contents of summary from the semantic representation of the documents but not from the source document sentences itself. Therefore, the goal is to extract the gist of the content from the multi-documents in easily readable, grammatically correct format. In this framework, contents of the source documents are represented by predicate argument structures after implementation of semantic role labelling. Then source text is analysed semantically based on the predicate argument structure. Post this step, the cluster of semantic similar predicate argument structures across the text are formed. The proposed framework differs from other abstractive summarization models. First, it employs semantic role labelling for semantic representation of text. Secondly, it analyses the source text semantically by utilizing semantic similarity measure to cluster semantically similar predicate argument structures across the text; and finally, it ranks the predicate argument structures based on features weighted by genetic algorithm (GA) and generates the abstract summary of the folder.

Chapter 2: Introduction

Extensive summarization is a technique that creates a more concise and accurate version of one or more text documents. Finding information from a huge text corpus or the internet requires the use of automatic text summarization. What originally began as a text summation of a single document has since expanded into the generation of multi-document summaries.

It is the process of extracting the most essential information and removing the redundant information from a text to produce a summarized version for a particular task and user so that it takes less time to extract the most important information from the document. It is essential for the summary to be fluent, continuous and depict the most important information of the entire text.

In the era of Digitalization more and more digitized text is available, especially with the development of the Internet and making it affordable to the common people in the society. In the current era of the information overload through the web, it is extremely important to develop a tool that can summarize the information by preserving the main contents of the source document. The automatic multi-document summarization is now one of the major topics in NLP (Natural Language Processing) and now one of the key topics of development in recent years. It becomes extremely important how to help people effectively and efficiently extract the information from the data. Several techniques have been implemented till now and multi-document text summarization is one of them.

In the era of web, it becomes difficult for the common people to go through many documents which sometimes contain many similar topics repetitively which is in turns very time consuming and sometimes because of the repetitive similar information important information might be missed out. So summarizing the synthesized common information from multiple text documents will be very useful for users and it will take less time to find the key information in the text. The framework is designed in such a way that multi-documents are processed, shortened and semantic similarity matrices of the sentences are generated and then use genetic algorithms to generate the precise abstractive summary.

Types of Text Summarization:

Text Summarization method can be segregated into two different categories: Extractive and Abstractive Method.

Extractive Method

- It is one of the traditional methods which is developed initially. The main purpose is to select the best effective sentence and add it to the summary. It is based on techniques like sentence extraction, statistical analysis, discourse structure and various machine learning techniques have been implemented on this to extract the summary sentences.

Abstractive Method

- It is a more advanced method; many advancements keep coming out frequently. The purpose is to select the important sentence, interpret the sentence and generate the sentences having the same semantic sense. This process ensures that the important information is conveyed precisely in the shortest possible text. In Abstractive sentences the summary is not generated from the original sentences, it is based on the semantic similarity of the original sentences so that the extracted summary becomes the semantic representation of the original sentences. It requires deeper analysis of the text, compressed version of the original which might be from the original text or might not be from the original text. The goal is to improve the focus of the summary, reduce the unnecessary and similar information and generate a good, compressed version of the original text.

Chapter 3: Literature Survey

There is prior research on the abstractive summaries using different methods - Syntactic based approach which apply syntactic parser to analyse and represent the text syntactically Usually, in this approach, syntactic parser identifies verbs and nouns which are used for text representation and these texts are further processed to generate the summary of the sentences. On the other hand, semantic based approach aims to produce abstractive summary from semantic representation of document text. Different semantic representations of text used in the literature are ontology based and template-based representation. Titov and Klementiev discover the differences between the syntactic based approach

and semantic based approach. Syntactic based approach does not represent the meaning of the source sentence but only extract the syntactic part of the source sentences. All the linguistic based approaches proposed for the abstractive summarization rely on the syntactic representation of the source document. These approaches employ a syntactic parser to represent the source text syntactically. The main limitation of the syntactic based approach is the lack of semantic representation of the source text as summarization requires deep analysis of the source text and semantic the source text, therefore semantic representation is a more suitable approach for the summarization of the text.

On the other side there are some semantic based approaches for the abstractive summary generation. Template based multi-document summarization system produces abstractive summary from multiple newswire/newspaper documents depending on the output of the information extraction. This approach uses template for topic representation of document. This method has limitations because of the linguistic patterns and extraction rules template is also created by humans which is also time consuming Also this method cannot take care about the similarities and differences across multiple documents.

Then a fuzzy ontology-based approach was used for Chinese news summarization to model some uncertain information and hence can better describe the domain knowledge. But this methodology has many limitations. First domain ontology and Chinese dictionary has to be defined by some expert who has complete knowledge on this and also it is time consuming and second is this method is limited to Chinese language might not be applicable to other language.

Fuzzy Ontology approach was proposed for the Chinese News Summarization. This approach was used basically to model the uncertain information therefore it requires domain knowledge. This approach is an extension of the domain ontology-based model and has several limitations. First, domain experts are required for defining the domain ontology which is time consuming as it is manual work which needs to be done. Secondly, this approach is limited to Chinese news, and might not be applicable to other languages apart from Chinese language.

Another approach named Fully Abstractive approach to guided summarization generates short and very precise summary of the cluster of documents. This methodology uses a rule based customized Information extraction module which is integrated with content selection and generation in order to write short, very specific and precise summary of the cluster of documents. The major drawback of this Fully Abstractive approach to guided summarization is that it uses a handwritten Information extraction module which is integrated with content selection and generation which is a very time-consuming process.

Another approach is proposed for generating the summarization which is abstractive in nature for a Multimodal document. Multimodal documents are basically different from the normal source document as a multimodal document contains both text and graphical contents. This approach is a little bit different as we cannot extract the information from the graphics content and make a summary out of it. Therefore, a semantic model is created based on the knowledge representation of concepts which captures the information content both from the text of the sentences and also from the Graphics. But the limitation of the framework is that semantic models are manually created models by human beings. As the model preparation requires manual activity therefore it is a very time-consuming process.

Semantic Graph Reduction approach of abstractive summary generation is an approach of summarizing the source document by creating a Rich Semantic Graph. Rich semantic graph is basically a graph where verbs and nouns of documents are represented as nodes and semantic and linguistics relationships between the verbs and nouns are represented by edges. Like any other approaches this approach also has some limitations like this approach also relies on manually built ontology which requires time to build up the ontology.

The main limitation for all the semantic based approaches are basically for abstractive summary generation they are dependent on the human interaction which in one side taking time also may be error prone, As all the model are dependent on human expert based domain ontology and rules so it is major drawback for all semantic based approaches,

But in this paper, we are trying to remove the limitation of human expert-based domain ontology and rules by using a new technique called Semantic Role Labelling in short form called SRL. Semantic Role Labelling techniques are used to generate the semantic representation of the source document automatically. It is basically used to detect the semantic argument associated with the predicate or verb and then classify them based on their specific roles. Text retrieval, information extraction, text categorization, and sentiment analysis are just a few of the tasks that SRL has been used for. Introduced a study in the field of text summarising that coupled semantic role labelling with the general statistical technique (GSM) to identify relevant phrases for single document extractive summary. To compute the phrase similarity score, they initially used SRL and semantic similarity measurements. Second, the sentence score was computed using a standard statistical procedure that did not take into consideration the weights of the features. Finally, the aggregate score for each sentence in the document is assigned using the sentence scores acquired from both approaches, and the top ranked sentences are retrieved using a 20% compression rate. However, our work differs in the following ways. We concentrate on multi-document abstractive summarising, while single-document extractive summarization is our main focus. In the second step of the framework, we use SRL to represent text in the document collection in a semantic way. On the other hand, the sentence semantic similarity score was computed using SRL and a semantic similarity measure. To the best of our knowledge, the semantic role labelling (SRL) technique, which makes use of a semantic role parser, has never been used to represent text semantically in multi-document abstractive summarization.

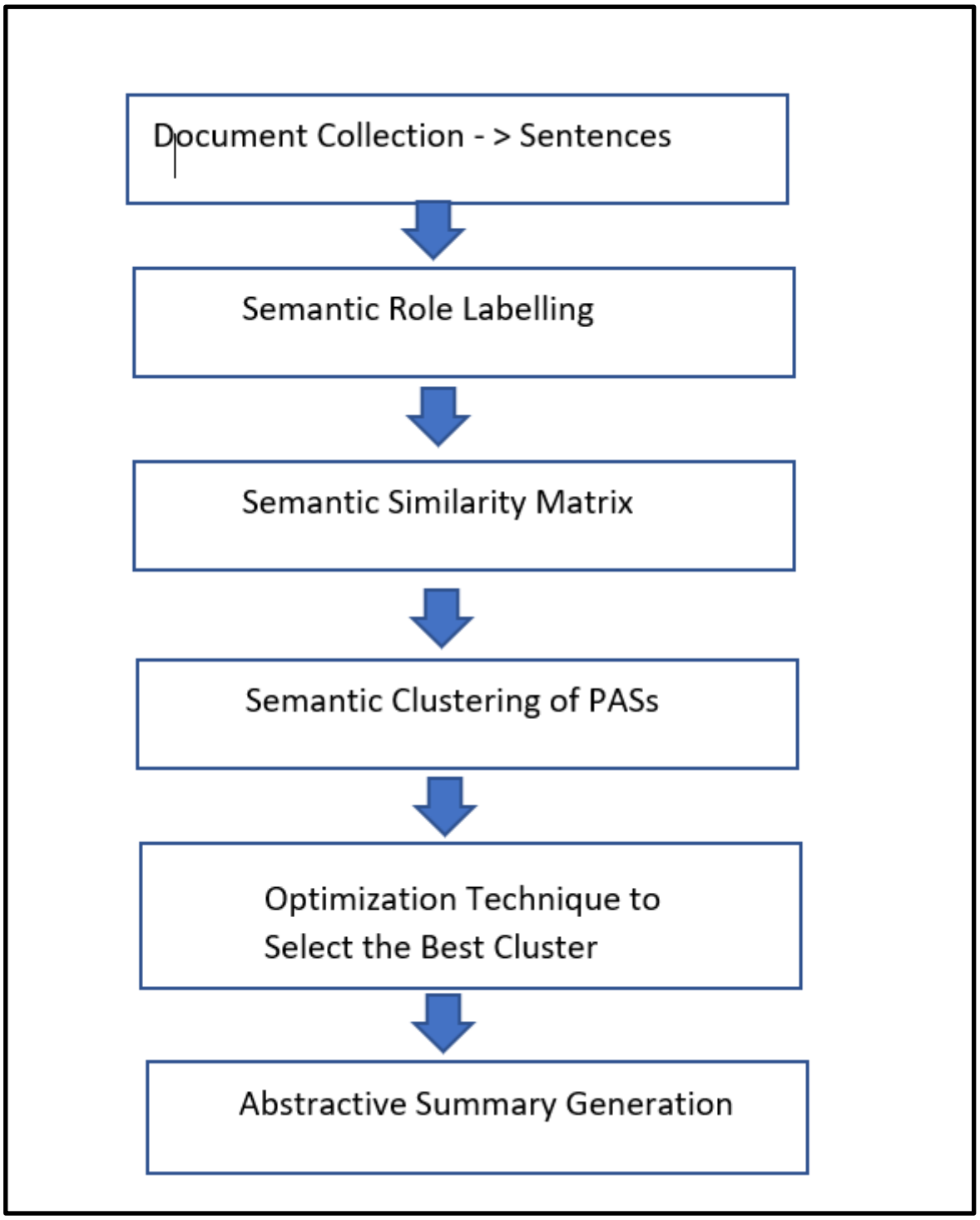
As a result, this thesis paper presents a framework that will use SRL to represent text semantically in order to provide a suitable abstractive summary. In a few ways, the framework for multi-document abstract summarization provided in this paper differs from earlier approaches to abstract summarization. It starts by extracting predicate argument structure (semantic representation) from the contents of source documents using semantic role labelling. Second, it uses a semantic similarity measure to analyse the text semantically in order to cluster semantically similar predicate argument structures across the text; and third, it ranks the predicate argument structures using a feature

weighted and optimised genetic algorithm, because text features are sensitive to the quality of the generated summary.

Chapter 4: Methodology

4.1. Overview of approach

Figure 1 depicts the framework of our proposed strategy. When we have a document collection that needs to be summarised, we first divide it into sentences, with each sentence being preceded by the matching document number and sentence position number. Following that, the SENNA semantic role labeller is used to extract predicate argument structure from each sentence in the document collection. The similarities of predicate argument structures (PASs) are estimated pairwise using Jiang's semantic similarity measure and the edit distance algorithm in the semantic similarity matrix computation phase. We use an agglomerative hierarchical clustering (HAC) technique based on the average linkage approach to cluster semantically comparable predicate argument structures once the similarity matrix for PASs is produced. The number of clusters will be determined by the compression level. The PASs in each cluster are scored using features, then weighted and optimised using a genetic algorithm, and the top ranked predicate argument structures from each cluster are chosen. Finally, from the selected predicate argument structures, the SimpleNLG realisation engine is used to create sentences. The final abstractive summary will be made out of the sentences generated.


Document Collection - > Sentences
Semantic Role Labelling
Semantic Similarity Matrix
Semantic Clustering of PASs
Optimization Technique to
Select the Best Cluster
Abstractive Summary Generation

4.2. Semantic Role Labelling

Semantic role labelling is a procedure in natural language processing that assigns labels to words or phrases in a sentence that reflect their semantic role in the sentence, such as agent, purpose, result , and the adjunctive arguments of the predicate such as Locative, Temporal and Manner. The goal of semantic role labelling (SRL) is to identify the syntactic constituents/arguments of a sentence in relation to the sentence predicates, as well as the semantic roles of the arguments (such as Agent, Patient, and Instrument) and the predicate's adjective arguments (such as Locative, Temporal, and Manner). SRL's main goal is to figure out what semantic relationship a predicate has with its participants/constituents. Because abstractive summarization necessitates a more in-depth semantic examination of text, this work uses semantic role labelling to derive predicate argument structure from the document collection's phrases. The framework makes use of the SENNA tool, which is open source and non-commercial, and produces a number of natural language processing (NLP) predictions, including semantic role labelling (SRL), part-of-speech (POS) tagging, named entity recognition (NER), and chunking (CHK). Here we used the SENNA tool for the SRL , POS tags and NER.

We begin by breaking down the document collection into sentences, with each sentence being preceded by its associated document number and sentence position number. After that, the SENNA semantic role labeller is used to interpret each sentence and label the semantic word phrases correctly. Semantic arguments are the terms used to describe these words. As stated in Table 1, semantic arguments can be divided into two categories: core arguments (Arg) and ancillary arguments (ArgM). We use A0 for subject, A1 for object, and A2 for indirect object as core arguments, with ArgM-LOC for location and ArgM-TMP for time as adjunctive arguments for predicate (Verb) V in this study. To avoid losing essential terms that contribute to the meaning of the sentence, we evaluate all of the entire predicates connected with the single sentence structure, as well as the actual predicate of the sentence. If a predicate has at least two semantic arguments, we consider it complete. Each sentence in the document collection has a semantic representation based on the extracted predicate argument structure. A simple predicate argument structure represents a sentence with only one complete

predicate, whereas a composite predicate argument structure represents a sentence with multiple complete predicates.

Processing of Predicate Argument Structure

After obtaining the predicate argument structures (PASs), they are separated into meaningful words or tokens, and stop words are removed. Using the Porter stemming technique, the words in PASs are stemmed to their base form. The SENNA POS tagger is then used to assign part of speech (POS) tags or grammatical roles to each term of semantic arguments (connected with the predicates). NN stands for noun, V for verb, JJ for adjective, and RB for adverb, among other POS tags. This phase is necessary because the semantic arguments of the predicates will be compared based on the words' grammatical responsibilities. Only terms of the semantic arguments of the predicates designated as nouns (NN) are compared in this study, while the remainder are discarded.

In the semantic arguments of predicate, we also use SENNA NER to recognise named things such as individual names (cabral) and organisation names (civil defence). This study analyses predicate argument structures based on noun–noun, verb–verb, location–location, and time–time arguments. These named entities are recorded for each predicate argument structure (PAS) and are required in a subsequent phase for grading the PAS based on proper noun characteristic. As a result, the framework only retrieves tokens from the predicate argument structure that have been tagged as noun, verb, location, and time in previous steps. In the comparison, all PASs associated with the sentence will be included. After the nouns, verbs, and additional arguments (time and location, if any) have been retrieved, the predicate argument structures will be formed based on Noun , Verb and other arguments (Location and time)

The noun from PAS P1's semantic argument A1 will be compared to both the nouns from PAS P2's semantic argument A1. Because there is no comparable temporal argument in P2, the temporal (time) semantic argument in P2 is skipped from comparison.

4.3. Semantic Similarity Matrix Calculation

This phase's goal is to create a semantic similarity score matrix for each pair of predicate argument structures. Similarity of predicate argument structures (PASs) is computed pair by pair in this phase using admissible comparisons of noun–noun, verb–verb, location–location, and time–time. Among all the semantic similarity measures, the Jiang and Conrath measure has the closest association with human judgement, according to experimental results in the literature. As a result, for computing semantic similarity between each pair of PASs, this study uses Jiang's semantic similarity metric. Jiang's metric is based on information content, and each idea in the WordNet holds specific information. The similarity of two conceptions is determined by the information they share, according to this metric.

Given two sentences S_i and S_j, the similarity score between predicate argument structure (PAS) k of sentence $Si(v_{ik})$ and PAS l of sentence $S_j(v_{jl})$ is calculated using Eq. (5), where $sim_p(v_{ik},v_{jl})$ is the sum of similarities between predicates (verbs) determined using Eq. (2), and $sim_{arg}(v_{ik},v_{jl})$ is the sum of similarities between the predicates' corresponding arguments determined (1). For computing similarity between noun terms in the semantic arguments of predicate argument structures and verbs of predicate argument structures, both equations (1) and (2) use Jiang's semantic similarity measure. Simplicity between corresponding temporal arguments, such as $sim_{tmp}(v_{ik},v_{jl})$,is computed using Eq. (3), and similarity between corresponding location arguments, such as $sim_{loc}(v_{ik},v_{jl})$is computed using Eq (4). The time and geographical arguments may not be found in Jiang's Measure because it is based on WordNet.

The similarity score between the two predicate argument structures is computed using Eqs. (1)–(5).

$$sim_{arg}(v_{ik},v_{jl}) = sim(A_{0i}, A_{0j}) + sim(A_{1i}, A_{1j}) + sim(A_{2i}, A_{2j}) \quad \text{.....(1)}$$

$$sim_p(v_{ik},v_{jl}) = (sim(Pi, Pj)) \quad \text{----------} \quad (2)$$

$$sim_{tmp}(v_{ik},v_{jl}) = (sim(Tmp_i, Tmp_j)) \quad \text{……..(3)}$$

$$sim_{loc}(v_{ik},v_{jl}) = (sim(Loc_i, Loc_j)) \quad \text{……..(4)}$$

Eqs. (1)–(4) are combined to give Eq. (5) as follows

$$sim(v_{ik},v_{jl}) = sim_p(v_{ik},v_{jl}) + [sim_{arg}(v_{ik},v_{jl}) + sim_{tmp}(v_{ik},v_{jl}) + sim_{loc}(v_{ik},v_{jl})]$$

……..(5)

For example concept C1 and concept C2 to compute the similarity score of the two terms/concepts in the semantic argument 'A0' of predicates P1 and P2. Jiang's measure computes the least common subsumer (lso) of two concepts using WordNet, then determines IC(C1), IC(C2), and IC(lso (C1, C2) using WordNet. The information content (IC) of a concept is determined by calculating the probability of a concept appearing in a large text corpus and is measured as follows:

$$IC(C) = -\log P(C) \quad \text{……..(6)}$$

where P(C) is the probability of occurrence of concept ‘C’ and is computed as follows:

$$P(C) = Freq(C)/N \quad \text{……..(7)}$$

where Freq (C) is the number of occurrences of concept ‘C’ in the taxonomy and N is the maximum number of nouns. Jiang’s measure calculates the semantic distance to obtain semantic similarity between any two concepts as follows:

$$Jiangdist(C1, C2) = IC(C1) + IC(C2) - 2 \times IC(lso(C1, C2))$$

Jiang's measure uses WordNet to establish that the word is the most common subsumer of the two concepts 'hurricane' and 'cyclone,' as seen below.

< entity < physical entity < process < phenomenon < natural phenomenon < physical phenomenon < atmospheric phenomenon < storm < windstorm < cyclone < entity < physical entity < process <

phenomenon < natural phenomenon < physical phenomenon< atmospheric phenomenon < storm < windstorm < cyclone < hurricane

The similarity of other concepts/terms is determined in the same manner. However, $sim_{tmp}(v_{ik}, v_{jl})$ and $sim_{loc}(v_{ik}, v_{jl})$ are set 0 as there are no temporal and location arguments for comparison in both predicate argument structures.

In order to normalize the result in the range of [0,1], we use a scaling factor $e^{-K * sim(v_{ik}, v_{jl})}$ where K is constant set to 0.05(optimal value).

The semantic similarity matrix $M_{i,j}$ is created from the similarity scores of the predicate argument structure Pi and Pj once the semantic similarity score for each pair of predicate argument structure (PAS) is attained.

$M_{i,j} = M_{sim}(Pi, Pj)$ if i <> j

0 otherwise

where $M_{sim}(Pi, Pj)$ refers to the semantic similarity score of predicate argument structure Pi and Pj in the Matrix $M_{i,j.}$

4.4. Semantic Clustering of PAS

Agglomerative hierarchical clustering is a well-known technique in the hierarchical clustering method, which is relatively old but still effective in a variety of applications . Agglomerative hierarchical clustering (HAC) has five well-known linking methods: single linkage, complete linkage, average linkage, ward and centroid technique. The literature studies indicated that average linkage is the best suited method for document clustering based on several measurements (Entropy and F-Score,

as well as the Kendall W test). As a result, this research makes use of the HAC algorithm, which is based on the average linkage approach. The value at location (i j) represents the semantic similarity between the ith and jth predicate argument structures. The value at point (i j) in the semantic similarity matrix is defined as semantic similarity between the ith and jth clusters, assuming that the similarity matrix is constructed starting with each predicate argument structure as a separate cluster. The pseudo code for clustering predicate argument structures that are similar is shown below.In this study, we consider 20% compression rate of summary.

Pseudo code for agglomerative clustering algorithms

Input: Semantic Similarity Matrix

Output: Clusters of similar predicate argument structures

a. Merge the two clusters that are more similar

b. Update the semantic similarity matrix to represent the pairwise similarity between the newest cluster and the original cluster based on the average linkage method.

c. Repeat step 1 and 2 until the compression rate of summary is reached.

Average linkage method:

Average-linkage is where the distance between each pair of observations in each cluster is added up and divided by the number of pairs to get an average inter-clustering distance.

Average linkage clustering - the distance between two clusters is defined as the average distance between all pairs of objects, where each pair is made up of one object from each group

4.5. Optimization Technique for selection of PAS from Each of the Cluster

The goal of this phase is to choose high-ranking predicate argument structures from each cluster using a genetic algorithm that weights and optimises features (GA). To begin, SENNA SRL extracts the matching predicate argument structure (PAS) for each sentence in the document collection. For each predicate

rgument structure (PAS) in the document collection, the features listed below are extracted, and each PAS is represented by a vector containing the weight of these features. P = { P_F1, P_F2, P_F3,.., P_F10 }

4.5.1. Different Text features

4.5.1.1. Title Features : The number of matched contents words in the predicate argument structure (PAS) and the title of a document are used to identify this feature.

P_F1 = Number of title words in PAS / Number of words in document title

```
p_F1(predicate,document):
  title = document.title
  titleWords = title.split()
  len_title_words = len(titleWords)
  count = 0
  for x in predicate.getWords():
    if x in titleWords:
      count = count+1
  p_F1 = count / len_title_words
  return p_F1
```

4.5.1.2. Length of predicate argument structure.

The normalized length of the PAS is the ratio of the number of words in the PAS to the number of words in the document's longest PAS.

P_F2 = Number of words occurring in the PAS / Number of words occurring in the longest PAS of the document

```
longestPredicate(document):
  longestpredicate = None
  length=0
  for p in document.getPredicates():
    lengthofpredicate= p.lengthOfPredicate()
    if lengthofpredicate > length:
      length = lengthofpredicate
      longestpredicate = p
  return longestpredicate
```

```
longestPredicateGivenPredicates(predicates):
  longestpredicate = None
  length=0
  for p in predicates:
    lengthofpredicate= p.lengthOfPredicate()
    if lengthofpredicate > length:
      length = lengthofpredicate
      longestpredicate = p
  return longestpredicate

p_F2(predicate,document):
  return predicate.lengthOfPredicate()/longestPredicate(document).lengthOfPredicate()
```

4.5.1.3. PAS to PAS Similarity

The semantic similarity between P and other predicate argument structures in the document collection is computed for each predicate argument structure P using Eq 5 mentioned in the above. Once each predicate argument structure (PAS) has a similarity score, the score for this feature is calculated by dividing the total of similarities of PAS P with all other PASs by the maximum of summary in the document collection.

P_F3 = Sum(Similarity Score of PAS (Pi , Pj)) / Max (Sum (Similarity Score of PAS (Pi,Pj)))

Pseudo Code

```
maximumPASToPASSimilarity(semanticsimilaritymatrix):
  matrix = np.matrix(semanticsimilaritymatrix)
  sum = matrix.sum(axis=1)
  return   sum.max()
p_F3(predicateindex, semanticsimilaritymatrix):
  matrix = np.matrix(semanticsimilaritymatrix)
  sum = matrix.sum(axis=1)
  return sum[predicateindex][0,0]/maximumPASToPASSimilarity(semanticsimilaritymatrix)
```

4.5.1.4. Position of Predicate argument Structure

PAS position indicates the prominence of the PAS in the text document and corresponds to the position of the phrase from which PAS is extracted. Consider the following ten sentences in the

document: the first sentence receives a score of 10/10, the second sentence receives a score of 9/10, and so on. This feature's score is calculated as follows:

P_F4 = Length of document - PAS Position + 1 / Length of PAS

Pseudo Code is Below

```
p_F4(predicate,document):
  return (document.lengthOfDocument() - predicate.getPosition() + 1)/ document.lengthOfDocument()
```

4.5.1.5. Proper Nouns

More proper nouns in the predicate argument structure are regarded as relevant for inclusion in summary generation. Proper nouns are defined as words that begin with a capital letter. The ratio of the number of proper nouns in the PAS to the length of the PAS is used to calculate the score for this attribute The number of words/terms in a PAS determines its length.

P_F5 = Number of proper nouns in the PAS / Length of PAS

Pseudocode of p_F5

```
p_F5(predicate):
  return predicate.numberOfProperNouns()/predicate.lengthOfPredicate()
```

4.5.1.6. Numerical Data

The predicate argument structure, which contains numerical data such as the number of people killed, is thought to be significant for summary production. The number of numerical data in the PAS multiplied by the length of the PAS yields the score for this characteristic.

p_F6 = Number of Numerical Data in the PAS / Length of PAS

```
p_F6(predicate):
  #return len(predicate.getNumericalData())
  return len(predicate.getNumericalData())/predicate.lengthOfPredicate()
```

4.5.1.7. Number of nouns and verbs

Some phrases may be related with more than one predicate argument structure, which is represented by a composite predicate argument structure and is important for summary. This feature's score is calculated as follows:

p_F7 = Total number of nouns and verbs in the PAS / Length of the PAS

```
p_F7(predicate):
  nounsLen= len(predicate.getNouns())
  if predicate.getVerb() == "":
   verbsLen= 0
  else:
   verbsLen= 1
  totalNounsAndVerbs= nounsLen + verbsLen
 # return totalNounsAndVerbs
  return totalNounsAndVerbs/predicate.lengthOfPredicate()
```

4.5.1.8. Temporal Feature

For the purpose of creating a summary, the predicates argument structure that includes the time and date of an event is important. The ratio of the number of temporal information (time and date) in the PAS over the length of the PAS is used to calculate this feature's score.

p_F8 = Number of temporal information in the PAS / Length of PAS

```
p_F8(predicate,document):
  str= "I reached office at 9 pm and 10 pm"
  # jar_files = os.path.join(os.path.dirname(__file__), 'jars')
  # sutime = SUTime(jars=jar_files, mark_time_ranges=True)
  return 0
```

4.5.1.9. Frequent Semantic Run

In this study, we take the top 10 frequent semantic phrases as the maximum number. Frequent terms are most likely relevant to the theme of the document. In the predicate argument structure, nouns and verbs are regarded as frequent semantic concepts. The ratio of the number of frequent semantic terms in the PAS over the maximum number of frequent semantic terms is used to determine this feature's score.

p_F9 = Number of frequent semantic terms in the PAS / MAX (Number of frequent semantic terms)

```
p_F9(predicate,document):
  nouns= len(predicate.getNouns())
  verb = len(predicate.getVerb())
  semanticTerms = nouns + verb
  maxsemanticTerms = 0
  for x in document.getPredicates():
    nouns= len(predicate.getNouns())
    verb = len(predicate.getVerb())
    sum = nouns + verb
    if sum > maxsemanticTerms:
      maxsemanticTerms = sum
  return semanticTerms / maxsemanticTerms
```

4.5.1.10. Semantic term weight

The score of important terms Wi can be determined by the TF-IDF method.We use the TF-IDF approach to analyse the predicate argument structures in the document collection and take into account the word weights for semantic terms like nouns and verbs.

The weight of semantic term is calculated as follows:

$$W_i = Tf_i \times Idf_i = Tf_i \times \log N/n_i \quad \text{.....(19)}$$

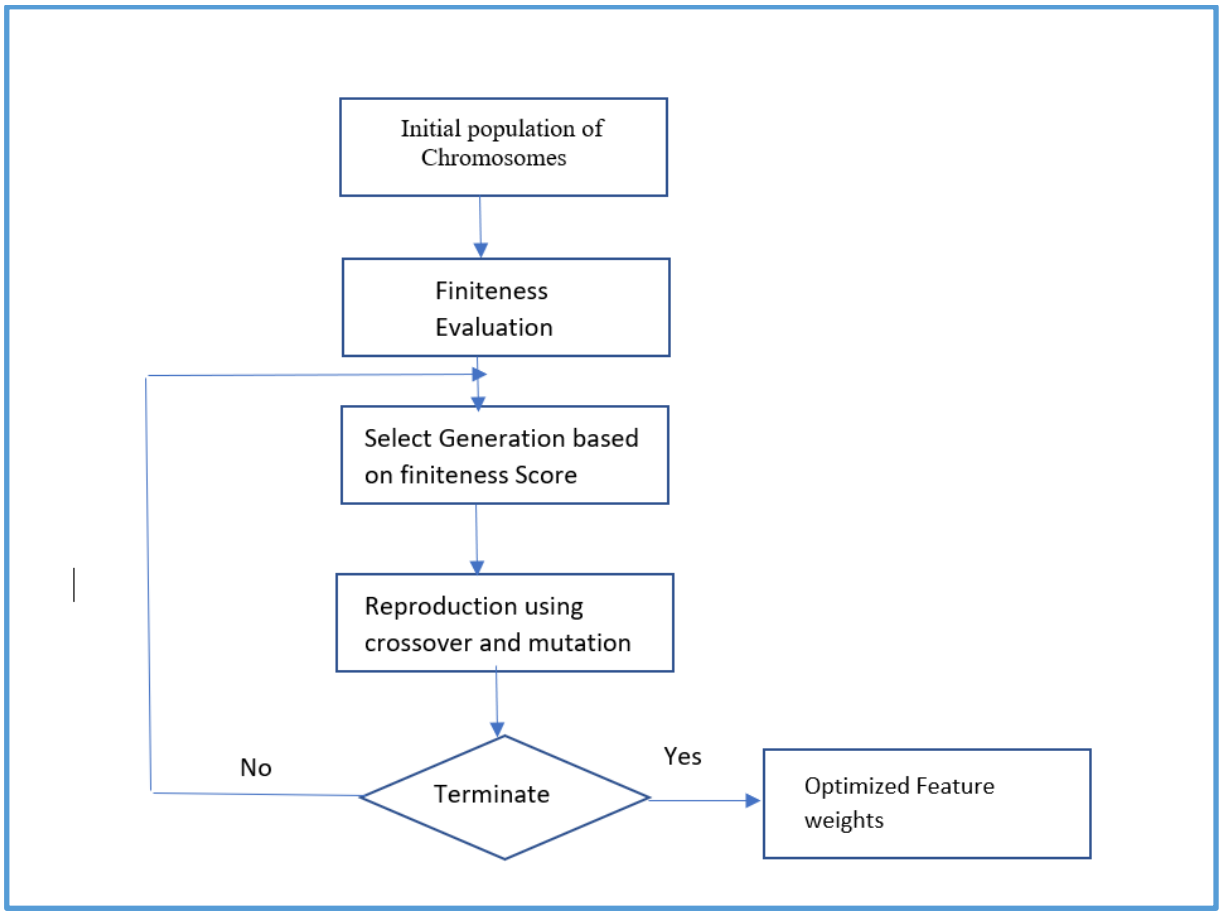
where Tf_i is the term frequency of the semantic term i in the document, N is the total number of documents, and n_i is the number of documents in which the term i occurs. This feature is computed as the ratio of sum of weights of all semantic terms in the PAS over the maximum summary of the term weights of PAS in the document collection.

p_F10 = Sum (Weight of semantic term for all the semantic terms in the PAS) / Max (Sum of weight of semantic terms of PAS in the document collection)

```
tFi(term, documentContent):
   return documentContent.count(term)

numberOfdocuments(documentCollection, term):
  documentsCount= 0
  for d in documentCollection:
    if tFi(term,d) > 0:
      documentsCount = documentsCount + 1
  return documentsCount


termWeight(term, documentContent, documentCollection):
  return tFi(term, documentContent) *
np.math.log(documentCollection.totalNumberOfDocument()/numberOfdocuments(documentCollection,ter
m))

p_F10(predicate, documentContent, documentCollection):
  nouns= predicate.getNouns()
  verb = predicate.getVerb()
  #semanticTerms = nouns + verb
  pfscore = []
  sumOfTFi = 0
  for s in semanticTerms:
    sumOfTFi = sumOfTFi + termWeight(s, documentContent, documentCollection)
    pfscore.append(s,sum)
  return pfscore
```

4.5.2. Genetic algorithm for optimal feature weighting

The quality of the summary in automatic text summarization depends on the text features; not all features are equally relevant in terms of the summary. As a result, feature weighting is essential for creating summaries. The genetic learning algorithm (GA) will be used in this study to determine the best feature weights for ranking predicate argument structures for summary generation. The reason GA was selected is that it is a reliable and well-known adaptive optimization technique that is employed in a variety of research and application areas [37]. The DUC 2002 dataset is used to evaluate this particular GA-based experiment against many texts. The Algorithm procedure is defined as below.



4.5.2.1 **Initial population of chromosomes.**

A global population of 50 chromosomes is formed at random in the first step of GA. The initial values for the chromosomes are chosen at random and range from 0 to 1. Each chromosome is a representation of the weights of traits in the form of (W_1, W_2, . . ., W_{10}). We generated a population of N chromosomes at random

Population = { P_{10} P_{11} P_{12} P_{13} P_{14} P_{15} P_{16} P_{17} P_{18} P_{19}

…….

P_{n0} P_{n1} P_{n2} P_{n3} P_{n4} p_{n5} P_{n6} P_{n7} P_{n8} P_{n9} }

where each row represents the chromosome, and each column value corresponds to the weight of each feature.

4.5.2.2 Fitness Evaluation

The fitness evaluation function in GA determines the best chromosome in the population based on its fit-ness rating. The chromosome having higher fitness value has more chances to survive and continue in the following generation. When the summarization procedure is used on many documents as described in Eq21We define fitness function F(x) as the average recall acquired with each chromosome.

$$F(x) = \frac{\sum_{S \in \{Reference\ Summaries\}} \sum_{gram_n \in S} Count_{match}(gram_n)}{\sum_{S \in \{Reference\ Summaries\}} \sum_{gram_n \in S} Count(gram_n)}$$

where n is the length of the n-gram, Count match is the number of n-grams shared between system generated summary and reference summaries, and Count is the total number of n-grams in the reference summaries

4.5.2.3 Selection

The chromosome that will survive in the following generation is determined by this procedure. Parents have been chosen from the present population using the stochastic universal sampling (SUS) method . This technique makes use of M evenly spaced steps in the range [0,Sum], where M is the number of required selections and Sum is the total of the scaled fitness values across all of the individual chromosomes in the current population.

4.5.2.4 Reproduction Operation

These processes outline how GA develops its newest generation. Crossover and mutation are the two reproduction processes we use in this study. By transferring information within the parent chromosomes, the crossover operation creates new chromosomes from the two parent chromosomes. To create a new genetic structure, a mutation operation involves changing a chromosome's gene by another gene.

4.5.2.5 Termination Criteria

Until the maximum number of generations is reached, this mechanism will continue to produce new generations and assess their fitness. We employed a maximum of 100 gener-ations in this study. The individual chromosome with the highest fitness value is selected as the ideal feature weights once the process is complete. In Section 3.2, the obtained ideal feature weights are provided. Eq. (22) is used to calculate the score of predicate argument structures in each cluster after the feature weights have been calculated.

$$Score(P_i) = \sum_{k=1}^{10} W_k x P\_F_k(P_i)$$

where $P\_F_k(P_i)$ is the score of feature k for predicate P_i, which is the weight of feature k. The top scored predicate argument structures are selected from each cluster and are given as input to the summary generation process in the next phase.

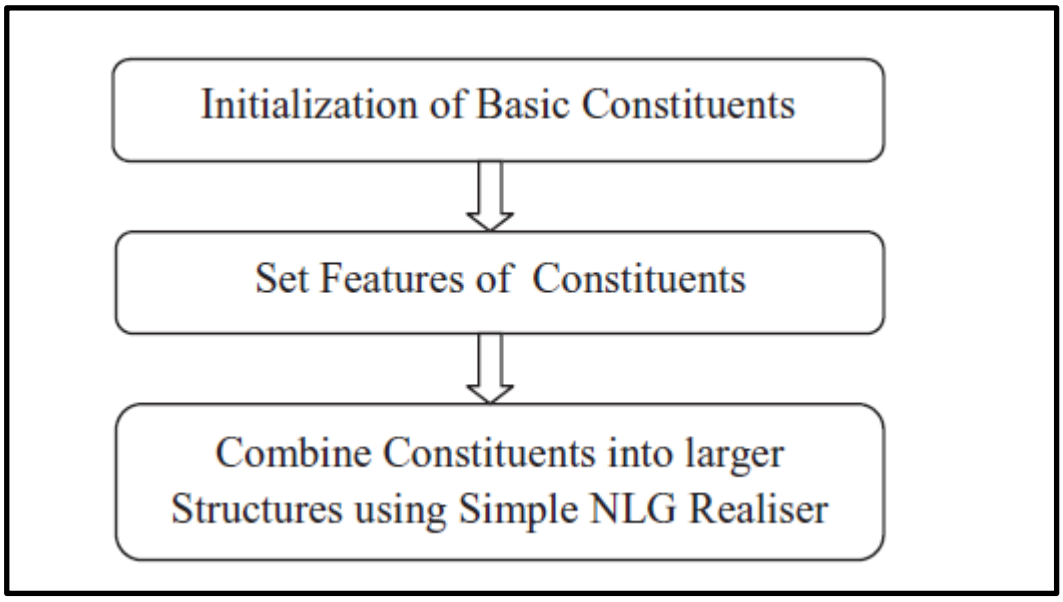
Chapter 4.6.: Abstractive Summary Generation

This stage shows how the predicates' arguments will be merged, modified, and realised as summary sentences. This phase uses the highest scoring predicate argument structures (PASs) from the previous phase and generates summary sentences using SimpleNLG and a straightforward heuristic method.

According to the heuristic rule, if two or more predicate argument structures (PASs) have subjects that refer to the same thing, they should be combined by removing the subject from all but the first PAS, separating them with a comma (if there are more than two PASs), and using the conjunction "and."

An English realisation engine called SimpleNLG offers straightforward interfaces for creating syntactical structures and turning them into whole sentences using basic grammar rules. The realisation engine offers customers an API that aims to give them programming control over the realisation process while also providing strong coverage of English syntax and morphology.

The engine's robustness, which prevents it from crashing when input syntactic structures are missing or poorly constructed, is another key benefit. Typically, this engine will provide the wrong result. The three key steps of the SimpleNLG engine's process are shown in the below figure



The fundamental syntactic components that make up a phrase must be defined as the first step in the SimpleNLG engine. The constituents are given features in the second phase. Subject, object verb, tense, and other properties can all be set for components. The syntactic components are combined

with simple grammar rules in the final phase of the SimpleNLG realiser to create a syntactically correct sentence.

As mentioned earlier, we only take into account a few arguments, namely A0 for subject, A1 for object, and A2 for indirect object, as core arguments for predicate (Verb) V, and ArgM-LOC for location and ArgM-TMP for time as adjectival arguments. The rest of the arguments are disregarded. The final summary sentences that are generated from the predicate argument structures will therefore, in most instances, be compressed versions of the original source phrases. The simple heuristic rule implemented in SimpleNLG combines the predicate argument structures that relate to the same subject throughout the summary sentence generation process (entity). The following example shows how to create an abstractive summary using the source input sentences that are provided. Examples of source input include the following:

S1: Hurricane Gilbert claimed to be the most intense storm on record in terms of barometric pressure.
S2: Hurricane Gilbert slammed into Kingston on Monday with torrential rains and 115 mph winds.
S3: Hurricane Gilbert ripped roofs off homes and buildings.
After applying SENNA SRL, the corresponding three predicate argument structures P1, P2 and P3 are obtained as follows:

P1: [A0: Hurricane Gilbert] [V: claimed] [A1: to be the most intense storm on record]

P2: [A0: Hurricane Gilbert] [V: slammed] [A1: into Kingston] [AM-TMP: on Monday]

P3: [A0: Hurricane Gilbert] [V: ripped] [A1:roofs off homes and buildings]

The top-scoring predicate argument structures chosen from various clusters in the preceding phase are assumed to be P1, P2, and P3. The subject A0 is detected as repeated in the example above in accordance with the aforementioned rule, and all predicate argument structures except the first one remove it. Applying the heuristic rule to the three predicate argument structures mentioned above, the SimpleNLG creates a summary sentence that is a condensed version of the original source sentences.

The stepwise details of SimpleNLG Code as below.

Step 1. Simple NLG comes with a built in lexicon that contain information (part of speech) about words, and accessed via:

Lexicon lexicon = Lexicon.getDefaultLexicon();

Step 2. Once we get a lexicon, we can create an NLGFactoryobject to create SimpleNLG structures.

NLGFactory nlgFactory = new NLGFactory(lexicon);

Step 3. This step creates a realiser object that will trans-form/combine SimpleNLG structures (constituents of sentence)into text.

Réaliser realiser = new Réaliser(lexicon);

Step 4: This step creates a simple NLG construct SPhraseSpec by calling a method of nlgFactory object, which allows us to define the syntactic constituents that will form a sentence.

SPhraseSpec p = nlgFactory.createClause();

Step 5: This step defines the constituent subject along with the specification of subject feature.

p.setSubject(subject);

Step 6: This step defines the constituent verb along with the specification of verb feature.p.setVerb(verb);

Step 7: This step defines the constituent object along with the specified object feature.p.setObject(object);

Step 8: This step defines the constituent indirect object along with the specified indirect object feature.p.setIndirectObject(indirectObject);

Step 9: This step uses realiser object created in step3, which takes different constituents stored in object p obtained in steps 5,6,7 and 8, assemble them using simple grammar rules embodied in simple NLG, and finally the sentence is realized to make the result syntactically and morphologically correct:

String output = realiser.realiseSentence(p);

The output variable will contain the generated sentence.Once the summary sentences are generated from predicate argument structures, they are rearranged based on the position of predicate argument structures. The position of predicate argument structure (PAS) is determined from the position of source sentence in the document collection, from which this PAS is extracted. In case, the summary sentence is generated from more than one predicate argument structure then its probable position in the summary is determined based on the position of the first predicate argument structure.

```
realiseSentence(self, subject, verb, object):
    l = lexicon.Lexicon.getDefaultLexicon()
    nlg = framework.NLGFactory(l)
    r = english.Realiser(l)
    p = nlg.createClause()
    p.setSubject(subject)
    p.setVerb(verb)
    p.setObject(object)
    return r.realiseSentence(p)

 nlg = SimpleNLG()
print(nlg.realiseSentence("Hurricane Gilbert", "claimed", "to be the most intense storm on record"))
```

Chapter 5: Result and Discussion

This section demonstrates how our suggested framework for abstractive summarization was evaluated. The DUC 2002 / DUC 2004 dataset is used to evaluate the proposed framework . The Document Understanding Conference (DUC), a commonly used corpus in text summarising research, contains documents and the summaries created by human models for those documents. The National Institute of Standards and Technology created 59 document sets for the DUC 2002 data set (NIST). Each document collection includes documents, single document abstracts, and multi-document extracts with sets determined by various sorts of criteria, including event sets and biographical sets. In

DUC-2002, three tasks were assessed: fully automatic summarization of a single newswire or newspaper document, fully automatic summarization of numerous newswire or newspaper documents on a single topic by generating document extracts, and fully automatic summarization of numerous newswire or newspaper documents on a single topic by generating document abstracts. DUC 2002 is the best appropriate dataset for our research because it corresponds to task 3 (multi-document abstractive summarization on a single subject), which was only defined for DUC 2002. The remaining DUC data sets, on the other hand, deal with additional summarization tasks, like question-answering and update summarization. The document collection is initially divided into sentences, with the corresponding document number and sentence position number placed before each sentence. Each sentence in the document collection has its predicate argument structure extracted using the SENNA semantic role labeller. The similarities of predicate argument structures (PASs) are computed by comparing them pair-wise based on Jiang's semantic similarity measure and edit distance algorithm in the semantic similarity matrix computation phase. After obtaining the similarity matrix for PASs, we use the aforementioned agglomerative hierarchical clustering approach to group semantically related predicate argument structures. We find feature weights that are optimised by running the genetic algorithm in order to distinguish between significant and less-significant features (GA). Using 10-fold cross validation, GA is trained and tested on 59 multi-documents (obtained from DUC 2002). 50 chromosomes were randomly initialised with real values between 0 and 1 to create the first population. The average recall acquired with each chromosome after applying the summarization algorithm to the training corpus is the fitness function. Each chromosome's quality is assessed by the fitness function, which establishes its fitness value. The best chromosomes from the present population are chosen as parents for the following generation based on fit-ness rating. Then, in each generation, selected chromosomes are reproduced through crossover and mutation procedures. We execute a maximum of 100 generations before stopping the GA process to allow the fitness value to converge. The ideal feature weights are determined by selecting the particular chromosome with the highest fitness value. The top-ranked predicate argument structure from each cluster is then scored and chosen using the optimised features weights. For the purpose of creating summary sentences, the chosen PASs are put into the SimpleNLG realisation engine.

Recall-Oriented-Understudy for Gisting Evaluation (ROUGE) and Pyramid are two commonly utilised evaluation metrics that have been employed in the evaluation of text summaries. Previous studies revealed that the ROUGE measure was used to assess extractive summaries. The ROUGE score measures the exact n-gram matches between system and human model summaries. Semantically comparable sentences cannot be measured using this metric. In contrast, abstractive summaries are evaluated using the Pyramid metric. Pyramid metric clearly has an edge over ROUGE in that it can identify different sentences in summaries that employ various words but convey similar ideas . We use the Pyramid evaluation metric to assess our suggested framework because our study focuses on multi-document abstract summarization. This metric measures the quality of a system generated summary by comparing it with human model summaries.

Pyramid score (Mean Coverage Score) for peer summary of candidate summary is computed as follows.

Mean Coverage Score = Total Peer SCUs Weight / Average SCU in the Model Summary

where SCUs refers to the summary content units and their weights correspond to the number of model (human) summaries they appeared in.The precision for peer summary or candidate summary is computed as follows.

Precision = Number of Model SCUs expressed in Peer Summary / Average SCU in the Peer Summary

Results

The proposed framework is evaluated in the context of multidocument abstractive summarization task, using 50 news articles/datasets provided by the Document Understanding Evaluations 2004 (DUC, 2004). For each data set, our framework generates a 12-sentence summary, the task undertaken by other systems participating in a multi-document abstractive summarization task.

Method	Rouge-1	Rouge-2	Rouge-L
P.R. +Δ	31.7	5.56	10.1
Lead+Δ	31.8	5.74	10
Lex.+Δ	32.9	6.28	10.8
Text.+Δ	33.3	6.1	10.7
Cent.+Δ	34.4	6.68	11.1
Δ+Lead	33.2	6.12	10.6
Δ+Cov.	34.4	6.84	11.2
Δ+Lex.	34	6.3	11
Δ+Text.	34.3	6.71	11.1
Δ+Cent.	32.8	5.77	10.3
Δ+Δ 31.	34.7	9.6	52
Our Model	**37.7**	**8.8**	**35.3**

TABLE : Comparison results with abstractive baselines on the DUC 2004 test set

Human Evaluation

For human review, we randomly select 10 document sets from the DUC 2002 dataset and an additional 10 document sets from the DUC 2004 dataset. Coherence, Non-Redundancy (N.R. for short), and Readability manual assessments were requested from three volunteers who are fluent in English. The ratings are presented as numerical values between 1 and 5, which are not always integral. Higher scores indicate higher quality. It can be seen that our method achieves good coherence and readability, outperforming existing abstractive summarization approaches in human review.

Method	Coherence	N.R.	Readability
Lead+Δ	2.32	2.74	2.71
Cent.+Δ	2.63	2.84	3.29
Δ+Cov.	2.3	3.53	2.92
Δ+Text.	3.18	3.75	3.34
Δ+Δ	2.23	2.57	2.57
Our Model	**3.82**	**3.92**	**4.12**

TABLE : Human evaluation results on 20 samples from the DUC 2002 and DUC 2004 datasets.

Chapter 6: Conclusion

The problem of Abstractive Multi-Document Summarization (MDS) is still difficult and unsolved. Despite the fact that sequence-to-sequence models have made considerable strides in single document summarization, applying them to the MDS problem is challenging due to the high training data requirements.

Fully abstractive summarising is a difficult task, but our suggested framework demonstrates the viability of this new line of inquiry in summarization research. The proposed effort, in our opinion, tries to regulate the content and structure of the summary, which are the true objectives of automatic summarising. According to experimental findings, the suggested framework performs better than other comparison models and trails only the average of human model summaries. Furthermore, we used GA in our proposed framework to assign the proper weights to each feature in order to assess its importance, and we used the optimised feature weights to choose the highest ranked predicate argument structures from each cluster for the summary generation phase. In comparison to the SRL-based framework without GA, the results show that including GA into the suggested framework enhances summarization outcomes. In future, we plan to integrate graphs with SRLto build a semantic graph for multi-document abstractive summarization.

Chapter 7 : References

[1] R. Barzilay, K.R. McKeown, Sentence fusion for multi document news summa-rization, Comput. Linguist. 31 (2005) 297–328.

[2] J. Kupiec, J. Pedersen, F. Chen, A trainable document summarizer, in: Proceed-ings of the 18th Annual International ACM SIGIR Conference on Research andDevelopment in Information Retrieval, 1995, pp. 68–73.

[3] K. Knight, D. Marcu, Statistics-based summarization-step one: sentence com-pression, in: Proceedings of the National Conference on Artificial Intelligence,2000, pp. 703–710.

[4] B. Larsen, A trainable summarizer with knowledge acquired from robust NLPtechniques, in: Advances in Automatic Text Summarization, MIT Press, 1999,pp. 71.

[5] H.P. Luhn, The automatic creation of literature abstracts, IBM J. Res. Dev. 2(1958) 159–165.

[6] P.-E. Genest, G. Lapalme, Framework for abstractive summarization using text-to-text generation, in: Proceedings of the Workshop on Monolingual Text-To-Text Generation, 2011, pp. 64–73.

[7] I. Titov, A. Klementiev, A Bayesian approach to unsupervised semantic role induction, in: Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics, 2012, pp. 12–22.

[8] R. Barzilay, K.R. McKeown, M. Elhadad, Information fusion in the context ofmulti-document summarization, in: Proceedings of the 37th Annual Meetingof the Association for Computational Linguistics on Computational Linguistics,1999, pp. 550–557.

[9] H. Tanaka, A. Kinoshita, T. Kobayakawa, T. Kumano, N. Kato, Syntax-driven sentence revision for broadcast news summarization, in: Proceedings of the 2009 Workshop on Language Generation and Summarisation, 2009,pp. 39–47.

[10] S.M. Harabagiu, F. Lacatusu, Generating single and multi-document summaries with GISTEXTER, in: Document Understanding Conferences, 2002.

[11] C.-S. Lee, Z.-W. Jian, L.-K. Huang, A fuzzy ontology and its application tonews summarization, IEEE Trans. Syst. Man Cybern. B: Cybern. 35 (2005)859–880.

[12] P.-E. Genest, G. Lapalme, Fully abstractive approach to guided summarization,in: Proceedings of the 50th Annual Meeting of the Association for Computa-tional Linguistics: Short Papers – vol. 2, 2012, pp. 354–358.

[13] C.F. Greenbacker, Towards a framework for abstractive summarization of mul-timodal documents, in: ACL HLT 2011, 2011, p. 75.

[14] I.F. Moawad, M. Aref, Semantic graph reduction approach for abstractive text summarization, in: 2012 Seventh International Conference on Computer Engi-neering & Systems (ICCES), 2012, pp. 132–138.

[15] S. Shehata, F. Karray, M.S. Kamel, An efficient concept-based retrieval model for enhancing text retrieval quality, Knowl. Inf. Syst. 35 (2013)411–434.

[16] L. Del Corro, R. Gemulla, ClausIE: clause-based open information extraction, in:Proceedings of the 22nd International Conference on World Wide Web, 2013,pp. 355–366.

[17] J. Persson, R. Johansson, P. Nugues, Text Categorization UsingPredicate–Argument Structures, vol. 1, 2008, pp. 142–149.

[18] N. Jadhav, P. Bhattacharyya, Dive deeper: deep semantics for sentiment analy-sis, ACL 2014 (2014) 113.

[19] N. Salim, SRL–GSM: a hybrid approach based on semantic role labeling and general statistical method for text summarization, J. Appl. Sci. 10 (2010) 166–173.

[20] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, P. Kuksa, Natu-ral language processing (almost) from scratch, J. Mach. Learn. Res. 12 (2011)2493–2537.

[21] J.J. Jiang, D.W. Conrath, Semantic Similarity Based on Corpus Statistics andLexical Taxonomy, 1997 arxiv:cmp-lg/9709008.

[22] A. Gatt, E. Reiter, SimpleNLG: a realisation engine for practical applications, in:Proceedings of the 12th European Workshop on Natural Language Generation,2009, pp. 90–93.

[23] C. Aksoy, A. Bugdayci, T. Gur, I. Uysal, F. Can, Semantic argument frequency-based multi-document summarization, in: 24th International Sym-posium on Computer and Information Sciences 2009 (ISCIS 2009), 2009,pp. 460–464.

[24] M.F. Porter, Snowball: A Language for Stemming Algorithms, 2001.

[25] Y. Li, Z.A. Bandar, D. McLean, An approach for measuring semantic similaritybetween words using multiple information sources, IEEE Trans. Knowl. DataEng. 15 (2003) 871–882.

[26] G.A. Miller, WordNet: a lexical database for English, Commun. ACM 38 (1995)39–41.

[27] P. Achananuparp, X. Hu, C.C. Yang, Addressing the variability of natural language expression in sentence similarity with semantic structure of the sen-tences, in: Advances in Knowledge Discovery and Data Mining, Springer, 2009,pp. 548–555.

[28] F. Murtagh, V. Contreras, Methods of Hierarchical Clustering, 2011,arxiv:1105.0121.

[29] S. Takumi, S. Miyamoto, Top-down vs bottom-up methods of linkage for asymmetric agglomerative hierarchical clustering, in: 2012 IEEE InternationalConference on Granular Computing (GrC), 2012, pp. 459–464.

[30] M. Steinbach, G. Karypis, V. Kumar, A comparison of document clustering tech-niques, in: KDD Workshop on Text Mining, 2000, pp. 525–526.

[31] Y. Zhao, G. Karypis, U. Fayyad, Hierarchical clustering algorithms for document datasets, Data Mining Knowl. Discov. 10 (2005) 141–168.

[32] A. El-Hamdouchi, P. Willett, Comparison of hierarchical agglomerative clustering methods for document retrieval, Comput. J. 32 (1989) 220–227.

[33] L. Suanmali, N. Salim, M.S. Binwahlan, Fuzzy Logic Based Method for ImprovingText Summarization, 2009, arxiv:0906.4690.

[34] M.A. Fattah, F. Ren, GA, MR, FFNN, PNN and GMM based models for automatic text summarization, Comput. Speech Lang. 23 (2009) 126–144.

[35] J.-M. Lim, I.-S. Kang, J. Bae, J.-H. Lee, Sentence extraction using time fea-tures in multi-document summarization, in: Information Retrieval Technology,Springer, 2005, pp. 82–93.

[36] G. Salton, Automatic Text Processing: The Transformation, Analysis, andRetrieval of Information by Computer, Addison-Wesley, 1989.

[37] M. Srinivas, L.M. Patnaik, Genetic algorithms: a survey, Computer 27 (1994)17–26.

[38] DUC, Document Understanding Conference 2002, 2002.

[39] J.E. Baker, Reducing bias and inefficiency in the selection algorithm, in: Proceedings of the Second International Conference on Genetic Algorithms, 1987,pp. 14–21.

[40] C.-Y. Lin, Rouge: a package for automatic evaluation of summaries, in: TextSummarization Branches Out: Proceedings of the ACL-04 Workshop, 2004, pp.74–81.

[41] A. Nenkova, R. Passonneau, Evaluating Content Selection in Summarization:The Pyramid Method, 2004.