www.ijecs.in
International Journal Of Engineering And Computer Science ISSN: 2319-7242
Volume 5 Issue 12 Dec. 2016, Page No. 19432-19438

Text Summarization using topic modeling and cluster based MapReduce framework

N.Prameela
Assistant Professor
Department of Computer science & Engineering
Marri Laxman Reddy Institute of Technology,Hyderabad
prameelakotipalli@gmail.com

Abstract:*Document summarization provides an instrument for faster understanding the collection of text documents and has a number of real life applications. Semantic similarity and clustering can be utilized efficiently for generating effective summary of large text collections. Summarizing large volume of text is a challenging and time consuming problem particularly while considering the semantic similarity computation in summarization process. Summarization of text collection involves intensive text processing and computations to generate the summary. MapReduce is proven state of art technology for handling Big Data. In this paper, a novel framework based on MapReduce technology is proposed for summarizing large text collection. The proposed technique is designed using semantic similarity based clustering and topic modeling using Latent Dirichlet Allocation (LDA) for summarizing the large text collection over MapReduce framework. The summarization task is performed in four stages and provides a modular implementation of multiple documents summarization. The presented technique is evaluated in terms of scalability and various text summarization parameters namely, compression ratio, retention ratio, ROUGE and Pyramid score are also measured. The advantages of MapReduce framework are clearly visible from the Experiments and it is also demonstrated that MapReduce provides a faster implementation of summarizing large text collections and is a powerful tool in Big Text Data analysis.*

Keywords: Summarizing large text Semantic similarity Text clustering, Clustering based summarization Big Text Data analysis

Introduction

Text summarization is one of the important and challenging problems in text mining. It provides a number of benefits to users and a number of fruitful real life applications can be developed using text summarization. In text summarization a large collections of text documents are transformed to a reduced and compact text document, which represents the digest of the original text collections. A summarized document helps in understanding the gist of the large text collections quickly and also save a lot of time by avoiding reading of each individual document in a large text collection. Mathematically, text summarization is a function of converting large text information to small text information in such a manner that the small text information carries the overall picture of the large text collection as given in equation (1), where D represents the large text collection and d represents the summarized text document and the size of large text collection D is larger than the size of summarized document d.

$$f:D\rightarrow d||D|\ll|d|f:D\rightarrow d||D|\ll|d| \quad (1)$$

The algorithm performs the task of text summarization is called as text summarizer. The text summarizers are broadly categorized in two categories which are single-document summarizer and multi-document summarizers. In single-document summarizers, a single large text document is summarized to another single document summary, whereas in multi-document summarization, a set of text documents (multi documents) are summarized to a single document summary which represents the overall glimpse of the multiple documents.

Multi-document summarization is a technique used to summarize multiple text documents and is used for understanding large text document collections. Multi-document summarization generates a compact summary by extracting the relevant sentences from a collection of documents on the basis of document topics. In the recent years researchers have given much attention towards developing document summarization techniques. A number of summarization techniques are proposed to generate summaries by extracting the important sentences from the given collection of documents. Multi-document summarization is used for understanding and analysis of large document collections, the major source of these collections are news archives, blogs, tweets, web pages, research papers, web search results and technical reports available over the internet and other places. Some examples of the applications of the Multi-document summarization are analyzing the web search results for assisting users in further browsing [1], and generating summaries for news articles [2]. Document processing and summary generation in a large text document collection is computationally complex task and in the era of Big Data analytics where size of data collections is high there is need of algorithms for summarizing the large text collections rapidly. In this paper, a MapReduce framework based summarization method is proposed to generate the summaries from large text collections. Experimental results on UCI machine learning repository data sets reveal that the computational time for summarizing large text collections is drastically reduced using the MapReduce framework and MapReduce provides scalability for accommodating large text collections for summarizing. Performance measurement metric of summarization ROUGE and Pyramid scores are also gives acceptable values in summarizing the large text collections.

Single-document summarization is easy to handle since only one text document needs to be analyzed for summarization, whereas handling multi-document summarization is a complex and difficult task. It requires a number of (multiple) text documents to be analyzed for generating a compact and informative (meaningful) summary. As the number of documents increases in multi-document summarization, the summarizer gets more difficulties in performing the summarization. A summarizer is said to be good, if it contains more fruitful and relevant compact representation of large text collections. Considering semantic similar terms provide benefits in terms of generating more relevant summary but it is more compute intensive, since semantic terms will be generated and considered for creating summary from a large text collection. In this work the problems with multi-document text summarization are addressed with the help of latest technologies in text analytics. A multi-document summarizer is presented in this work with the help of semantic similarity based clustering over the popular distributed computing framework MapReduce.

Background and Literature review

MapReduce is a popular programming model for processing large data sets. It offers a number of benefits in handling large data sets such as scalability, flexibility, fault tolerance and numerous other advantages. In recent years a number of works are presented by researchers in field of Big Data analytics and large data sets processing. The challenges, opportunities, growth and advantages of MapReduce framework in handling the Big Data is presented in a number of studies [6–12]. MapReduce framework is widely used for processing and managing large data sets in a distributed cluster, which has been used for numerous applications such as, document clustering, access log analysis, generating search indexes and various other data analytical operations. A host of literature is present in recent years for performing Big Data clustering using MapReduce framework [3, 4, 13–16]. A modified K-means clustering algorithm based on MapReduce framework is proposed by Li *et al.* [17] to perform clustering on large data sets.

For analyzing large data and mining Big Data MapReduce framework is used in a number of works. Some of the work presented in this direction is web log analysis [18], matching for social media [19], design and implementation of Genetic Algorithms on Hadoop [20], social data analysis [21, 22], fuzzy rule based classification system [23], log joining [24], online feature selection [25], frequent item sets mining algorithm [26] and compressing semantic web statements [27].

Handling large text is a very difficult task particularly in knowledge discovery process. MapReduce framework is successfully utilized for a numbers of text processing tasks such as stemming [28], distribute the storage and computation loads in a cluster [29], text clustering [30], information extraction [31], storing and fetching unstructured data [32], document similarity algorithm [33], natural language processing [34] and pair-wise document similarity [35]. Summarizing large text collection is an interesting and challenging problem in text analytics. A numbers of approaches are suggested for handling large text for automatic text summarization [36, 37]. A MapReduce based distributed and parallel framework for summarizing large text is also presented by Hu and Zou [38].

A technique is proposed by Lai and Renals [39], for meeting summarization using prosodic features and augment lexical features. Features related to dialogue acts are discovered and utilized for meeting summarization. An unsupervised method for the automatic summarization of source code text is proposed by Fowkes *et al.* [40]. The proposed technique is utilized for code folding, which allows one to selectively hide blocks of code. A multi-sentence compression technique is proposed by Tzouridis *et al.* [41]. A parametric shortest path algorithm using word graphs is presented for multi-sentence compressions. A parametric way of edge weights is used for generating the desired summary. Parallel implementation of Latent Dirichlet Allocation namely, PLDA is proposed by Wang et al. [42]. The implementation is carried using MPI and MapReduce framework. It is demonstrated that PLDA can be applied to large, real-world applications and also achieves good scalability.

Methodology

The process of proposed multi-document summarization is shown in the Fig. 1 and Fig. 2. The summarization is performed in four major stages. The first stage is the document clustering stage where text clustering technique is applied on the multi document text collection to create the text document clusters. The purpose of this stage is to group the similar text document for making it ready for summarization and ensures that all the similar set of documents participates as a group in summarization process.

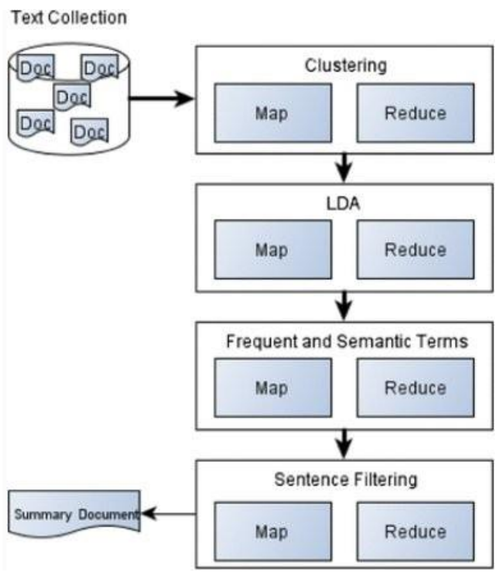


Fig. 1
Methodology of multi document summarization

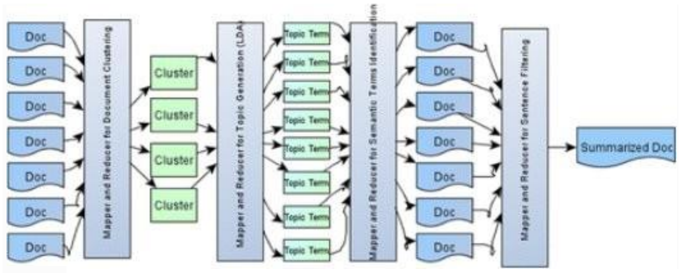

Doc
Mapper and Reducer for Document Clustering
Cluster
Mapper and Reducer for Topic Generation (LDA)
Topic Term
Mapper and Reducer for Semantic Terms Identification
Doc
Mapper and Reducer for Sentence Filtering
Summarized Doc

Fig. 2
Stages in MapReduce framework for multi document summarization

In the second stage Latent Dirichlet Allocation (LDA) topic modeling technique is applied on each individual text document cluster to generate the cluster topics and terms belonging to each cluster topic. In the third stage, global frequent terms are generated from the collection of multiple text documents. The process of frequent terms generation from the multiple text documents is shown in the Fig. 3. The topic terms generated for text clusters are taken as input to the summarizer which are shuffled and broadcasted to the mappers in Map-Reduce framework. The frequency of these topic terms is calculated and frequent terms are selected and semantic similar terms for these selected terms are computed using WordNet application programming interface (API) [43] which are collectively computed and taken as input to the next stage. WordNet is a popular API which provides an excellent way for generating semantic similar terms for a given term. In the last stage, sentence filtering is performed from each individual input text document on the basis of frequent and semantic similar terms generated from previous stage. For each document the sentences which are containing the frequent terms and semantic similar terms to the frequent terms are selected for participation in the summary document. Finally the approximate duplicate sentences are identified and removed from the summary report and final summary document is generated.

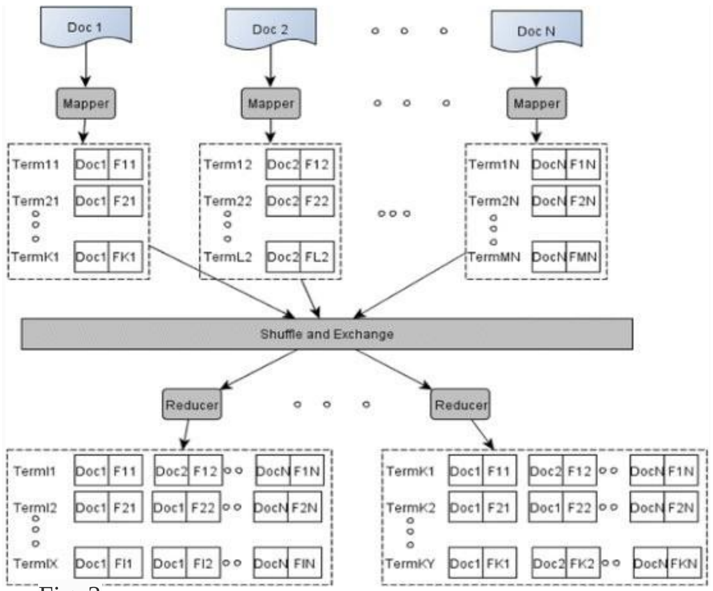


Fig. 3
Frequent terms counting from text collection using MapReduce framework

Latent dirichlet allocation

Latent Dirichlet Allocation (LDA) [44] is a popular topic modeling technique which models text documents as mixtures of latent topics, which are key concepts presented in the text. A topic model is a probability distribution technique over the collection of text documents, where each document is modeled as a combination of topics, which represents groups of words that tend to occur together. Each topic is modeled as a probability distribution φk over lexical terms. Each topic is presented as a vector of terms with the probability between 0 and 1. A document is modeled as a probability distribution over topics In LDA, the topic mixture is drawn from a conjugate Dirichlet prior that is the same for all documents. The topic modeling for text collection using LDA is performed in four steps. In the first step a multinomial distribution for each topic tt is selected from a Dirichlet distribution with parameter ββ. In second step for each document d, a multinomial distribution θbθb is selected from a Dirichlet distribution with parameter αα. In third step for each word w in document s a topic t from θbθb is selected. And finally in fourth step a word w from θtθt is selected to represent the topic for the text document. The probability of generating a corpus is given by the equation (4) [44].

∫∫∏t=1KP(θt|β)∏b=1NP(θb|α)(∏t=1Nb∑b=1KP(ti|θ)P(wi|t,∅)))dθd∅∫∫∏t=1KP(θt|β)∏b=1NP(θb|α)(∏t=1Nb∑
b=1KP(ti|θ)P(wi|t,∅l))dθd∅

LDA estimates the topic-term distribution and the document topic distribution from an unlabelled collection of documents using Dirichlet priors for the distributions over a fixed number of topics. Graphical representation of LDA topic modeling technique is presented in the Fig. 5.

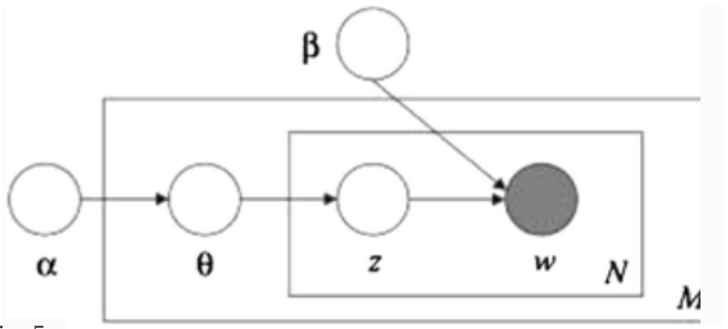


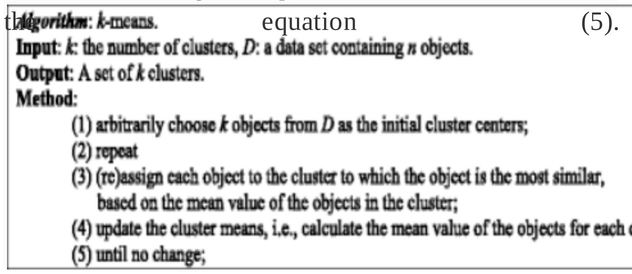
Fig. 5

Graphical representation of LDA process

K-means clustering algorithm

Clustering is a process of creating groups of similar objects. Clustering algorithms are categorized into five major categories namely, Partitioning techniques, Hierarchical techniques, Density Based techniques, Grid Based techniques and Model based techniques. Partitioning techniques are the simplest techniques which creates K number of disjoint partitions to create K number of clusters. These partitions are created using certain statistical measures like mean, median etc. K-means is a classical unsupervised learning algorithms used for clustering. It is a simple, low complexity and a very popular clustering algorithm.

The k-means algorithm [45] is a partitioning based clustering algorithm. It takes an input parameter, k i.e. the number of clusters to be formed, which partitions a set of n objects to generate the k clusters. The algorithm works in three steps. In the first step, k number of the objects is selected randomly, each of which represents the initial mean or center of the cluster. In the second step, the remaining objects are assigned to the cluster with minimum distance from cluster center or mean. In the third step, the new mean for each cluster is computed and the process iterates until the criterion function converges. The algorithm is presented in the Fig. 6 and the performance of k-means is measured using the square-error function defined in the equation (5).



```
Algorithm: k-means.
Input: k: the number of clusters, D: a data set containing n objects.
Output: A set of k clusters.
Method:
    (1) arbitrarily choose k objects from D as the initial cluster centers;
    (2) repeat
    (3) (re)assign each object to the cluster to which the object is the most similar,
        based on the mean value of the objects in the cluster;
    (4) update the cluster means, i.e., calculate the mean value of the objects for each cluster;
    (5) until no change;
```

Fig. 6

K-Means clustering algorithm
E=∑i=1k∑p∈Ci| | p−mi| | 2E=∑i=1k∑p∈Ci|p−mi|2

Where E is the sum of the square error, p is the point in space representing a given object and mi is the mean of cluster Ci. This criterion tries to make the resulting k clusters as compact and as separate as possible. The algorithm is consisting of five major steps which are summarizes as given below.

Result analysis

The scalability of the proposed work in MapReduce framework up to four nodes is shown in the Fig.7. The scalability is calculated using different nodes and different numbers of text document reports for generating the summary using the proposed MapReducer based summarizer. Scalability tends to increase in proportion to the number of text documents with maximum numbers of nodes. The scalability of the proposed work is also supported by the Amdahl’s law. As per the Amdahl’s law [57], the optimal speedup possible for a computation is limited by its sequential components. If *f* is the fraction of the computational task then the theoretically maximum possible speedup for N parallel resources is SN=1(f+1−fN)SN=1(f+1−fN)

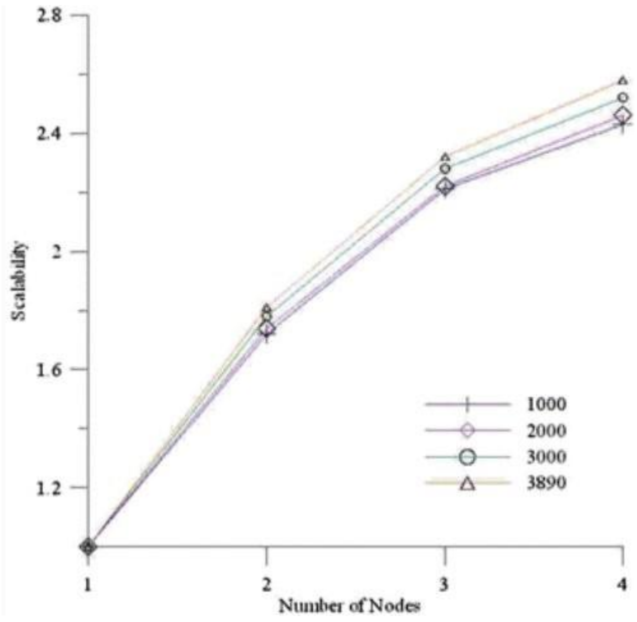


Fig. 7

Scalability of MapReducer based summarizer
The time required for generating summary from the text collection of different size and for different nodes in MapReduce

framework is also shown in the Fig.8. Time to compute the summary tends to decrease with increase in number of nodes. As the nodes increases the computation time tends to linear and up to four nodes it becomes just linear in proportionate to the number of text documents participating in summary. When the number of nodes are changed from one to two the computational time downfall in exponential manners and when the nodes reaches up to four the computational time becomes linear with proportionate to the number of text document collection.

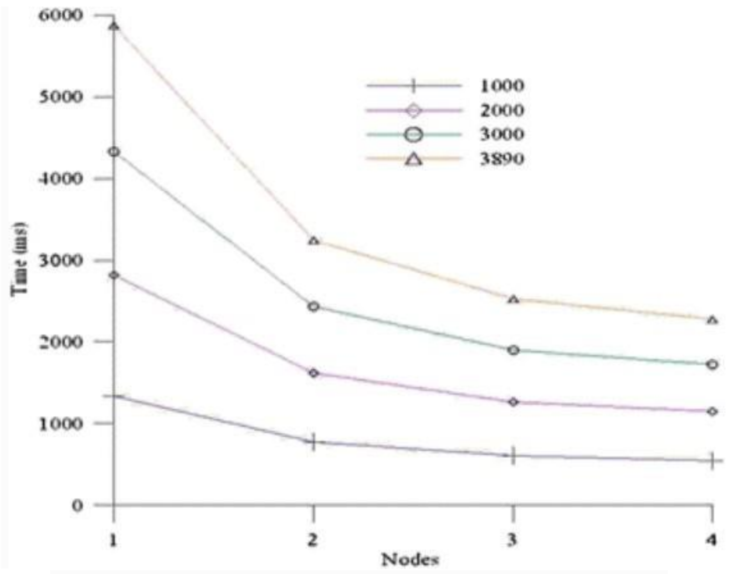


Fig. 8
Time in ms for summarizing the text reports

The performance parameters of proposed summarizers i.e. compression ratio, retention ratio, ROUGE and Pyramid scores are evaluated for three different scenarios. The summarizers are evaluated for the following three cases:

∑ Case 1: Summarization without performing clustering and semantic similarity.
∑ Case 2: Summarization with clustering but without considering semantic similarity.
∑ Case 3: Summarization by considering both clustering and semantic similarity.

The compression ratio for different number of nodes for the three different scenarios is shown in the Fig.9. Similarly, the retention for the possible three cases is presented in the Fig. 10. It is apparent from the graphs that considering the semantic similarity (Case 3) will definitely give better results for generating effective and meaningful summary of text document collections. These results clearly indicates that semantic similarity along with the clustering gives better summarization results as compared to the summarization without semantic similarity and clustering. Semantic similarity provides meaningful grouping of similar text segments as summarization content units for generating summary of the text collections. Semantic similarity ensures better chunking of meaningful text groups as compared to the plain clustering of text documents (Case 2). Semantic similarity along with clustering provides a mechanism of participation of the different summarization content units from the different groups of text documents.

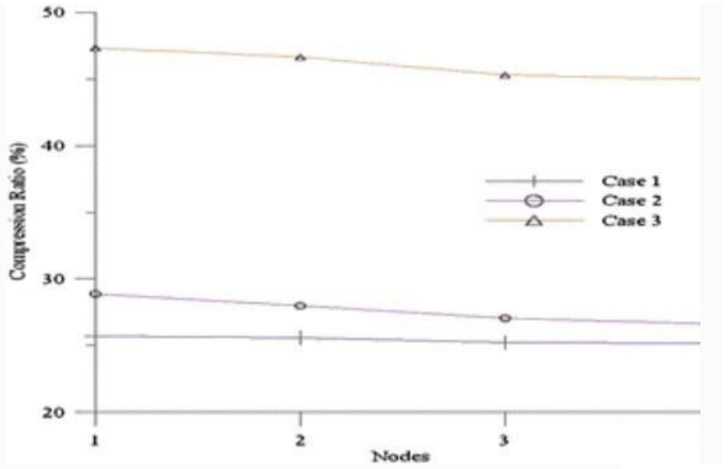


Fig.9

Compression ratio for different cases

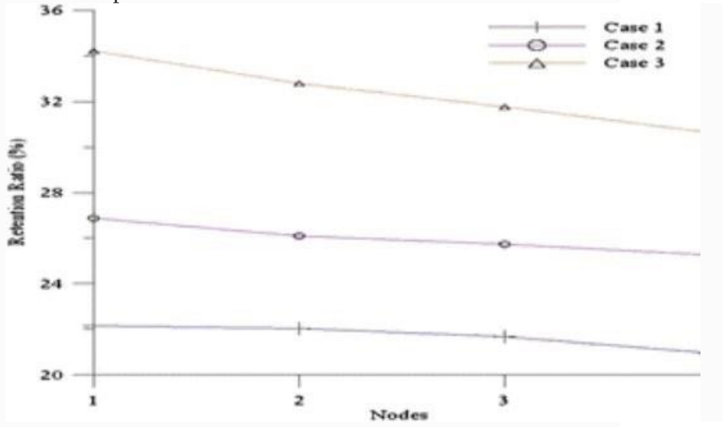
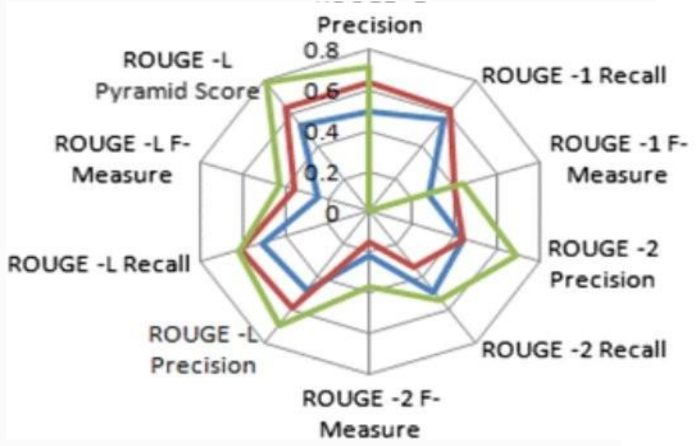


Fig. 10
Retention ratio for different cases

The rouge and pyramid scores of the presented summarization approaches are tabulated for the three different cases in the Table 1. ROUGE unigram and bigram scores are calculated for the presented work. ROUGE unigram gives better results for summarization as compared to the ROUGE bigram approach. The pyramid score gives a normalized score in the range of 0 to 1 in order to evaluate the summary.

As expected from the results the ROUGE and Pyramid scores are found higher for the case III than the other two cases. Case III consider both the textual similarity (using clustering) and semantic similarity which makes sure that best summarization content units participate in the summary generation. Case II gives better results than the Case I results, in other words summarization using clustering gives better summarization results as compared to the summarization performed without performing clustering. It indicates that summarization performed on the clustered text documents is more accurate since similar text information is grouped within the same clusters.

Higher pyramid scores indicating that relatively more of the content is as highly weighted as possible. High pyramid score reflects the greater likelihood that more SCUs (Summarization Content Units) in the summary appear in the pyramid [53]. Just like the ROUGE score, maximum pyramid score is achieved for the case III, where both semantic and textual similarity (clustering) is considered for summarizing the text collections. It is also shown that clustering (grouping the similar text segments) provides better summarization in context to the summarization performed with non-clustered text collections. Clustering provides better summarization units (text segments) for summarizing the text collections. It is also clear that clustering along with the semantic similarity provides better summarization content units for generating summary from the text collections. To better demonstrate the results of the different cases, Fig:11 visually illustrate the comparison. Figure 11 demonstrates spider chart showing the comparisons of the three different cases, it is clearly visible from the chart that the values of performance parameters for case-III (considering both the clustering with semantic similarity) gives better results as compared to the rest of the two cases.



Comparison of ROUGE and pyramid scores

Conclusion and Future enhancements

A multi-document text summarizer based on MapReduce framework is presented in this work. Experiments are carried using up to four nodes in MapReduce framework for a large text collection and the summarization performance parameters compression ratio, retention ratio and computation timings are evaluated for a large text collection. It is also shown experimentally that MapReduce framework provides better scalability and reduced time complexity while considering large number of text documents for summarization. Three possible cases of summarizing the multiple documents are also studied comparatively. It is shown that effective summarization is performed when both clustering and semantic similarity are considered. Considering semantic similarity gives better retention ratio, ROUGE and pyramid scores for summary. Future work in this direction can be providing the support for multi lingual text summarization over the MapReduce framework in order to facilitate the summary generation from the text document collections available in different languages.

References

[1] Turpin A, Tsegay Y, Hawking D, Williams H (2007) Fast generation of result snippets in web search. Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval, Amsterdam, Canada, pp 127–134Google Scholar

[2] Sampath G, Martinovic M (2002) Proceedings of the 6th International Conference on Applications of Natural Language to Information Systems, NLDB 2002, 2002nd edn. Proceedings of the 6th International Conference on Applications of Natural Language to Information Systems, Stockholm, Sweden, pp 208–212Google Scholar

[3] Dean J, Ghemawat S (2004) MapReduce: Simplified data processing on large clusters. Proc. of the 6th Symposium on Operating System Design and Implementation (OSDI 2004). San Francisco, California, pp 137–150Google Scholar

[4] Dean J, Ghemawat S (2010) MapReduce: A flexible data processing tool. Commun ACM 53(1):72–77CrossRefGoogle Scholar

[5] Borthakur, D. (2007) The hadoop distributed file system: Architecture and design. Hadoop Project Website (Availableonlineat- https://hadoop.apache.org/docs/r1.2.1/hdfs_design.pdf). p 1–14 Accessed 15 April 2014

[6] Steve L (2012) The Age of Big Data. Big Data's Impact in the World, New York, USA, pp 1–5Google Scholar

[7] Russom P (2011) Big Data Analytics. TDWI Research Report, US, pp 1–38Google Scholar

ACM, New York, USA, pp 681–689CrossRefGoogle Scholar

[11] Kolb L, Thor A, Rahm E (2013) Don't Match Twice: Redundancy-free Similarity Computation with MapReduce. Proc. of the Second Workshop on Data Analytics in the Cloud, ACM, New York, USA, pp 1–5Google Scholar

Management Revolution. Harv Bus Rev 90(10):60–68Google Scholar

[9] Shim K (2013) MapReduce Algorithms for Big Data Analysis. Databases in Networked Information Systems, Springer, Berlin, Heidelberg, Germany, pp 44–48Google Scholar [10] Ene A, Im S, Moseley B (2011) Fast Clustering using MapReduce. Proc. of the 17th ACM SIGKDD international conference on Knowledge discovery and data minin