**International Journal of Web-Based Learning and Teaching Technologies**
Volume 16 • Issue 5 • September-October 2021

# Summarizing Learning Materials Using Graph Based Multi-Document Summarization

Krishnaveni P, National Institute of Technology, Tiruchirappalli, India
Balasundaram S R, National Institute of Technology, Tiruchirappalli, India

## ABSTRACT

The learners and teachers of the teaching-learning process highly depend on online learning systems such as E-learning, which contains huge volumes of electronic contents related to a course. The multi-document summarization (MDS) is useful for summarizing such electronic contents. This article applies the task of MDS in an E-learning context. The objective of this article is threefold: 1) design a generic graph based multi-document summarizer DSGA (Dynamic Summary Generation Algorithm) to produce a variable length (dynamic) summary of academic text based learning materials based on a learner's request; 2) analyze the summary generation process; 3) perform content-based and task-based evaluations on the generated summary. The experimental results show that the DSGA summarizer performs better than the graph-based summarizers LexRank (LR) and Aggregate Similarity (AS). From the task-based evaluation, it is observed that the generated summary helps the learners to understand and comprehend the materials easily.

## KEYWORDS

Dynamic Summary Generation, E-Learning, Learning Materials Summarization, Maximal Cliques, Online Learning Systems, Task-Based Evaluation, Teaching-Learning Process, Variable Length Summary

## INTRODUCTION

To access relevant information quickly in today's vast amount of online information, the automatic text summarization (ATS) is an important and timely too1. The ATS produces a summary text from the given input text whose size is less than half of the original text and contains important information (Radev et al., 2002). The ATS process can be either single document summarization (SDS) or multi-document summarization (MDS). The single document summarization generates the summary of a single document, whereas the multi-document summarization generates the summary of a group of related or unrelated documents. The multi-document summary should contain the relevant information shared among all the documents, plus the unique information about some of the documents which are essential (Goldstein et al., 2000).

This article generates a summary of learning materials using the MDS with sentence similarity graphs. It uses the graph structure, maximal clique to provide a concept oriented summary *(*Tomita et al., 2011). A clique is a complete sub graph of a graph. Since all nodes (sentences) in a clique are related to each other, each clique represents one concept or main idea of the given text. A maximal clique is a clique which is not a proper subset of any other clique. This article covers all important concepts of the given text based learning materials by selecting summary sentences from a diverse

DOI: 10.4018/IJWLTT.20210901.oa3

set of maximal cliques of the sentence graphs of the given input text. Throughout this article clique means maximal clique.

The same teacher teaches for all the students in the class at the same time in traditional classroom teaching. But, different students have different learning capacity due to the knowledge difference among the students (Wang & Cai, 2009). Some students may not have an interest in getting more details of a topic. Some may not understand if the content is too long. Therefore, the E-learning is used nowadays for improving the learning interests and efficiency of the learners.

In E-learning, the learning material is delivered to the remote learners through a computer network. The learning object of an E-learning environment is a chunk of electronic content that can be accessed individually. Since each learning resource contains a vast amount of information to be read, the learners feel difficult when they are reading at their earlier stage. About 60% of students wanted the summarized material rather than the entire content (Shimada et al., 2015). Hence, providing a summary for either a single learning material or a group of learning materials on a specific topic can help the learners understand the resource better. This is the reason why does this article aim to summarize the learning materials.

This article aims to provide a dynamic multi-document summary of academic learning materials especially computer science theory subjects in the form of textual documents from various sources such as an E-learning resource, lecture notes, a learner's class notes, previous year notes, book chapters, and some other online documents based on a learner's requests. Why does this article aim to provide dynamic summaries? To satisfy the varying summary requirements of different kinds of learners, dynamic summaries are provided. A learner can decide the summary size up to fifty percent of the length of the document-set (set of input documents). Once get a summary, if the learner is not satisfied, he may go for further levels of summaries. This summary can be used as a preview before reading the text first time and can be useful during revision (Baralis & Cagliero, 2016). Also, it can be used as a summarized learning material to learners with fewer skills, students studying in part-time mode, and students studying in distance education.

The structure of this work is framed as follows. The Background section discusses the related works on MDS, graph based MDS, and the E-learning context and compares them with the proposed work. The Proposed work section describes the problem statement, overview, feature extraction, sentence score calculation, summary generation, and summary evaluation of the proposed work. The Evaluation results and discussion section shows the results of the intrinsic content based evaluation and the extrinsic task based evaluation and discusses these results. The Conclusion section concludes the current work and suggests the future work. The reference section lists the references cited in this work.

## BACKGROUND

This section discusses various earlier works, graph based, algorithmic, and E-learning works of MDS. The earlier approaches for MDS are statistical, linguistic, and feature based (Ferreira et al., 2014), centroid based (Rossiello et al., 2017), clustering (Cai & Li, 2013; Fejer & Omar, 2015), machine learning (Cao et al., 2017), etc. The word or sentence specific statistical feature based approaches lack of semantic and group relations. In clustering, the sentences are tightly connected. Machine learning techniques require huge training corpus. This article utilizes group relations using the sentence similarity graphs of the input documents with maximal cliques.

Later, graph based ranking models become popular (Ramanujam & Kaliappan, 2016; Calvo et al., 2018; Feiyue & Xinchen, 2018). Such models use the sentence similarity graph to score and rank sentences. The iterative graph algorithms compute a score for each sentence based on centrality measures (Erkan & Radev, 2004; Mihalcea & Tarau, 2004). But, they can consume more processing power while processing a large amount of text. Generally, the graph based summarization approach lacks of semantics (kanitha et al., 2018). The semantic relatedness is very important for news domain, because there is a possibility of occurrence of different words with the same meaning. But in the

educational domain, the possibility of occurrence of semantically similar words is very less. Hence, the semantic relatedness is not so important for educational domain. This article combines the benefits of clustering and graph based techniques by providing a graph based summarization using maximal cliques. In addition, it includes two heuristics heading words and cue words.

Recently, the algorithmic approaches with graphs are used for generating multi-document summaries. The vertex cover problem is used in the work of John and Wilscy (2014) to select the sentences that cover the predominant concepts of the given input text. The multi-document summarization problem is formulated as an orienteering problem to optimize the coverage of information in the output summary in (Al-Saleh & Menai, 2018). Generally, the algorithmic approaches select a sentence which is related to a group of sentences as summary sentence, but the sentences in this group may or may not be related to each other. But, this article selects summary sentences from distinct groups of highly interconnected sentences using the concept based summary generation (CBSG) algorithm, which is based on maximal cliques. The CBSG algorithm takes the sentence similarity graph and the sentence score of all sentences in this graph as input and produces a summary list as output. It utilizes maximal cliques to produce the summary list. Every time, it takes the sentence with the highest score from the largest size maximal clique as summary sentence. In this article, the CBSG is used to generate local summary list (LSL) and global summary list (GSL) of the given input documents. The LSL contains locally redundant or popular information. The LSL is generated for each document. The GSL contains globally redundant or popular information. The GSL is generated for the entire document-set.

Some research works have taken effort to integrate summarization algorithms into E-learning context (Wang & Cai 2009; Baralis & Cagliero, 2016 ). In the work of Shimada et al. (2015), lecture slides are summarized to enhance the learners understanding of the content prior to teaching. Normally, people with dyslexia have difficulty in understanding the text based learning documents. There is a need for an assistive summary to help the dyslexic people to understand the learning material with low difficulty (Nandhini & Balasundaram, 2013). It produces a user focused summary. But this article produces generic summaries for all kinds of learners. The above two works on E-learning context are done in a single document. This article provides a dynamic summary of multiple learning materials on a common topic using a graph based MDS system. In the work of Cagliero et al. (2019), a personalized summary is provided to learners based on the performance of the test conducted at the end of the lecture on a class. A learner's mentality varies time to time. So providing a summary to learner's needs according to the result of a comprehension test performed at the end of the lecture on a class is not so good.

In this article, the learners decide the summary size according to their requirements. This article aims to provide a coherent dynamic multi-document summary of learning materials on a common topic that covers all important concepts of the source documents without redundancy and with novelty using the DSGA algorithm and the CBSG algorithm. The DSGA generates a summary of the given input documents using six different kinds of summary lists and summary length (SL). It improves the coherence of the summary through post-processing the summary. It provides novel or diverse information by including thirty percent of isolated sentences of the sentence graph of the document-set into the summary. Thus, the summary of this article improves the learner's knowledge and understanding of the given topic. Does the summary incorporate the main ideas (concepts or themes) of the given learning materials? The CBSG utilizes maximal cliques to extract the concepts of the given learning materials. How to provide summaries to fulfill the varying requirements of different kinds of learners? Dynamic summaries are provided to a learner based on the learner's request. How to evaluate the generated summary? This article performs intrinsic content based evaluation and extrinsic task based evaluation to evaluate the generated summary. What is the significance of this article? This article is useful in the situation where a learner has one or more text based learning materials on a specific topic and need a summary of those materials to understand the content quickly. Also, the learner needs variable size summaries up to 50% of the entire document-set length.

## PROPOSED WORK

### Problem Statement

The problem is to generate a dynamic summary of learning materials from various sources related to a common topic using a graph based summarizer DSGA based on a learner's request. The learners use this summary as an additional material for study and revision. The goal of generating a summary of learning materials is to improve the knowledge and understanding of the learners on the specified topic. With the help of such summaries, the learners can read more documents in a short period. By studying the summary of learning materials, the learners can memorize the contents easily and score more marks in examinations.

### Method Overview

First, get the learning materials related to a specific topic one by one, split each document into heading text and normal text and store them in separate lists. Combine the text of all documents into one document-set. Each sentence in the document-set is numbered in ascending order to form a sequence of natural numbers start at zero from the first sentence of the first document to the last sentence of the last document. The entire document-set length (DL) is the sum of the length of the individual documents. Then, preprocess the document-set using the Python package, natural language tool kit (NLTK). The basic preprocessing operations such as sentence tokenization, word tokenization, stop word removal, and word stemming perform on the entire document-set.

This work computes a global sentence similarity matrix of the document-set and a local sentence similarity matrix for each document using Cosine Similarity (Han & Kamber, 2006). The Cosine Similarity is used to measure the lexical similarity between two sentences. Each value $a_{ij}$ in the similarity matrix A is the lexical similarity between the pair of sentences i and j. It is a symmetric matrix, i.e., $a_{ij} = a_{ji}$. The Cosine Similarity value ranges from zero to one for every pair of sentences. It is close to one when both sentences are highly similar. It is close to zero when both sentences are highly differing. Then, it constructs a sentence similarity graph using the global similarity matrix. It is a global graph of the document-set. Also, it constructs a separate sentence similarity graph for each document using its local similarity matrix. It is a local graph. The Python package NetworkX is used for graph construction. The NetworkX utilizes the Bron-Kerbosch algorithm to find all maximal cliques of an undirected graph in linear time (Conte, 2013). A link is created between two sentences in the similarity graph only if their similarity value is greater than or equal to a particular threshold value. This work uses the average similarity value of the cosine similarity matrix as the threshold value. If there is a link between two sentences, then definitely they should share some common information.

Maximal cliques form the natural clusters of sentences in these sentence similarity graphs. Highest score sentences of the largest size maximal cliques are selected from the global graph to form the global summary list (GSL) of the document-set and are selected from the local graph to form the local summary list (LSL) of each document. The dynamic summary generation algorithm DSGA generates the dynamic summary of the user specified percentage using the CBSG algorithm and the various summary lists generated in this work. The architecture of the overall work is shown in Figure 1.

### Feature Extraction

The main task of ATS process is the salient sentence selection. Various features are used to identify salient sentences. This work uses the features Word frequency and Sentence length, because they provide the best balance in electing important sentences and in execution-time performance (Ferreira et al., 2013). Since, it aims to produce a summary of learning materials on a specific topic, Topic similarity would be calculated for each sentence. In a graph based summarization, a sentence links with many other sentences is very important (Mihalcea & Tarau; 2004; Erkan & Radev, 2004). So this work calculates the intra-link feature for the sentences of each document and the interlink feature for the sentences of the entire document-set. It uses two more features Sentence in number of Cliques

(SC) and Sentence Links with number of Documents (SLD) to include group and document side information.

## Sentence Score Calculation

It calculates local score and global score for each sentence using the related features discussed in the feature extraction section.

### *Global Score (GS)*

The global score GS of a sentence i is the sum of the scores of its features Word Frequency (WF), Sentence Length (SenLen), Topic Resemblance (TR), Interlink, Sentence in number of Cliques (SC), and Sentence Links with number of Documents (SLD). It is given in equation (1).

$$GS\left(i\right) = WF\left(i\right) + SenLen\left(i\right) + TR\left(i\right) + Interlink\left(i\right) + SC\left(i\right) + SLD\left(i\right) \quad (1)$$

### *Local Score (LS)*

The local score LS of a sentence i is the sum of the scores of its features Word Frequency (L_WF(i)), Sentence Length (L_SenLen(i)), Topic Resemblance (TR), Intra_link, and Sentence in number of Cliques in its own document.

$$LS\left(i\right) = L\_WF\left(i\right) + L\_SenLen\left(i\right) + TR\left(i\right) + Intra\_link\left(i\right) + L\_SC\left(i\right) \quad (2)$$

where $L\_WF\left(i\right)$ is the word frequency of sentence i in its own document, $L\_SLen\left(i\right)$ is the sentence length score of sentence i in its own document, and $L\_SC\left(i\right)$ is the number of cliques, the sentence i occurs in its own document graph (local graph).

## Summary Generation

The summary generation step generates output summary text from the given input documents using the DSGA algorithm. It has three major tasks of local summary list generation, global summary list generation, and final summary generation. The CBSG algorithm is used for local and global summary lists generation.

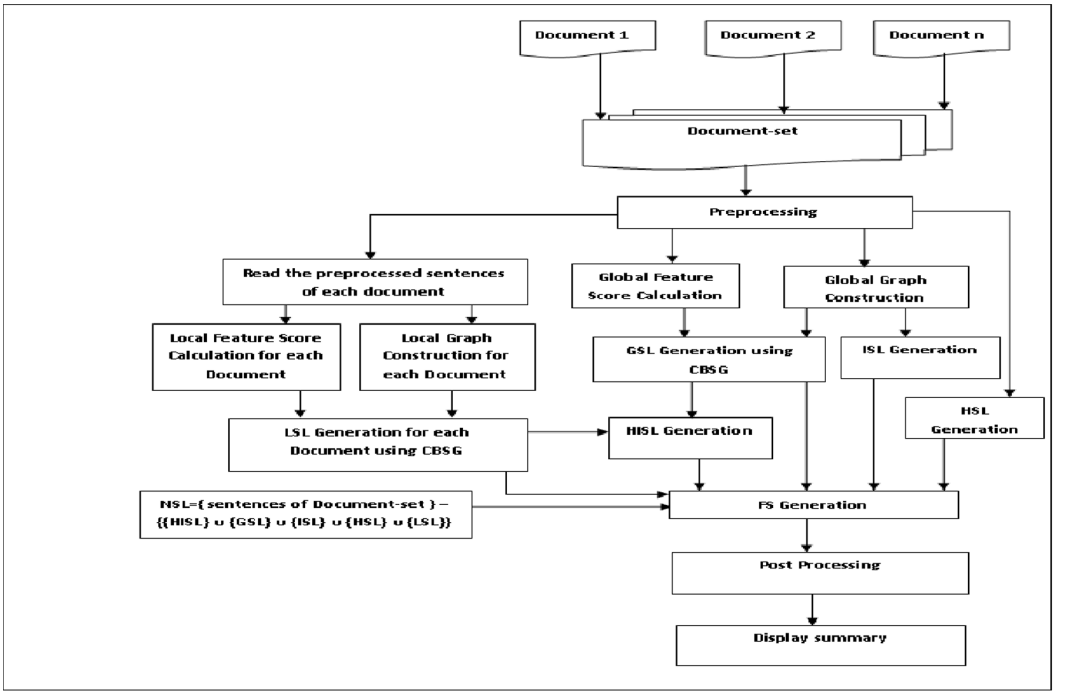
### *Local Summary List Generation*

Highest score (local score) sentences are extracted from the largest size maximal cliques of the local graph of each document using CBSG algorithm. These sentences are stored in the local summary list (LSL) document wise. The LSL contains locally redundant or popular information.

### *Global Summary List Generation*

Highest score sentences (global score) are extracted from the largest size maximal cliques of the global graph using CBSG algorithm. These sentences are stored in the global summary list (GSL). The GSL contains globally redundant or popular information. Also, it stores all cliques of size one which are called as isolated sentences of the global graph in the isolated summary list (ISL). Since the isolated sentences have no link with other sentences, they contain novel or diverse information.

**Figure 1. System architecture**




### *The CBSG Algorithm*

The algorithm CBSG is given in Algorithm 1. It selects the highest score sentences from the largest size maximal cliques, next largest size maximal cliques, and so on until the graph is empty. Since it selects at most one sentence either from an independent clique or a group of related cliques, these sentences are disjoint and have no redundancy.

**Algorithm 1:** Concept Based Summary Generation (CBSG) Algorithm
// Throughout the algorithm clique means maximal clique.
**Input:** Sentence similarity graph (local or global), Sentence score (LS or
GS)of all sentences in this graph.
**Output:** *LSL*, or *GSL* and *ISL*.
Find all the cliques of the given graph.
Remove all cliques of size one from the graph. Store it in *ISL* in case of global graph.
Store all cliques of size >=2 in *Allcliques*.
Find the maximum clique size of the graph.
List all cliques of maximum clique size.
Select all distinct sentences in these cliques and store them in a temporary list. Arrange the sentences in descending order of their sentence score.
Select the first sentence from the temporary list and store it into the *GSL* if the input graph is global or store it into the *LSL* if the input graph is local.
Remove the selected sentence and its adjacent sentences from the sentence graph.

```
Store all cliques of the modified graph in Allcliques.
If the length of Allcliques >0 go to step 4.
Display the summary list either LSL or GSL and ISL that covers all
concepts of the given input text.
```

### *Final Summary Generation*

The DSGA compares each sentence of the local summary list with the global summary list. If a sentence occurs in both lists, then it would be highly informative. So remove it from both lists and store it in the highly informative summary list (HISL). The heuristically important summary list (HSL) is obtained by extracting sentences from the document-set which start by heading words or contain cue words (called as, defined as, referred as etc.) related to education data. The non-summary list (NSL) is obtained by removing the sentences of the HISL, GSL, ISL, LSL, and HSL from the document-set.

Finally, the DSGA provides a variable size summary called the final summary (FS) based upon a learner's request. It calculates the summary length (SL) based upon the percentage of summary requested by the learner as in (3). In this work, the GSL contains information from the common sections (cliques) and the ISL contains information from unique sections of the document-set. The LSL contains information from the common sections of the individual documents.

$$SL = \left( \quad POS \ /100 \quad \right) * DL \tag{3}$$

where POS is the Percentage Of Summary requested by the learner.

The six different groups of summary lists generated by the DSGA are prioritized in the following order based on their importance. In all the cases, the sentences are sorted by their sentence score in descending order. The LSL uses the local score of each sentence. The remaining summary lists uses the global score of each sentence.

1. HISL (Globally and Locally important).
2. GSL (Globally important).
3. ISL (Globally diverse).
4. HSL (Heuristically important).
5. LSL (Locally important).
6. NSL (Remaining sentences of the document-set).

The final summary (FS) is generated by including the sentences from the first higher priority group, next higher priority group, and so on until the FS size equals the SL. In this manner, the DSGA always keeps the final summary more informative for any compression rate requested by a learner. First, the DSGA fills the FS with sentences from HISL. Next, it fills the FS with sentences from the GSL and ISL. It takes only thirty percent of the available ISL sentences because they are generally short sentences. So far there is no redundancy in FS. Then, it fills the FS with sentences from HSL. Then, it fills the FS with sentences from LSL, and NSL respectively, until the FS size is equal to the SL. It uses Cosine Similarity to avoid the insertion of redundant sentences into the FS. Finally, it sorts the FS by sentence number in ascending order. The FS contains the sentences of the first document, second document, and so on and provides an ordered look. Through post-processing, it resolves the referring expressions and arranges the summary sentences heading wise. Thus, it makes the final summary more coherent and meaningful.

## The DSGA Algorithm

The entire process of this work is done by the DSGA algorithm. The DSGA algorithm is given in Algorithm 2.

**Algorithm 2** Dynamic summary generation algorithm (DSGA)
**Input:** Preprocessed document-set, Local similarity matrix, and Global similarity matrix
**Output:** Final summary (FS)
Construct the global graph for the entire document-set and compute global features score for all sentences.
Construct the local graph for each document and compute local features scores for each sentence in each document.
Generate the *GSL* and *ISL* for the entire document-set using the *CBSG* algorithm.
Generate the *LSL* for each document using the *CBSG* algorithm. Also generate *HISL*, *HSL* and *NSL*.
Get the summary size in percentage from the learner and compute *SL*.
Fill the *FS* with the sentences of the *HISL*, *GSL*, *ISL*, *HSL*, *LSL* and *NSL* respectively till the size of the *FS* reaches *SL*.
Sort the *FS* in the increasing order of the sentence numbers and post-process it.
Display the original text of the sentences in the *FS*.
If the learner wants to continue go to 5.
Terminate the algorithm.

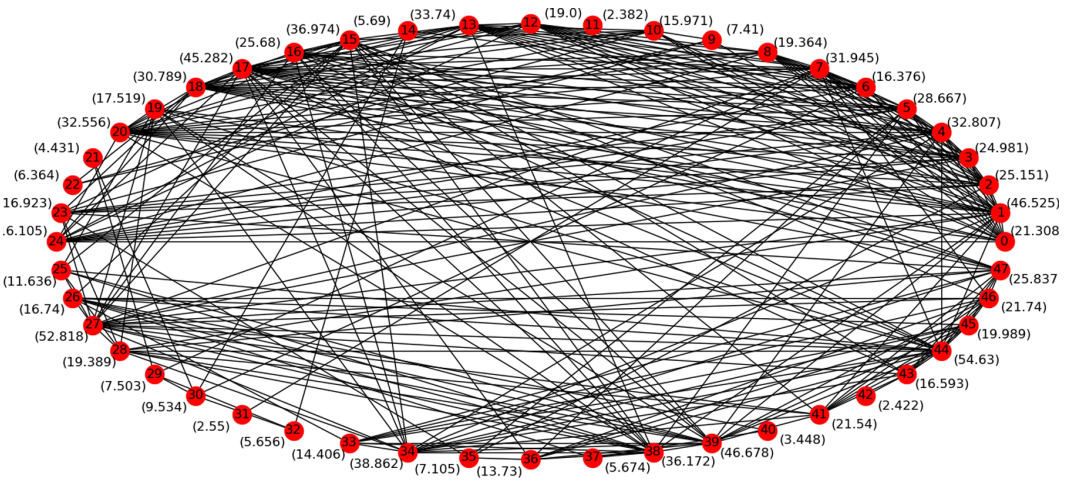
### Summary Generation Using One Sample Document-Set

This section explains the functioning of the proposed work with one sample topic, Basics of Operating System. It is the topic 1 of Table 1. It has two documents of length 48 and 13 sentences respectively. The length of the entire document-set DL is 61 sentences. The input documents document 1 and document 2 are given in Appendix A and Appendix B respectively. The sentence similarity graphs (local graphs) for document one and document two are shown in Figure 2 and Figure 3 respectively. The sentence graph of the entire document-set (global graph) is shown in Figure 4. In these graphs, the nodes represent sentences and the outside value of each node represents its sentence score. The various summary lists generated for the given input text are:

1. The GSL of the entire document-set is [17, 45, 28, 30, 22, 40, 55, 16, 35, 54, 49, 31, 42].
2. The LSL of individual documents are: [[1, 15, 6, 45, 23, 29, 37, 22, 31, 42], [49, 55, 48]].
3. The sentences common to GSL and LSL are stored in HISL. The HISL of the entire document-set is: [45, 22, 31, 42, 49, 55].
4. The GSL after removing HISL is: [17, 28, 30, 40, 16, 35, 54].
5. The LSL after removing HISL is: [[1, 15, 6, 23, 29, 37], [48]].
6. The Heuristically important sentences (HSL) are: [44,18,20,38,7,12,41,46,24,29] and
7. The Non summary sentences (NSL) are: [27,39,53,34,4,2,13, 51,3,0,5,10, 8,47,19,26,25,43, 33,50,36,9,58,14, 32,21,56,60,52,59].

The ISL is used to include unique information in the summary. But generally it contains very short sentences. So thirty percent sentences of the ISL are included in the summary. The thirty percent sentences of the ISL of the entire document-set are: [11, 57]. The SL greatly affects the summarization performance. A lower SL produces a concise summary with less information. A higher SL produces a detailed summary with insignificant information. Hence, this work performs a summary evaluation on a moderate-size summary, i.e. on a twenty five percent summary. So the size of the summary of

**Figure 2. Sentence graph for document one**




the sample document-set is 15 i.e. (25/100)*61=15. The summary generation process of the sample document-set is explained as follows.

1. Initially, the FS is empty and the SL is 15. First, it fills the FS with sentences from the HISL. Now, the FS is: [45, 22, 31, 42, 49, 55] and the SL is 9.
2. Next, it fills the FS with sentences from GSL. Now, the FS is: [45, 22, 31, 42, 49, 55, 17, 28, 30, 40, 16, 35, 54] and the SL is 2.
3. Next, it fills the FS with sentences from ISL. Now, the FS is: [45, 22, 31, 42, 49, 55, 17, 28, 30, 40, 16, 35, 54, 11, 57] and the SL is 0. Now it stops the summary generation process and post-processes the summary.
4. Referring expression 'it' occurs in the sentence 49. So add the previous sentence 48 in FS. Now the FS is: [45, 22, 31, 42, 48, 49, 55, 17, 28, 30, 40, 16, 35, 54, 11, 57].
5. The FS in sorted order is: [11, 16, 17, 22, 28, 30, 31, 35, 40, 42, 45, 48, 49, 54, 55, 57].
6. Finally, arrange the summary sentences heading wise and display the summary. The post processed output summary text is shown in Figure 5. The human summary is shown in Figure 6.

From the output summary text of Figure 5, it is clearly understood that the summary give an idea of the entire content of the document-set. Its ROUGE-1, ROUGE-2, and ROUGE-L recalls are 0.506, 0.204, and 0.277 respectively. It shows that the generated summary correlates with the human summary. So the generated summary is relevant to human summary and is informative. Also, Figure 5 has no redundant sentences. The connectedness of ideas in a text is known as text coherence. Since the referring expressions are resolved and the summary sentences are arranged heading wise, the coherence of the summary is good.

## Summary Evaluation

To assess the quality of the generated summary, summary evaluation is used. The two kinds of summary evaluations are: intrinsic evaluation and extrinsic evaluation (Jones & Galliers, 1995). The intrinsic evaluation is focused mainly on the quality and informativeness of the summaries. The extrinsic evaluation judges the summary quality on the basis of how the summaries are helpful for a given task. This article performs both kinds of evaluations. The experiments were performed on 2.5GHZ processor with 4GB RAM and 64bit OS. The summarizer of the proposed work is implemented in

**Figure 3. Sentence graph for document two**

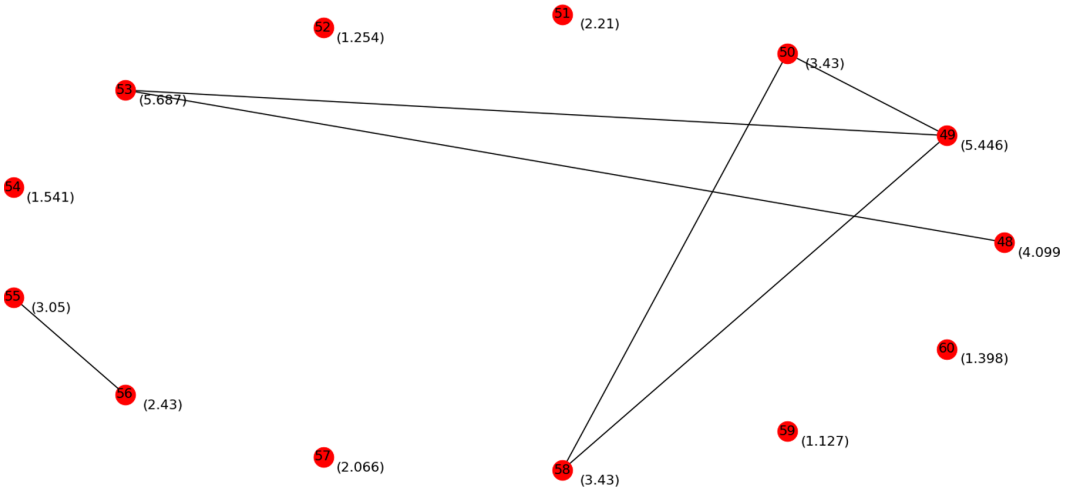
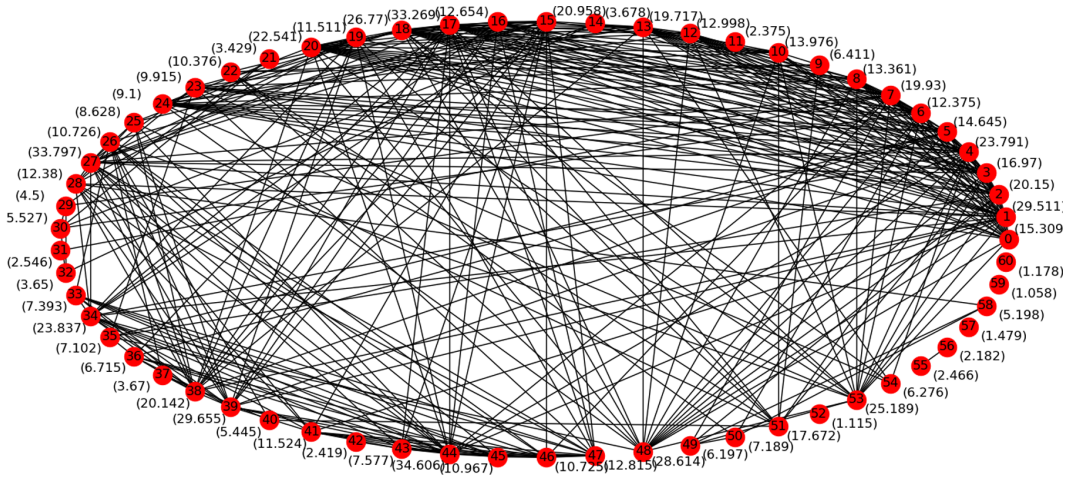



**Figure 4. Sentence graph for document-set**



Python 3.6 (32 bit). The Python packages NLTK 3.3 is used for preprocessing and NetworkX 2.1is used for graph construction.

The authors have constructed an Education dataset of fifteen topics on three engineering subjects such as Operating system, Computer architecture, and Software engineering. Each topic has three documents from various sources. Human summaries are also collected for each topic from subject experts. This work uses the graph algorithms Aggregate similarity (AS) and LexRank (LR) as baseline summarizers for comparison purpose. The AS measures the importance of a node by summing the weights of the edges, the node (sentence) has with other nodes in the graph (Al-Radaideh & Afif 2009). This article has implemented the AS using the Python 3.6 software and the Python package NetworkX 2.1. From the sentence graph of the document-set with a given threshold value, the LR selects summary sentences based on eigenvector centrality (Erkan & Radev 2004). The LR software lexrank 0.1.0 is taken from https://pypi.org/project/lexrank/.

**Figure 5. Output summary text**

```
25  Percent System Summary:
Final summary sentences after post processing are: [11, 16, 17, 22, 28, 30, 31, 35, 40, 42, 45, 48, 49, 54,
55, 57]
                    basics of operating system
11 . ability to evolve: an os should be constructed in such a way as to permit the effective development,
     testing and introduction of new system functions at the same time without interfering with service.
16 . the application program consists of business programs, database programs.
22 . os is designed to serve two basic purposes:it controls the allocation and use of the computing
     System's resources among the various user and tasks.
28 . provide routines that handle the details of i/o programming.
30 . each i/o device has a device handler that resides in a separate process associated with that device.
31 . the i/o subsystem consists of a memory management component that includes buffering caching
     and spooling.
35 . at one time, the computer programmer had at his disposal a basic machine that interpreted,
     through hardware, certain fundamental instructions.
42 . there are various loading schemes: absolute, relocating and direct-linking.
48 . an operating system is basically a intermediary agent between the user and the computer
     hardware.
49 . it manages the computer's resources.
54 . simplifies hardware control for applications.
55 . enforcer of sharing, fairness and security with the goal of better overall performance.
57 . trade-off between optimal algorithms and lean algorithms os is overhead.

operating system :
17 . every computer must have an operating system to run other programs.

assembler :
45 . in a simple loading scheme, the assembler outputs the machine language translation of a program
     on a secondary device and a loader places it in the core.

compiler :
0 . the same name  is often used to designate both a compiler and its associated language.
```

**Figure 6. Human summary**

**Human Summary:**

An Operating system is basically a intermediary agent between the user and the computer hardware.
It manages the computer's resources.
The purpose of an operating system is to provide an environment in which a user can execute programs in a convenient and efficient manner.
OS is designed to serve two basic purposes:It controls the allocation and use of the computing System's resources among the various user and tasks.
It provides an interface between the computer hardware and the programmer that simplifies and makes feasible for coding, creation, debugging of application programs.
The Operating system must support the following tasks.
Provides the facilities to create, modification of programs and data files using an editor.
Access to the compiler for translating the user program from high level language to machine language.
Provide a loader program to move the compiled program code to the computer's memory for execution.
Provide routines that handle the details of I/O programming.
The most important functions of an operating System are Memory Management, Processor Management, Device Management, and File Management.
some other functions are Security, Control over system performance, Job accounting, and Error detecting aids.
Various types of operating system are Batch Operating System, Time sharing operating System, Distributed operating System, Network operating system, and Real time operating system.

### *Intrinsic Content Based Evaluation*

This work performs the content based evaluation by comparing the DSGA summarizer with the baseline summarizers LR and AS experimentally using ROUGE evaluation. The ROUGE has a set of metrics that measure the content quality of a summary by comparing it with human summaries. These measures count the number of word overlaps between the computer generated summaries and the human summaries. This work computes ROUGE-1, ROUGE-2, and ROUGE-L for all the summaries (Lin & Hovy, 2003; Lin, 2004). ROUGE-1 is the unigram based co-occurrence statistics, ROUGE-2 is the bigram based co-occurrence statistics, and ROUGE-L is the longest common subsequence (LCS) based co-occurrence statistics. Independent words of a given text are known as unigrams. The sub sequences of two words of a given text are called bigrams. The longest common subsequence (LCS) of the given two sequences A and B is a common subsequence with maximum length. The LCS considers only in sequence matches. It does not require consecutive matches. The summaries of the DSGA, LR, and AS are produced for the fifteen topics (document-sets) in the Education dataset. These summaries are compared with human summaries and the Precision, Recall, and F-measure of these summaries are computed for ROUGE-1, ROUGE-2, and ROUGE-L.

### *Extrinsic Task-Based Evaluation*

The task-based evaluation is used to evaluate the level of satisfaction of real time learners. The authors selected 30 students studying first year MCA for conducting the task-based evaluation. The five topics namely, Bus structure, Cache memory, Direct memory access, Memory hierarchy, and Virtual memory in the subject Computer Architecture were used to perform the task-based evaluation. Each topic has three documents from various sources. First, the students were asked to answer ten descriptive type questions on each topic by providing only the source documents within 20 minutes in five different sessions. After one week, they were asked to answer the same set of questions for each topic by providing the source documents with summary within 20 minutes in five different sessions. The effect of summary on students' comprehension in answering descriptive type questions is measured.

## EVALUATION RESULTS AND DISCUSSION

Since ROUGE is a recall oriented measure, ROUGE-1, ROUGE-2, and ROUGE-L recalls of all the summaries for the fifteen topics are shown in Table 1. The bar charts for ROUGE-1, ROUGE-2, and ROUGE-L recalls are shown in Figures 7, 8, and 9 respectively. From Table 1and Figures 7, 8, and 9, it is clearly understood that the ROUGE-1, ROUGE-2, and ROUGE-L recalls of the DSGA summary is greater than or equal to the ROUGE-1, ROUGE-2, and ROUGE-L recalls of the LR and AS summaries in most of the cases. Also, the average ROUGE-1, ROUGE-2, and ROUGE-L recalls of the DSGA summary is greater than or equal to the ROUGE-1, ROUGE-2, and ROUGE-L recalls of the LR and AS summaries. So the DSGA summarizer correlates highly with human summarizer than the baseline summarizers. Hence, it produces more informative summary than baseline summarizers. Also from Table 1, it is clearly understood that ROUGE-1 correlates highly with human summarizer than ROUGE-2 and ROUGE-L.

The task-based evaluation results are given in Table 2. From Table 2, it was seen that the students performed well for each topic when provided with summary than provided without summary. Also, the authors observed that the students answered quickly when provided with a summary. They took more time to answer when no summary is given. Thus, it was clearly understood that the summary improves the students' understanding and comprehension.

**Table 1. ROUGE evaluation results**

| ROUGE Evaluation of Education Dataset | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Topic ID** | **ROUGE-1 Recall** | | | **ROUGE-2 Recall** | | | **ROUGE-L Recall** | | |
| | **DSGA** | **LR** | **AS** | **DSGA** | **LR** | **AS** | **DSGA** | **LR** | **AS** |
| **1** | 0.3614 | 0.3494 | 0.3373 | 0.1947 | 0.1593 | 0.1858 | 0.2410 | 0.1205 | 0.2289 |
| **2** | 0.656 | 0.6 | 0.6 | 0.4352 | 0.4249 | 0.4560 | 0.448 | 0.28 | 0.392 |
| **3** | 0.6547 | 0.6906 | 0.6978 | 0.2745 | 0.4461 | 0.5049 | 0.2950 | 0.3597 | 0.3813 |
| **4** | 0.5877 | 0.5 | 0.5088 | 0.2995 | 0.2030 | 0.2183 | 0.3246 | 0.2982 | 0.2281 |
| **5** | 0.6847 | 0.4054 | 0.3694 | 0.3614 | 0.2169 | 0.2048 | 0.3604 | 0.1892 | 0.2342 |
| **6** | 0.625 | 0.5469 | 0.4688 | 0.4167 | 0.4167 | 0.3125 | 0.4375 | 0.3438 | 0.3438 |
| **7** | 0.648 | 0.544 | 0.488 | 0.3980 | 0.3383 | 0.2836 | 0.36 | 0.264 | 0.288 |
| **8** | 0.6491 | 0.5789 | 0.5965 | 0.3571 | 0.3452 | 0.4167 | 0.3333 | 0.2982 | 0.4561 |
| **9** | 0.5918 | 0.6327 | 0.5714 | 0.2206 | 0.3235 | 0.3088 | 0.3469 | 0.2857 | 0.3878 |
| **10** | 0.7313 | 0.6866 | 0.7015 | 0.4706 | 0.4902 | 0.5196 | 0.4329 | 0.2836 | 0.6119 |
| **11** | 0.4176 | 0.4505 | 0.3407 | 0.2035 | 0.2743 | 0.1504 | 0.2967 | 0.2418 | 0.2637 |
| **12** | 0.6813 | 0.4945 | 0.6154 | 0.4394 | 0.2727 | 0.3182 | 0.5165 | 0.2637 | 0.3077 |
| **13** | 0.5377 | 0.5 | 0.5755 | 0.1813 | 0.325 | 0.4438 | 0.2547 | 0.3302 | 0.5 |
| **14** | 0.7541 | 0.6557 | 0.4590 | 0.5625 | 0.5313 | 0.2396 | 0.6557 | 0.4426 | 0.2623 |
| **15** | 0.7711 | 0.8072 | 0.6627 | 0.5116 | 0.6434 | 0.5194 | 0.4458 | 0.4819 | 0.5663 |
| **Average** | 0.623 | 0.563 | 0.533 | 0.36 | 0.36 | 0.339 | 0.383 | 0.299 | 0.363 |

## CONCLUSION

This work increases concept coverage and reduces redundancy by selecting sentences from the diverse set of cliques. Through post-processing, it resolves the referring expressions and arranges the summary sentences heading wise. Thus, it makes the final summary more coherent and meaningful. This work is suitable for all kinds of learners by providing a summary of user specified length. So it is generic. From the content based evaluation, it is understood that the DSGA summaries are relevant to human summaries and are informative. Thus, this work produces non-redundant, coherent, generic, and informative summaries that cover all concepts of the given input text. Since informativeness, coherence, coverage, and non-redundancy are the main characteristics of any learning material and this work supports all these characteristics, it is suitable for summarizing learning materials.

The benefits of this work are: (i) it is used for summarizing a single learning material or multiple learning materials; (ii) it is applicable to summarize learning materials that do not contain any summary or outline; (iii) it is used for the learners who want to access the key information of the source materials quickly; (iv) it is useful for learning through low bandwidth devices such as mobile phones.

The future work is to produce a query focused summary to clarify a learner's doubt or query regarding the generated summary. The query summary can be used to assess the learners' descriptive answers for the same query in the future. The learners' descriptive answers can be evaluated by measuring the relevance of the answers to the query-focused summary and can be awarded with marks based on the relevance score.

Figure 7. ROUGE-1 Evaluation Results

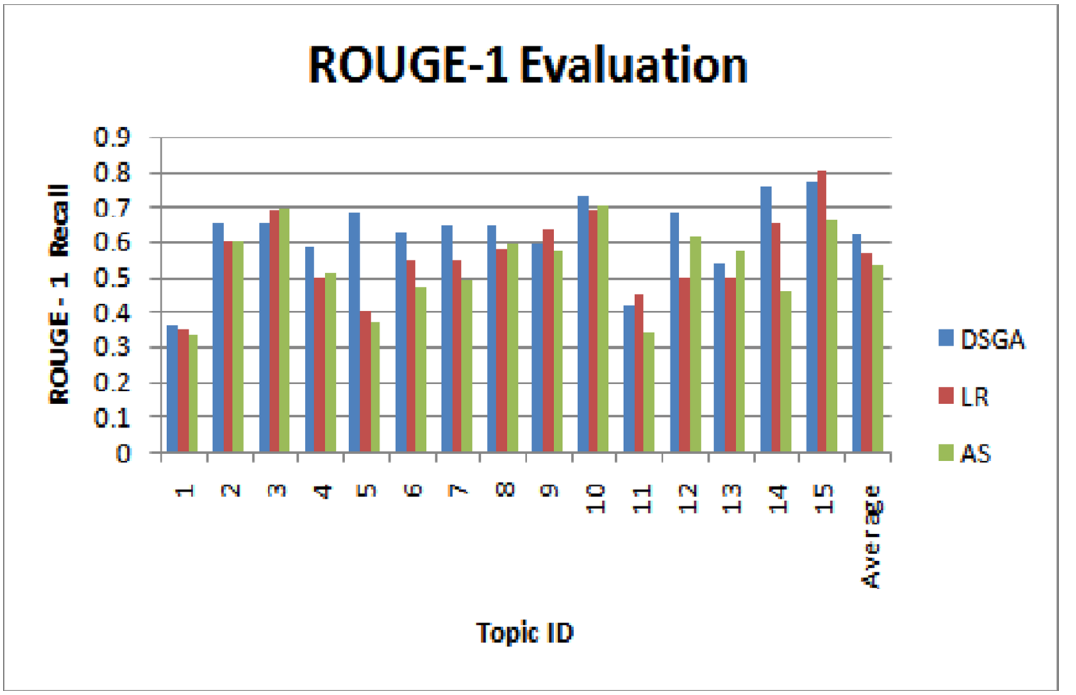



Figure 8. ROUGE-2 Evaluation Results

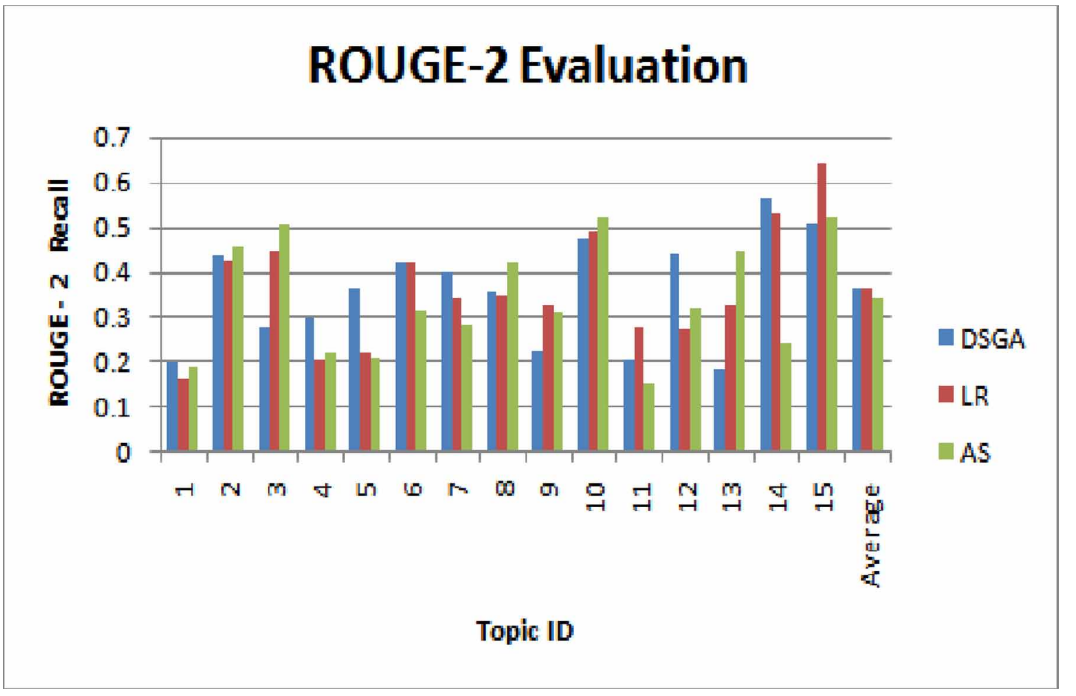


**Figure 9. ROUGE-L Evaluation Results**

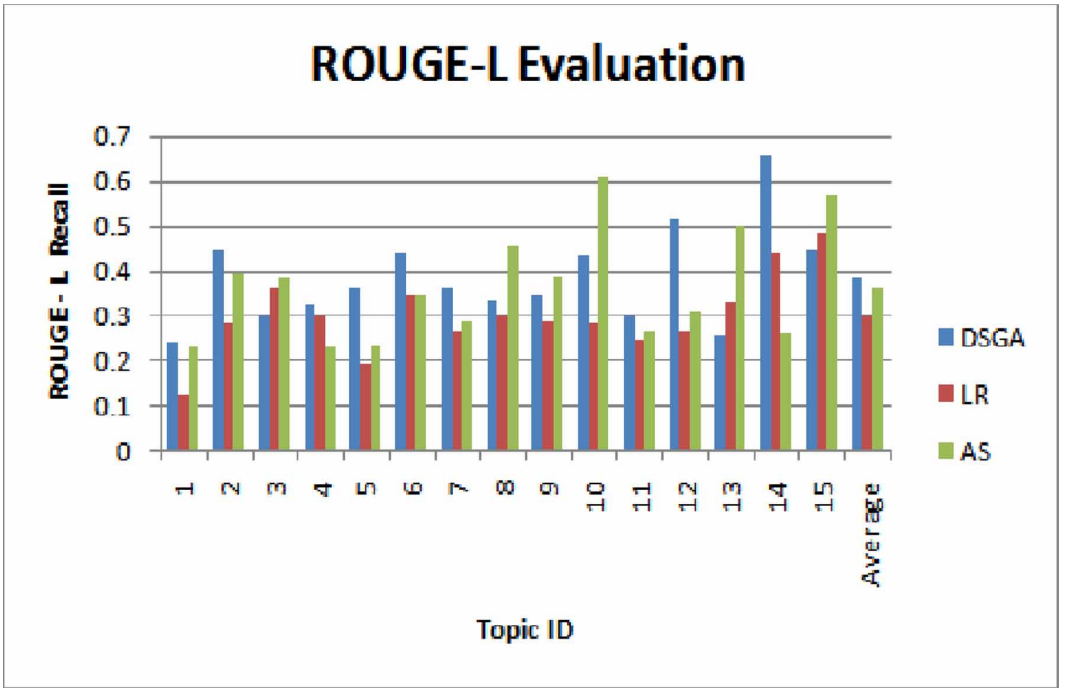



**Table 2. Task based evaluation results**

| | A test of 10 questions was conducted | | Source Documents Length | Summary Length |
|---|---|---|---|---|
| | Average Number of questions answered | | | |
| | With Summary | Without summary | | |
| **Topic 1** | 10 | 9 | 75 | 19 |
| **Topic 2** | 8 | 6 | 158 | 40 |
| **Topic 3** | 9 | 7 | 115 | 29 |
| **Topic 4** | 10 | 8 | 84 | 21 |
| **Topic 5** | 9 | 6 | 124 | 34 |

## REFERENCES

Al-Radaideh, Q. A., & Afif, M. (2009). Arabic text summarization using aggregate similarity. *The 2009 International Arab Conference on Information Technology (ACIT'2009)*.

Al-Saleh, A., & Menai, M. (2018). Solving multi-document summarization as an orienteering problem. *Algorithms*, *11*(7), 1–27. doi:10.3390/a11070096

Baralis, E., & Cagliero, L. (2016). Learning from summaries: Supporting e-learning activities by means of document summarization. *IEEE Transactions on Emerging Topics in Computing*, *4*(3), 416–428. doi:10.1109/TETC.2015.2493338

Cagliero, L., Farinetti, L., & Baralis, E. (2019). Recommending personalized summaries of teaching materials. *IEEE Access : Practical Innovations, Open Solutions*, *7*, 22729–22739. doi:10.1109/ACCESS.2019.2899655

Cai, X., & Li, W. (2013). Ranking through clustering: An integrated approach to multi-document summarization. *IEEE Transactions on Audio, Speech, and Language Processing*, *7*, 424–433.

Calvo, H., Carrillo-Mendoza, P., & Gelbukh, A. (2018). On redundancy in multi-document summarization. *Journal of Intelligent & Fuzzy Systems*, 1–11.

Cao, Z., Li, W., Li, S., & Wei, F. (2017). Improving multi-document summarization via text classification. *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 3053-3059.

Conte, A. (2013). *Review of the Bron-Kerbosch algorithm and variations*. School of Computing Science, University of Glasgow.

Erkan, G., & Radev, D. (2004). LexRank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, *22*, 457–479. doi:10.1613/jair.1523

Feiyue, Y., & Xinchen, X. (2018). Automatic multi-document summarization based on keyword density and sentence-word Graphs. *J. Shanghai Jiao Tong Univ (Sci.)*, *23*(4), 584–592. doi:10.1007/s12204-018-1957-2

Fejer, H. N., & Omar, N. (2015). Automatic multi-document arabic text summarization using clustering and key phrase extraction. *Journal of Artificial Intelligence*, *8*(1), 1–9. doi:10.3923/jai.2015.1.9

Ferreira, R., de Souza Cabral, L., Freitas, F., Lins, R. D., de França Silva, G., Simske, S. J., & Favaro, L. (2014). A multi-document summarization system based on statistics and linguistic treatment. *Expert Systems with Applications*, *41*(13), 5780–5787. doi:10.1016/j.eswa.2014.03.023

Ferreira, R., de Souza Cabral, L., Lins, R. D., Pereira e Silva, G., Freitas, F., Cavalcanti, G. D. C., Lima, R., Simske, S. J., & Favaro, L. (2013). Assessing sentence scoring techniques for extractive text summarization. *Expert Systems with Applications*, *40*(14), 5755–5764. doi:10.1016/j.eswa.2013.04.023

Goldstein, J., Mittal, V., Carbonell, J., & Kantrowitz, M. (2000). Multi document summarization by sentence extraction. In *Proceeding of the 2000 NAACL-ANLP Workshop on Automatic summarization*. ACM Press. doi:10.3115/1117575.1117580

Han, J., & Kamber, M. (2006). *Data mining: Concepts and techniques*. Morgan Kaufmann Publishers.

John, A., & Wilscy, M. (2015). Vertex cover algorithm based multi-document summarization using information content of sentences. *Proceedings of the International Conference on Information and Communication Technologies, 46*, 285-291. doi:10.1016/j.procs.2015.02.022

Jones, K. S., & Galliers, J. R. (1995). *Evaluating natural language processing systems: An analysis and review*. Springer Science and Business Media.

Kanitha, D. K., Noorul Mubarak, D. M., & Shanavas, S. A. (2018). Survey on text summarization methods. *International Journal of Computer Engineering and Technology*, *9*(1), 26–36.

Lin, C. (2004). ROUGE: A package for automatic evaluation of summaries. *Proceedings of the Workshop on Text Summarization*, 74-81.

Lin, C., & Hovy, E. (2003). Automatic evaluation of summaries using n-gram co-occurrence statistics. *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, 150-157. doi:10.3115/1073445.1073465

Mihalcea, R., & Tarau, P. (2004). TextRank: Bringing order into texts. In *Proceedings of the Empirical Methods in Natural Language Processing*, 404-411.

Mustamiin, M., Budi, I., & Santoso, H. B. (2017). Multi-documents summarization based on clustering of learning object using hierarchical clustering. *2nd International Conference on Computing and Applied Informatics, 978*(1). doi:10.1088/1742-6596/978/1/012053

Nandhini, K., & Balasundaram, S. R. (2013). Use of genetic algorithm for cohesive summary extraction to assist reading difficulties. *Applied Computational Intelligence and Soft Computing*, 1–11.

Radev, D. R., Hovy, E., & McKeown, K. (2002). Introduction to the special issue on summarization. *Computational Linguistics*, *28*(4), 399–408. doi:10.1162/089120102762671927

Ramanujam, N., & Kaliappan, M. (2016). An Automatic multi-document text summarization approach based on naive bayesian classifier using timestamp strategy. *The Scientific World Journal*, 1–10. doi:10.1155/2016

Rossiello, G., Basile, P., & Semeraro, G. (2017). Centroid-based text summarization through compositionality of word embeddings. *Proceedings of the Multi Ling 2017 Workshop on Summarization and Summary Evaluation Across Source Types and Genres*, 12–21.

Shimada, A., Okubo, F., Yin, C., & Ogata, H. (2015). Automatic summarization of lecture slides for enhanced student preview. In *Proceedings of the 23rd International Conference on Computers in Education*, (pp. 218-227). Asia-Pacific Society for Computers in Education.

Tomita, E., Akutsu, T., & Matsunaga, T. (2011). Efficient algorithms for finding maximum and maximal cliques: Effective tools for bioinformatics. *Biomedical Engineering, Trends in Electronics, Communications and Software*, 625-641.

Wang & Cai. (2009). Personalized E-learning model based on teacher student collaboration. IEEE Computer Society.

# APPENDIX A

## Input Document 1

**Figure 10.**

**Input File 1** **Introduction of Operating System**

An operating system acts as an intermediary between the user of a computer and computer hardware. The purpose of an operating system is to provide an environment in which a user can execute programs in a convenient and efficient manner.
An operating system is a software that manages the computer hardware. The hardware must provide appropriate mechanisms to ensure the correct operation of the computer system and to prevent user programs from interfering with the proper operation of the system.
Operating System
An operating system is a program that controls the execution of application programs and acts as an interface between the user of a computer and the computer hardware.
A more common definition is that the operating system is the one program running at all times on the computer (usually called the kernel), with all else being application programs.
An operating system is concerned with the allocation of resources and services, such as memory, processors, devices, and information. The operating system correspondingly includes programs to manage these resources, such as a traffic controller, a scheduler, memory management module, I/O programs, and a file system.
Functions of Operating system – Operating system performs three functions:
Convenience: An OS makes a computer more convenient to use.
Efficiency: An OS allows the computer system resources to be used in an efficient manner.
Ability to evolve: An OS should be constructed in such a way as to permit the effective development, testing and introduction of new system functions at the same time without interfering with service.
Operating system as User Interface.
Every general-purpose computer consists of the hardware, operating system, system programs, and application programs. The hardware consists of memory, CPU, ALU, and I/O devices, peripheral device, and storage device. System program consists of compilers, loaders, editors, OS, etc.The application program consists of business programs, database programs.
Every computer must have an operating system to run other programs. The operating system coordinates the use of the hardware among the various system programs and application programs for various users. It simply provides an environment within which other programs can do useful work.
The operating system is a set of special programs that run on a computer system that allows it to work properly. It performs basic tasks such as recognizing input from the keyboard, keeping track of files and directories on the disk, sending output to the display screen and controlling peripheral.
OS is designed to serve two basic purposes:It controls the allocation and use of the computing System's resources among the various user and tasks.
It provides an interface between the computer hardware and the programmer that simplifies and makes feasible for coding, creation, debugging of application programs.
The Operating system must support the following tasks.
Provides the facilities to create, modification of programs and data files using an editor.
Access to the compiler for translating the user program from high level language to machine language.
Provide a loader program to move the compiled program code to the computer's memory for execution.
Provide routines that handle the details of I/O programming.
I/O System Management
The module that keeps track of the status of devices is called the I/O traffic controller. Each I/O device has a device handler that resides in a separate process associated with that device.
The I/O subsystem consists of a memory Management component that includes buffering caching and spooling.
It also contains general device driver interface and drivers for specific hardware devices.
Assembler
The input to an assembler is an assembly language program. The output is an object program plus information that enables the loader to prepare the object program for execution. At one time, the computer programmer had at his disposal a basic machine that interpreted, through hardware, certain fundamental instructions. He would program this computer by writing a series of ones and Zeros (Machine language), place them into the memory of the machine.
Compiler
The High-level languages- examples are FORTRAN, COBOL, ALGOL and PL/I are processed by compilers and interpreters. A compiler is a program that accepts a source program in a "high-level language "and produces a corresponding object program. An interpreter is a program that appears to execute a source program as if it was machine language. The same name (FORTRAN, COBOL, etc.) is often used to designate both a compiler and its associated language.
Loader
A Loader is a routine that loads an object program and prepares it for execution. There are various loading schemes: absolute, relocating and direct-linking. In general, the loader must load, relocate and link the object program. The loader is a program that places programs into memory and prepares them for execution. In a simple loading scheme, the assembler outputs the machine language translation of a program on a secondary device and a loader places it in the core. The loader places into memory the machine language version of the user's program and transfers control to it. Since the loader program is much smaller than the assembler, those make more core available to the user's program.

## APPENDIX B

## Input Document 2

**Figure 11.**

**Input file 2 :**

An Operating system is basically a intermediary agent between the user and the computer hardware.
It manages the computer's resources.
It's a resource allocator.
It is also used to control programs to prevent errors and improper computer use.
It is interrupt driven.
Users and Processes access the Computer's resources through the Operating System.
1.1.2. Operating System Benefits
Simplifies hardware control for applications.
Enforcer of sharing, fairness and security with the goal of better overall performance.
Trade-off between fairness and performance.
Trade-off between optimal algorithms and lean algorithms – OS is overhead.
Provides abstract resources.
Sockets.
Inter-process communication.

*Krishnaveni P. received the B.E. degree in Computer Science and Engineering from Bharathidasan University, Tamil Nadu. She received the M.E. degree in Computer Science and Engineering from Government College of Engineering, Tirunelveli, Tamil Nadu. She is currently pursuing the Ph.D. degree in the Department of Computer Applications, National Institute of Technology, Tamil Nadu. Her research interests include Information retrieval, Natural language processing, and Automatic text summarization.*

*S. R. Balasundaram has been working since 1987 at the National Institute of Technology, Tiruchirappalli. He completed M.E. in Computer Science & Engineering during 1992. Currently, he is working as a Professor in the Department of Computer Applications, earned his doctorate in "E-Learning and Assessment" from NIT, Trichy, and has more than 40 papers in reputed Journals and Proceedings of International Conferences. His areas of interest are: Web & Mobile Technologies, Cognitive Sciences, and e-Learning Technologies. He has authored a book, co-author for 3 book chapters and edited a book.*