Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

ARTICLE TYPE

Unsupervised Summarization via Cliques Algorithm †

Nesreen Alsharman*[1] | Inna Pivkina[2]

[1]WISE, Amman, Jordan
[2]NMSU, NM, USA

Correspondence
*Nesreen Alsharman Email:
nesreen.alsharman@wise.edu.jo

Present Address
Tabarbour, PO Box 1101, Zip Code 11947
Amman - Jordan-Dr.Nesreen Alsharman

Summary

The paper introduces new approaches to generate summaries. Summaries are generated by detecting different topics (clusters of sentences) in a document, building summary sentences for each topic, and adding them to the summary. It allows summaries to reflect different ideas from the document. Different clusters in a document are identified by finding cliques in a sentence similarity graph. Selecting representative sentences from the clusters could be done in two ways. The first way is to use Multi-Sentence Compression method for each cluster to generate a compression sentence that reflects the main idea in the cluster (abstractive cliques method). All generated compression sentences are used to build a final abstractive summary. The second way is to extract one sentence from each cluster that is the most similar to sentences in the cluster. This results in an extractive summarization (extractive cliques method). By picking representative sentence for each cluster we reduce potential redundant information in the summary and decrease the chances of missing some important ideas from the text. The approaches are tested on DUC 2004 (Task 3 and 4) data set which contain news documents. ROUGE Perl toolkit is used to compare automatically produced summaries against a set of reference summaries from the data sets. Results show that both approaches perform better than Lex-Rank. Extractive cliques method performs slightly better than abstractive cliques method.

KEYWORDS:
extractive summarization, abstractive summarization, TED, cliques

1 | INTRODUCTION

The quantity of online information is growing exponentially. Generating a condensed version of a passage while preserving its meaning is known as text summarization[1]. Summarizing information allows the understanding of main ideas of documents without reading all the content of text which helps to reduce reading time. A summary can be employed in an indicative way as a pointer to some parts of the original document, or in an informative way to cover all relevant information of the text[2]. In both cases, the primary advantage of using a summary is reduced reading time.

Summarization systems can be broadly classified into two categories. Extractive summarization produces summaries by concatenating several sentences taken exactly as they appear in the original documents being summarized. By contrast, abstractive summarization uses different words to describe the contents of the original documents rather than directly copying original sentences[3].

Sentence similarity calculation is a crucial task for many approaches to text summarization[4,5]. Computation of similarity between sentences is a basic operation for text summarization[6,7,8]. The main principal operation for calculation of similarity of sentences is string comparison. There are extensive[9] studies conducted on the comparison of string data using simple word overlap measures (bag-of-words, n-grams), but they are not sufficient to solve these tasks accurately[10]. The main problem when using word overlap measures is the lack of understanding of the syntactic and grammatical relationship between words and phrases in the sentence. Also, the bag-of-words model is sometimes inadequate for capturing the syntactic and grammatical similarities among the sentences, which may affect the qualities of the summaries[9]. In this research, the sentence similarities are computed as similarities between dependency trees representations of sentences.

Similarity of dependency trees representations of sentences (syntactic n-grams) is computed using TED. TED is a generalization of the edit distance for two strings. Edit distance measures the similarity between two strings. TED extends edit distance to measuring the similarity between two trees. TED is defined as the minimum cost of edit operations to transform one tree into another. The most common TED edit operations are: (1) insert a node into a tree, (2) delete a node from a tree, and (3) change the label of a node[11].

The main features in this research are as follows. We use TED to find syntactic and grammatical similarities among the sentences and build a sentence similarity graph. We find clusters in the text as cliques in the graph. Summary contains sentences obtained from clusters.

The rest of the paper is structured as follows. Section 2 presents related work. The methodology is described in Sections 3-6. Sections 4-5 talk about computing sentence similarity scores and finding sentence clusters. Section 5 presents the proposed abstractive summarization method. Section 6 presents the proposed extractive summarization method. Evaluation results are presented in Section 7 that is followed by conclusions.

2 | RELATED WORK

The sub-field of automatic summarization has been investigated by the NLP community. Several different methods have been proposed in the literature for extractive text summarization: supervised approaches[12,13], and unsupervised approaches[14,6].

In general, the main idea of supervised approaches is given a set of training documents and their extractive summaries, the summarization process is modeled as a machine learning classification problems: sentences are classified as summary sentences and non-summary sentences based on the features that they possess.

Our proposed method is an unsupervised method. Recently, unsupervised graph-based summarization methods have attracted the increasing attention of researchers[6,7,8] and have been successful when compared to the other state-of-the art summarization approaches[14].

LexRank[6] is one of the most salient graph-based methods. The general idea is that sentences that are similar to many other sentences are likely to cover the most important points in the text and could be in a summary. Like most of the other graph-based studies, LexRank uses cosine similarity based on the Term Frequency-Inverse Document Frequency (TF-IDF) metric to measure the similarities among the sentences (nodes in a graph representation of the text). After finding similarity between sentences, LexRank extracts the most important sentences based on the concept of eigenvector centrality in the graph.

Some methods treat sentences as bags-of-words such as LexRank. This representation may fail to capture some of the grammatically related information, which in turn may affect the summary quality negatively[10].

Some researchers such as[15] address the bag-of-word representation by using the supervised probabilistic model over syntactic trees for summarization. Another research[16] also address this problem using Syntactic dependencies and semantic frames that reflect important grammatical and semantic relations in the text. They used dependency subtrees concept. In particular, they extract labeled and unlabeled syntactic dependencies, e.g., DEPENDENCY(drinks,Alaa) or SUBJECT(drinks,Alaa), from sentences and represent them by such syntactic concepts. They used the Stanford parser to augment documents with syntactic dependencies. In addition they used semantic frames feature that a summary should represent the most central semantic frames as[17]. As a first step in modeling relevant concepts, Brody and Kantor[18] also employed a word-gram representation, and use frequency as a measure of relevance.

In this research, we produce two methods to generate summaries that employ structure similarity: the first system is extractive summary could be built by selecting sentence with minimum TED average (closeness centrality) according to the clique to build a summary (extractive cliques summarization). TED algorithm that non-probabilistic and not trained on data that make our methods are general enough to summarize texts from any domain. Dependency trees are non-linear tree structures, the TED

should be the right measure for their comparison[9]. The second approach, we use Multi- Sentence Compression method for each clique to generate a compression sentence to reflect the main idea in the clique. All generated compression sentences are used to build a final abstractive summary (abstractive cliques summarization).

3 | COMPUTING SENTENCES SIMILARITIES

Computing sentences similarities stage consists of the following sub stages: 1) extracting Stanford dependency grammar relations, 2) building dependency tree for each sentence, 3) extracting candidate sentences, and 4) using TED between dependency trees for candidate sentences to find sentences similarities.

3.1 | Extracting Stanford Dependency Grammar Relations

According to[19] short sentences do not contain enough information to be included in a summary. We followed this recommendation in our work and ignored sentences that have less than ten words. For each sentence in the text which is not too short, we get syntactic grammatical relations between the words in the sentence. We use Turbo Semantic Parser for this task[20]. Turbo Semantic Parser performs tokenization and produces part-of-speech tagging and Stanford Dependency Grammar Relations.

3.2 | Building Dependency Tree for each Sentence

From the previous step, we have Stanford dependency grammar representations for all sentences in the text. The representations contain grammatical and syntactic relations between words in a sentence. From these relations, the dependency tree for each sentence is built to make a comparison between sentences. A tree T is a data structure made up of nodes $N(T)$ and edges E(T) $\subseteq N(T) \times N(T)$ without having any cycles and each node has at most one incoming edge. While building the dependency tree we maintain a queue of nodes (words) to be processed. Initially, a queue is empty. First, begin with root-relation to pick up the root node of the dependency tree. Add the root node to a queue. Second, for each node X in the top of queue, delete the node from the queue, find all words related to the node from relations and add them as children (as new nodes) to the node with edges that show the types of relations. Append all added nodes to the end of the queue. Third, recursively repeat the second step for new words (nodes) until all words are added to the tree.

A dependency tree satisfies the following constraints: 1) there is a single root node that has no incoming edges, 2) each vertex has exactly one incoming edge with the exception of the root node and 3) there is a unique path from the root node to each vertex in $N(T)$. Figure 1 shows dependency trees of the sentences: (a) “The dog was playing with a yellow ball”and (b) “The furry dog was playing with a ball”.

3.3 | Syntactic Representation vs. Bag-of-Words Representation

This research uses a syntactic representation of sentences like[15,16] methods. Some other summarization methods[11,21,6] use a bag-of-words representation. Syntactic representation can express the structure of the text, and can distinguish between the events and their environment in the text. Using a general set of relations between the events and their participants, the same events or entities may be represented in different perspectives. Keeping these features in mind, a dependency tree is built for each sentence in the text that reflects these features. One of the problems of the bag-of-words representation is that it ignores the order between words in a sentence. We illustrate this problem with a example. Let us consider two sentences: 1) “Tim votes for Mary”, 2) “Mary votes for Tim”. According to the bag-of-words representation, both sentences are the same. But according to dependency tree relations, sentences are different because they have different tree representations.

3.4 | Extracting Candidate Sentences

While building the dependency tree representation for each sentence, structures’ frequencies from grammatical relations are computed for the following five structures types: 1) subject, 2) verb, 3) object, 4) subject verb, and 5) verb object. These grammatical structures reflect the core meaning of the sentence. There are other types of grammatical structures in a sentence, but they

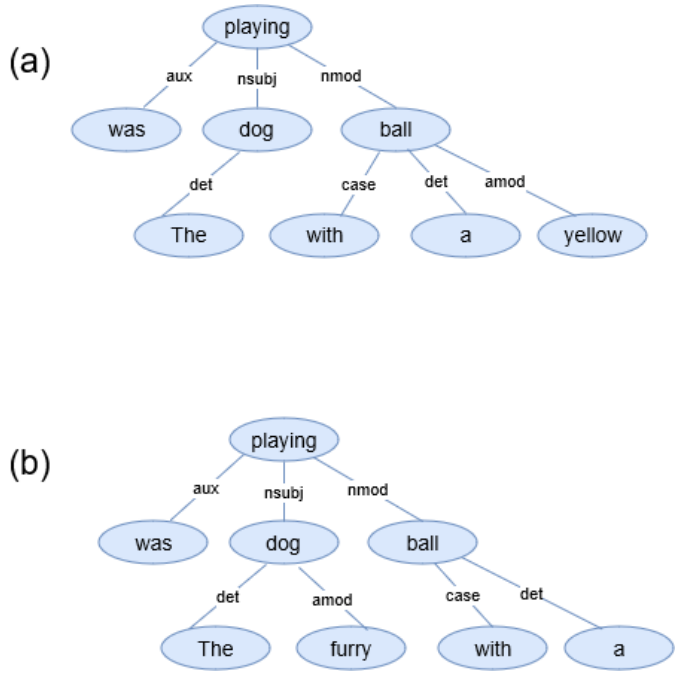

FIGURE 1 dependency trees: (a)"The dog was playing with a yellow ball"and (b) "The furry dog was playing with a ball"sentences

are less important for the purpose of selecting sentences for summaries. When a comparison between sentences is made to see if they are similar or not, the above five types of grammatical structures reflect the most overlap content between the sentences.

We determine the most frequent grammatical structures of the above five types for the whole text. Each sentence is assigned a score that is the number of occurrences of the most frequent grammatical structures in the sentence. Sentences with higher scores are selected as candidates to be in the summary. To calculate most frequent grammatical structures, term frequency (tf) method is utilized and the following equation is used to find most frequent grammatical structures:

$$R(f) = N \times tf(f, d), \tag{1}$$

where f is the candidate structure of a relation, N is the number of words in f, and $tf(f, d)$ is the number of times that structure f occurs in the input text d. Finally, the sentences that had the highest scores (most frequent grammatical structures) are returned as candidate sentences to be in a summary. Extracting candidate sentences reduce computation time for our approach because calculating tree edit distance (TED) in the following step needs a long time for large data set. In this way, we select candidate sentences to build summary and find similarity between them to reduce complexity time. To select sentences for a summary we compute tree edit distance between tree representations of sentences as discussed in the next section. However, computing tree edit distances is very computationally expensive. In order to make the process more efficient we compute tree edit distance not for all of the sentences in the text but only for the candidate sentences

3.5 | Calculating Tree Edit Distance (TED)

We have a set of candidate sentences which are represented as dependency trees. Our goal is to choose sentences which have higher similarity to others in the set. To compute similarity between sentences, TED is utilized. Zhang and Shasha[22] developed a TED algorithm to find similarity for a given pair of trees. Dynamic programming is often used to calculate the tree edit distance, and the most popular dynamic programming algorithm to compute tree edit distance between two trees is the algorithm of Zhang-Shasha[22]. We use Java implementation of the algorithm from[23] to calculate TED using dynamic programming.

Let us consider two sentences in Figure 1 as an example to calculate TED. They are very similar in meaning. TED between the sentences computed using the Zhang-Shasha algorithm is 2 (add "furry" and remove "yellow" words). It does not depend on the number of characters in the words that are being added or removed. But if we calculate string edit distance using the Levenshtein distance[24] between sentences (a) and (b) from Figure 1, the result is 11 characters because it counts how many characters there

are in each word. The words can have different lengths; therefore, the number of characters that need to be changed is not a good indicator of how different the sentences are. For instance, edit distance of 14 characters could mean adding 4 words (e.g., "I am busy now") or adding one long word (e.g. "characteristic"). The TED gives a more intuitive result when talking about similarities of sentences because it reflects the number of words (not the characters) that need to be changed when transferring one sentence into another.

Using TED for measuring similarity between sentences is sensitive to typos. Consider two sentences that mention the name of Albuquerque city and one sentence has a typo in the name of the city. If the typo is not corrected, TED distance would treat the city names as two different cities. Therefore, it is important to perform spell checking before applying TED algorithm to sentences.

Sentence similarity is computed as inverse similarity between dependency trees of the sentences which is defined as follows[9].

$$similarity_1(T_i, T_j) = \frac{1}{(1+d)} \tag{2}$$

where $d = TED(T_i, T_j)$ and T_i, T_j are trees. For example, the similarity between the tree in Figure 1 (a) (let us call it T_1) and the tree in Figure 1 (b) (let us call it T_2) using the TED is: $similarity(T_1, T_2) = 0.33$. The formula reflects the fact that the more the TED is the less similar the sentences are.

4 | SELECTING SENTENCE CLUSTERS

Given a set of dependency trees that reflect syntactic and grammatical relations between words in the sentences and TED scores between each pair of sentences (a pair of dependency trees) we build a sentence similarity graph in the following manner. Vertices in the graph are sentences. Two vertices are connected by an edge if their sentences are similar enough. By similar enough we mean that TED between the sentences is less than a certain threshold. The threshold value is selected according to the average length of sentences of the input text. For Example, the average sentence length in example in Figure 3 is 29 and threshold is set to be 30. The sentence similarity graph for the example in Figure 3 is shown in Figure 2. After building the sentence similarity graph , we find all cliques in the graph that contain more than one node. A clique in an undirected graph $G = (V, E)$ is a subset of the vertex set $C \subseteq V$, such that for every two vertices in C, there exists an edge connecting the two. Each clique corresponds to a cluster of sentences that interact with one another and share a similar idea. A set of clusters of related sentences from the text corresponds to a set of different ideas from the text. A summary could be generated by picking representative sentences for each cluster.

Figure 3 shows an example of a document from DUC 2004 (Task 3 and 4) data set. Figure 2 shows a sentence similarity graph for the document in Figure 3. The cliques in the graph that contain more than one node are the following: {sentence 5, sentence 6}, {sentence 4, sentence 9, sentence 2 }, { sentence 4, sentence 8, sentence 7}, and { sentence 4, sentence 10 }. Therefore, the document contains four clusters.

We find topic clusters to make sure that all significant ideas in the text are mentioned in a summary. In some previous work[14,6] sentences with highest similarity to the text are chosen to be in a summary. This may sometimes lead to similar sentences reflecting the same idea to appear together in a summary resulting in some duplication of information in the summary. Also, some ideas from the text may not be present in a summary because they are outnumbered by the sentences reflecting a more popular idea from the text. For example, if the first two-thirds of a text is devoted to one idea and the last one-third of the text is focused on another idea, then choosing sentences that are most similar to other sentences in the text will likely result in a summary consisting of sentences only from the first two-thirds of the text. To avoid that we first find clusters of related sentences and then pick representative sentences from each cluster to form a summary. By picking a representative sentence for each cluster we reduce potential redundant information in the summary and decrease the chances of missing some important ideas from the text.

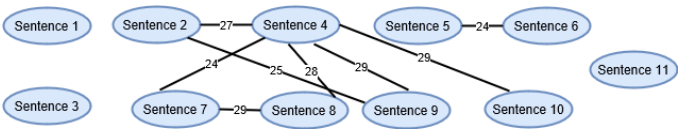

FIGURE 2 Sentence Similarity Graph for the document in Figure 3

FIGURE 3 Example of a document from DUC 2004 (Task 3 and 4)

Sentence #	Sentence
1	The American Defense Department (the Pentagon) reported today, Wednesday, that the bombing runs against Iraq are being especially carried out via cruise missiles launched from American aircraft carriers and B52 bombers.
2	The American television network quoted undisclosed sources at the American Defense Department today, Thursday, that the intense bombing operations on Iraq may cease starting the end of this week.
3	The humane organization, Caritas, announced today, Friday, in Germany quoting its branch in Iraq that several Iraqi hospitals, especially Saddam Hospital which is considered the biggest hospital in Iraq, were hit in the American-British bombing of the Iraqi capital.
4	German Foreign Minister Joschka Fischer demanded in Bonn today, Friday, a speedy stop to the bombing operations on Iraq.
5	An Iraq official reported today, Saturday, that 68 Iraqi civilians were killed as a result of the American and British bombing on Iraq and that their funerals were held today in Baghdad.
6	At least 68 Iraqi civilians were killed since Wednesday as a result of the American and British bombing on Iraq, as reported by an Iraqi official during their funerals today, Saturday, in Baghdad.
7	Representative Sultan Al-Shawi said during a funeral held for the victims, "They were all civilians killed during the savage bombing."
8	The funeral procession headed to the headquarters of the United Nations Development Program in Baghdad, where funeral goers held banners denouncing the Americans as "enemies of God."
9	The American television network "CNN" reported that officials at the American Defense Department (The Pentagon) will recommend to President Bill Clinton this evening, Saturday, halting the bombing on Iraq.
10	Iraq's Representative to the United Nations, Nizar Hamdoun, announced today, Sunday, that thousands of people were killed or injured during the four days of air bombardment against Iraq.
11	The Russian Foreign Ministry, in a first reaction to the American bombing in northern Iraq, stressed the need "to refrain from carrying out any actions that do not contribute to establishing suitable environments" to reach a political solution to the Iraqi problem, according to what the Russian agency Interfax reported.

In this research, we use the NetworkX software implementation[25] to find the cliques subgraphs. The software is based on the algorithm published by Bron and Kerbosch[26] as adapted by Tomita, Tanaka and Takahashi[27] and discussed in Cazals and Karande[28].

5 | ABSTRACTIVE CLIQUES SUMMARIZATION

After finding all cliques in the sentence similarity graph (a set of clusters of similar or related sentences), we aim to summarize the most salient theme of each cluster in a short single sentence called compression sentence. We build an abstractive summarization as a set of compression sentences of clusters. The obtained summary reflects the main ideas from a set of clusters without redundant information.

To generate a compression sentence for each clique in graph, Multi-Sentence Compression: Finding Shortest Paths in Word Graphs algorithm[29] is used. A Multi-Sentence Compression (MSC) generates a single compression sentence which would

preserve the important part of the content and be grammatically correct from the cluster. Finally, all compression sentences are used to build a summary.

6 | EXTRACTIVE CLIQUES SUMMARIZATION

The second way to select sentences from clusters to build a final summary is done by employing closeness centrality (or closeness) of a node that is a measure of centrality in a cluster. In the graph theory closeness centrality is calculated as the sum of the lengths of the shortest paths between the node and all other nodes in the cluster. Thus the more central a node is the closer it is to all other nodes[30,31,32]. We apply the same idea of graph theory closeness centrality in extractive cliques summarization method by defining the closeness centrality sentence in a cluster to be the sentence that has the minimum TED average in the cluster. The closeness centrality sentences from clusters are selected to be in a summary. Clusters may overlap, that is, a sentence may belong to more that one cluster. If a sentence is a closeness centrality sentence in more than one cluster, then we take this sentence as a representative of a smaller cluster. To represent a larger cluster we take the sentence that has the second minimum TED average in the larger cluster.

For example, in Figure 2 we have four clusters: {sentence 5, sentence 6}, {sentence 4, sentence 9, sentence 2 }, { sentence 4, sentence 8, sentence 7}, and { sentence 4, sentence 10 }. If a cluster consists of just two sentences, the, both sentences will have same TED average in the cluster, In such a case we pick one sentence at random to represent the cluster. For cluster {sentence 5, sentence 6}, sentence 5 is picked. For cluster { sentence 4, sentence 10 }, sentence 4 is picked. For cluster {sentence 4, sentence 9, sentence 2 }, sentence 2 is the closeness centrality sentence and it is selected as a representative sentence. For the cluster { sentence 4, sentence 8, sentence 7}, sentence 4 has minimum TED average for the cluster. In other words, sentence 4 is the closeness centrality sentence for two clusters. We keep it to represent the smaller cluster { sentence 4, sentence 10 }. To represent the larger cluster { sentence 4, sentence 8, sentence 7}, sentence 7 is selected since it has the second highest TED average for the cluster. Therefore, the resulting extractive summary for the example in Figure 3 consists of sentences 2, 4, 5, and 7.

7 | EVALUATION

The new approaches were evaluated on DUC 2004 (Task 3 and 4) data set[33]. The data set contains news documents in English together with human generated summaries for the documents. DUC 2004 data set was selected because it has been used in many previous related works, e.g.,[6,34,35].

To compare automatically generated summaries with human generated summaries, ROUGE Perl toolkit was used[36]. ROUGE works by comparing automatically produced summaries a set of reference summaries (typically human-produced). ROUGE is one of the standard ways to compute effectiveness of auto generated summaries. Many other researches used ROUGE to evaluate their summarization systems, e.g,[14,6]. ROUGE has different evaluation metrics (ROUGE-1, ROUGE-2, ROUGE-3, ROUGE-4, ROUGE-W, ROUGE-L, and ROUGE-SU) to determine the quality of a summary. ROUGE now generates three scores (precision, F-measure, and recall) for each evaluation.

We use Lex-Rank implementation from[37] to make a comparison with our research. Lex-Rank is the common bag-of-words extractive summarization approach that the other researchers used to make a comparison with their systems.

For each document from DUC 2004 (Task 3 and 4) data set, we generated system summaries using our proposed methods, and Lex-Rank method then compared the generated system summaries with human summaries from the data set DUC 2004. The average ROUGE scores for all documents are shown in Table 1. Extractive summarization method performs better in all ROUGE metrics evaluation measures. Among different scores of ROUGE, unigram-based ROUGE score (ROUGE-1) has been shown to agree with human judgments the most[38].

In this research, the lengths of abstractive and extractive cliques summaries depend on the number of generated clusters from a sentence similarity graph. Recall measure from ROUGE could not be used in such situations because it will always favor a longer summary. Therefore, we employ F score in all different evaluation metrics. F scores will provide an accurate comparison because the summary length is not enforced to reflect automatically produced summaries against a set of reference summaries according to[1]. Results in Table 1 show that both approaches perform better than Lex-Rank. Extractive cliques method performs slightly better than abstractive cliques method. The code is available for both, extractive and abstractive cliques methods, from the authors upon request.

ROUGE Average	Lex-Rank	Abstractive Cliques Method	Extractive Cliques Method
ROUGE-1 :	0.45	0.55	0.58
ROUGE-2 :	0.24	0.38	0.40
ROUGE-3 :	0.18	0.28	0.34
ROUGE-4 :	0.15	0.19	0.30
ROUGE-L :	0.39	0.51	0.52
ROUGE-W :	0.12	0.21	0.25
ROUGE-SU :	0.20	0.25	0.29

TABLE 1 F score ROUGE-Average Measures Generated by ROUGE

8 | CONCLUSION

In this research, summaries are generated by detecting clusters of related sentences in a document, finding representative sentences for each cluster, and adding them to the summary. Having the representative sentences for each cluster allows summaries to reflect different ideas from the document. Different clusters in a document are identified by finding cliques in a sentence similarity graph. Selecting representative sentences from each cluster could be done in two ways. The first way is to use Multi-Sentence Compression method for each cluster to generate a compression sentence that preserve the important part of the content in the cluster (abstractive cliques method). All generated compression sentences are used to build a final abstractive summary. The second way is to extract one sentence from each cluster that is the most similar to sentences in the cluster by selecting the closeness centrality sentence that is the sentence that has the minimum TED average according to the cluster. This results in an extractive summary (extractive cliques method). By selecting a representative sentence for each cluster we reduce potential redundant information in the summary and increase the chances of retaining the most important ideas from the text. The approaches are tested on DUC 2004 (Task 3 and 4) data set which contain news documents. ROUGE Perl toolkit is used to compare automatically produced summaries against a set of reference summaries from the data sets. Results show that the extractive cliques method performs slightly better than the abstractive cliques method. Both approaches perform better than Lex-Rank.

8.1 | Acknowledgements

We would also like to show our gratitude to the (Dr.Shuan Cooper, NMSU) for sharing his pearls of wisdom with us during the course of this research.

8.2 | Conflict of Interest

Authors have no conflict of interest relevant to this article.

References

1. Chopra S, Auli M, Rush AM. Abstractive Sentence Summarization with Attentive Recurrent Neural Networks. *Association for Computational Linguistics: Human Language Technologies* 2016: 93-98.

2. Fan W, Wallace L, Rich S, Zhang Z. Tapping Into the Power of Text Mining. *Communications of the ACM* 2006.

3. Yao J, Wan X, Xiao J. Recent Advances in Document Summarization. *Knowledge Information System* 2017.

4. Aliguliyev RM. A new sentence similarity measure and sentence based extractive technique for automatic text summarization. *Institute of Information Technology of National Academy of Sciences of Azerbaijan* 2009.

5. Mihalcea R, Tarau P. A language independent algorithm for single and multiple document summarization. *University of North Texas* 2005.

6. Erkan G, Radev DR. LexRank: Graph-based Lexical Centrality As Salience in Text Summarization. *Journal of Artificial Intelligence* 2004; 22(1): 457–479.

7. Wang D, Li T, Zhu S, Ding C. Multi-document Summarization via Sentence-level Semantic Analysis and Symmetric Matrix Factorization. In: SIGIR '08. ; 2008.

8. Shen C, Li T. Multi-document Summarization via the Minimum Dominating Set. *23rd International Conference on Computational Linguistics* 2010.

9. Sidorov G, Adorno HG, Markov I, Pinto D, Loya N. Computing Text Similarity using Tree Edit Distance. *5th World Conference on Soft Computing* 2015.

10. Ozates SB, Ozgur A, Radev D. Sentence Similarity based on Dependency Tree Kernels for Multi-document Summarization. *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC)* 2016.

11. Philip B. A Survey on Tree Edit Distance and Related Problems. *Theor. Comput. Sci.* 2005; 337(1-3).

12. Das D, Martins AF. A Survey on Automatic Text Summarization Single-Document Summarization. *Language* 2007; 4: 1–31.

13. Pei Y, Yin W, Fan Q, Huang L. A Supervised Aggregation Framework for Multi-Document Summarization. In: ; 2012.

14. Mihalcea R, Tarau P. TextRank: Bringing Order into Texts. *Association for Computational Linguistics* 2004.

15. Knight K, Marcu D. Summarization Beyond Sentence Extraction: A Probabilistic Approach to Sentence Compression. *Artif. Intell.* 2002; 139.

16. Schluter N, Sogaard A. Unsupervised extractive summarization via coverage maximization with syntactic and semantic concepts. *7th International Joint Conference on Natural Language Processing* 2015.

17. Fillmore CJ, Johnson CR, Petruck MR. Background to Framenet. *International Journal of Lexicography* 2003; 16(3): 235-250.

18. Brody S, Kantor PB. Automatic Assessment of Coverage Quality in Intelligence Reports. *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies* 2011.

19. Kaikhah K. Automatic text summarization with neural networks. *Texas State University* 2004.

20. Martins AFT, Almeida MSC. Priberam: A Turbo Semantic Parser with Second Order Features. *Proceedings of the 8th International Workshop on Semantic Evaluation* 2014.

21. Kumar YJ, Salim N. Automatic Multi Document Summarization Approaches. In: ; 2012.

22. Zhang K, Shasha D. Simple Fast Algorithms for the Editing Distance Between Trees and Related Problems. *SIAM J. Comput.* 1989; 18(6).

23. Augstent N. Tree edit distance using the Zhang Shasha algorithm. 2014. https://github.com/timtadh/zhang-shasha [Accessed: 10/8/2017].

24. Miller FP, Vandome AF, McBrewster J. *Levenshtein Distance: Information Theory, Computer Science, String (Computer Science), String Metric, Damerau?Levenshtein Distance, Spell Checker, Hamming Distance*. Alpha Press . 2009.

25. Hagberg AA, Schult DA, Swart PJ. Exploring Network Structure, Dynamics, and Function using NetworkX. 2008: 11 - 15.

26. Bron C, Kerbosch J. Algorithm 457: Finding All Cliques of an Undirected Graph. *Commun. ACM* 1973; 16(9): 575–577.

27. Tomita E, Tanaka A, Takahashi H. The Worst-Case Time Complexity for Generating All Maximal Cliques and Computational Experiments. *Theoretical Computer Science* 2006; 363: 28-42.

28. Cazals F, Karande C. A note on the problem of reporting maximal cliques. *Theoretical Computer Science* 2008; 407: 564-568.

29. Filippova K. Multi-Sentence Compression: Finding Shortest Paths in Word Graphs. *23rd International Conference on Computational Linguistics* 2010: 322-330.

30. Bavelas A. . Communication Patterns in Task-Oriented Groups. *Acoustical Society of America Journal* 1950.

31. Freeman LC. Centrality in social networks conceptual clarification. *Social networks* 1978.

32. Opsahl T, Agneessens F, Skvoretz J. Node centrality in weighted networks: Generalizing degree and shortest paths. *Social Networks* 2010.

33. DUC. Proceedings of Document Understanding Workshop (DUC 2004). 2004. Boston Park Plaza Hotel and Towers, Boston, USA, May 6-7, 2004.

34. Lin CY, Hovy E. From Single to Multi-document Summarization: A Prototype System and Its Evaluation. In: ACL '02. ; 2002.

35. Chali Y, Hasan SA. Query-focused multi-document summarization: Automatic data annotations and supervised learning approaches. *Natural Language Engineering* 2012; 18.

36. Ganesan K. ROUGE 2.0: Updated and Improved Measures for Evaluation of Summarization Tasks. 2015.

37. Belica M. sumy 0.4.1 is Module for automatic summarization of text documents and HTML pages.. 2014. http://pydoc.net/Python/sumy/0.4.1/ [Accessed: 4/12/2017].

38. Lin CY, Hovy E. Automatic Evaluation of Summaries Using N-gram Co-occurrence Statistics. In: NAACL '03. ; 2003.

How to cite this article: Nesreen Alsharman , and Inna Pivkina, (2020), Unsupervised Summarization via Cliques Algorithm