Nagwani *Journal of Big Data* (2015) 2:6
DOI 10.1186/s40537-015-0020-5

Journal of Big Data
a SpringerOpen Journal

RESEARCH Open Access

Summarizing large text collection using topic modeling and clustering based on MapReduce framework

N K Nagwani

Correspondence:
nknagwani.cs@nitrr.ac.in
Department of Computer Science and Engineering, National Institute of Technology Raipur, Raipur 492010, India

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.

$$f : D \rightarrow d \;||D| \ll |d| \tag{1}$$

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.

MapReduce [3, 4] is a programming model for implementation of distributed computations of high volume or big data and an execution framework for large-scale data processing on clusters of servers. One of the MapReduce open-source implementation is Hadoop [5] by Apache Foundation Projects. The mappers and reducers are mathematically presented in the equation (2) and equation (3) respectively. The (k_1, v_1), (k_2, v_2) and (k_3, v_3) represents the key-value pairs for *map* and *reduce* functions.

$$map : (k_1, v_1) \rightarrow [(k_2, v_2)] \tag{2}$$

$$reduce : (k_2, [v_2]) \rightarrow [(k_3, v_3)] \tag{3}$$

The mapper is applied to each input key-value pair to generate an arbitrary number of intermediate key-value pairs. The reducer is applied to all values associated with the same intermediate key to generate output key-value pairs. Mappers and reducers are objects that implement the Map and Reduce methods, respectively.

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 pairwise 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.

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

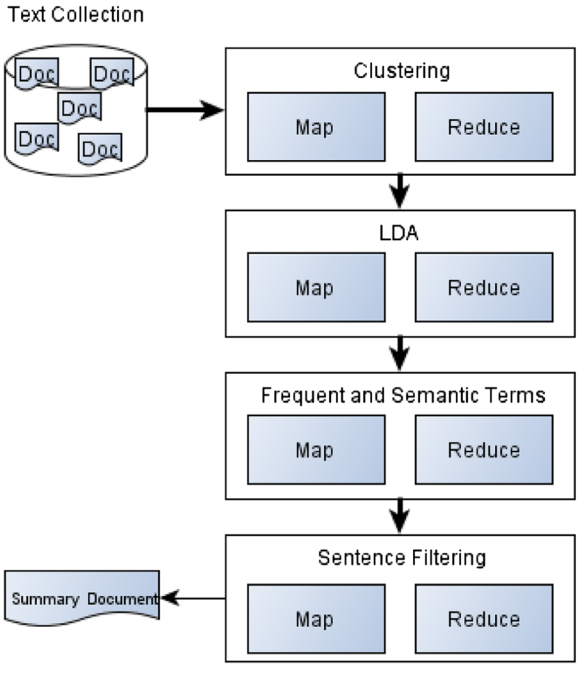

Fig. 1 Methodology of multi document summarization

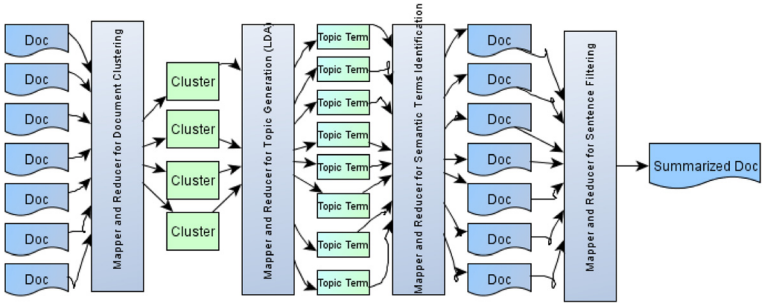

Fig. 2 Stages in MapReduce framework for multi document summarization

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

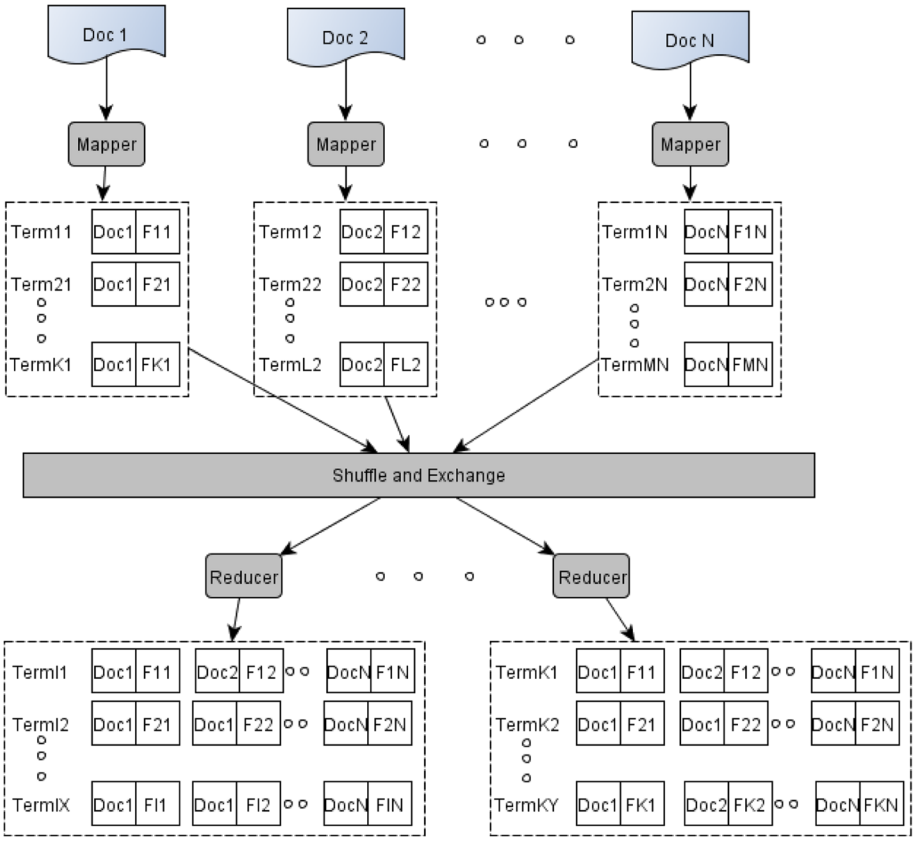

Fig. 3 Frequent terms counting from text collection using MapReduce framework

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.

Figure 4 illustrates the hypothetical process that is modeled for generating summary from the multiple text documents using clustering technique. In order to perform clustering of the text documents all the documents D_i are brought together into one data set, D. Then the K-Means clustering algorithm is applied to perform the clustering of on the whole document set. K-Clusters are generated. The set of clusters $C = \{C_1, C_2, \ldots, C_K\}$ where $C_k (k = 1, 2, \ldots, K)$ are consisting of group of similar documents belonging to a particular cluster C_i. Clustering ensures that similar set of text documents are group together and logically represents a theme (summarization unit) for effective summarization. The impact of clustering for summarization of large text collection is also demonstrated in this work. It is shown that summarization with clustering gives better summarization performance as compared to the summarization without clustering.

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 ϕ_k 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 θ_t distribution for each topic t is selected from a Dirichlet distribution with parameter β. In second step for each document d, a multinomial distribution θ_b is selected from a Dirichlet distribution with parameter α. In third step for each word w in document s a topic t from θ_b is selected. And finally in fourth step a word w from θ_t is

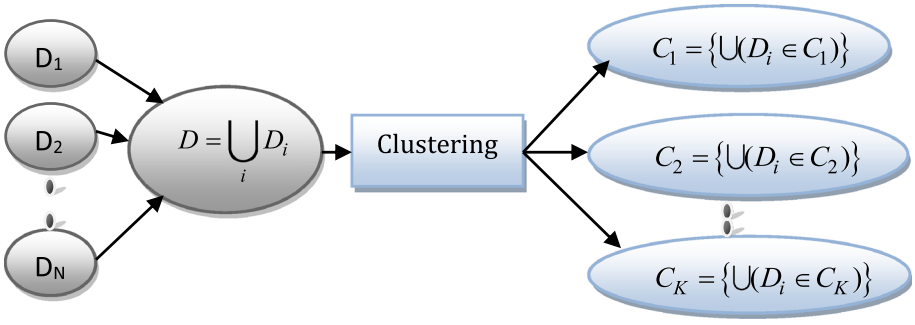

Fig. 4 Document clustering using clustering algorithm

selected to represent the topic for the text document. The probability of generating a corpus is given by the equation (4) [44].

$$\iint \prod_{t=1}^{K} P(\theta_t|\beta) \prod_{b=1}^{N} P(\theta_b|\alpha) \left(\prod_{t=1}^{Nb} \sum_{b=1}^{K} P(t_i|\theta) P(w_i|t, \varnothing) \right) d\theta d\varnothing \tag{4}$$

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.

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).

$$E = \sum_{i=1}^{k} \sum_{p \in C_i} |p - m_i|^2 \tag{5}$$

Where E is the sum of the square error, p is the point in space representing a given object and m_i is the mean of cluster C_i. 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.

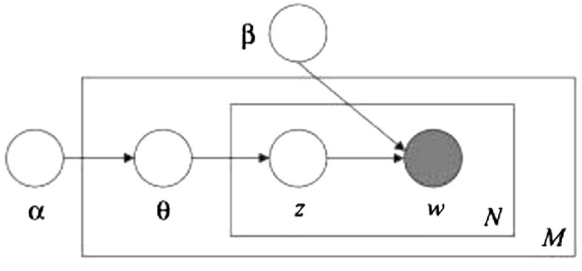

Fig. 5 Graphical representation of LDA process

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

The algorithm

Based on the methodology discussed in the previous section the algorithm for proposed multi document summarization using semantic similarity based clustering technique is presented in this section. The algorithm is logically divided in four major stages; the algorithm for each stage is explained in this section. In the first stage of document summarization, the document clustering is performed using K-means clustering algorithm on MapReduce framework. Mapper is responsible for part of documents and part of k centers. For each document, it finds closest of known centers and produces the output key as point, value identifies center and distance. Reducer takes minimum distance center and produces output key identifies center, value is document. A successive phase averages points in each center. The mapper and reducer for K-means algorithm is presented in the Fig. 7.

After creating the text document clustering, the document belonging to clusters are retrieved and text information present is each document is collected in aggregate. The topic modeling technique is then applied on collective information to generate the topics from each text document clusters. LDA (Latent Dirichlet Allocation) technique is used in this work for generating topics from each document cluster. The mapper and reducer for topic terms generation from document clusters is shown in the Fig. 8.

In the third stage, semantic similar terms are computed for each topic term generated in previous stage. WordNet Java API [43] is used to generate the list of semantic similar terms. The semantic similar terms are generated over the MapReduce framework and the generated semantic terms are added to the vector. Semantic similar term finding is an intensive computing operation. It requires going through with the vocabulary and

Mapper
a. Initialize the cluster centers randomly and Read it into memory from a sequence file
$$\{C_{Mean1}, C_{Mean2}, ..., C_{MeanK}\} \leftarrow Random$$
b. Iterate each cluster center for each input key-value pair. Computer Center $\forall \langle K, V \rangle$
c. Calculate distances and assign the nearest center with the lowest distance
$$C_i \leftarrow Min\left(d\left(key_1, C_i\right), ..., d\left(key_N, C_i\right)\right)$$
d. Update the cluster center with its vector to the file system. $\{\langle K_{i1}, V_{i1}\rangle, ..., \langle K_{iM}, V_{iM}\rangle\} \in C_i$

Reducer
a. Iterate each value vector and calculate the mean.
b. Update the new center from calculate mean. $C_i \leftarrow Avg\left(d_{i1}, ..., d_{ij}\right)$
c. Check whether the cluster center is same as new center. $if\left(C_{i-Old} = C_{i-New}\right)$ then stop else *GoTo* next step.
d. If it they are not equal, increment an update counter.

Run the Mapper and Reducer until the Clusters Converges.

Fig. 7 Mapper and reducer for document clustering

Mapper

For each cluster get the documents it contains and extract the text collection from these documents.

For each cluster $C_i \in \{C_1, C_2, ..., C_N\}$

Extract the documents in C_i as $\{D_{i1}, D_{i2}, ..., D_{iM}\}$

For each document extract and merge the text from the text collection.

$Text = Text \cup \{Text_{ij} \in D_{ij}\}$

Apply LDA topic modeling to these collection and get the list of topics for the cluster C_i as $T_i = \{T_{i1}, T_{i2}, ..., T_{iK}\}$.

Reducer

Integrate the topics of all the clusters

For each cluster $C_i \in \{C_1, C_2, ..., C_N\}$

Extract the topics discovered by LDA in the documents in C_i as $T_i = \{T_{i1}, T_{i2}, ..., T_{iK}\}$

For each document extract the text and computer the text collection.

$Topics = Topics \cup \{T_{ij} \in T_i\}$

Fig. 8 Mapper and reducer for LDA topic generation from document cluster

synonyms data for the given term in the hierarchy of semantic relationship. MapReduce framework is utilized efficiently for handling this operation. The Mapper computes the semantic similar terms for each topic term generated by the document cluster and reducer aggregate these terms and counts the frequencies of these terms (topic terms and semantic similar terms of topic terms) aggregately. The mapper and reducer for semantic terms generation from cluster topic terms is presented in the Fig. 9.

Then the terms are arranged in the descending order of frequency and top N topic terms (including the semantic similar terms) are selected. These filtered terms are called as semantic similar frequent terms available in the document collection using the method *ComputeSemanticSimilar*(T_i) . The algorithm counts the number of occurrences of every word in a text collection. Input key-values pairs take the form of (document id, doc) pairs stored on the distributed file system. The key parameter is a unique identifier for the document, and the value parameter is the text of the document itself. The Mapper takes key-value pair as input, generates tokens from the document, and emits an intermediate key-value pair for every word. The MapReduce execution makes sure that all values associated with the same key are brought together in the reducer. The final output of the algorithm is written to the distributed file system, one file per reducer.

Mapper

For each topic term in topic list $\{T_1, T_2, ..., T_N\}$

Get the semantic similar terms

$TS_i = ComputeSemanticSimilar(T_i)$

// Pass the term T_i in WordNet API and extract the semantic similar Terms in the set TS_i.

For all term t $t \in TS_i$ present in the document D do

Emit(term t; count 1)

Reducer

For each term t, counts $[c1, c2, ...]$

Initialize the sum of term frequency as 0.

For all count c $\in$ counts $[c1, c2, ...]$ do

Update sum by adding count, i.e. $sum+ = c$

Emit(term t; count sum)

Fig. 9 Mapper and reducer for semantic terms generation from cluster topic terms

In the last stage, the original text document collection is distributed over the Mappers and using parsing techniques, sentences are extracted from individual document by the Mappers. The sentences which are consisting of the frequent terms and its semantic similar terms are filtered from the original text collection and added to the summary document (in other words the filtered terms participates in the summary document). The final summary is generated after traversing all the documents in the document collections. The mapper and reducer for document filtering is presented in the Fig. 10. The performance parameters for summarization process are then evaluated to measure the performance of proposed summarizer.

Experiments and result analysis

The implementation is carried using the Java based open source technologies. The LDA implementation is performed using MALLET API [46], and the MapReduce implementation is performed using Hadoop API [5]. A textual corpus of around 4000 legal cases for automatic summarization is selected for performing the experiments, the dataset is available on UCI machine learning repository[A]. The dataset contains Australian legal cases from the Federal Court of Australia (FCA) all files from the year 2006, 2007, 2008 and 2009. The dataset is earlier used in the work of Galgani *et al.* [47, 48]. The experiments are performed over the dual core processor based systems with CPU speed 2.33 GHz, 2 GB of RAM (Random Access Memory), and 1.333 GHz bus clock in Windows XP operating system. The systems (up to four nodes) are interconnected over a 100 Mbps LAN (Local Area Network).

Summarization techniques are categorized into two major categories extractive or abstractive. Extractive summarization assigns a filter and extracts the sentences with highest matching criteria to form the summaries. Abstractive summarization, on the other hand, uses certain degree of understanding of the content expressed in the original documents and creates the summaries based on information fusion. Like most researchers in this field, the extractive summarization framework in used in this work.

[A]https://archive.ics.uci.edu/ml/datasets/Legal+Case+Reports

Zipf's power distribution law [49, 50] states that most of users use limited number of words (terms) frequently in the documents. The existing studies also shows that less number of dominating terms participates more in knowledge discovery tasks [45]. In this work 15 frequent terms from topic terms of document clusters are selected and its semantic similar terms are considered for document summarization task. As per the

Mapper

Select one document at a time from the document collection

For each Document $D \in \{D_1, D_2, ...\}$

Extract the sentences from Document D as $\{S_{i1}, S_{i2}, ...\}$ using parsing.

If S_{ik} contains the terms present in TS_i

Filter the sentences S_{ik} containing the terms and add it to Vector.

$Vector = Vector \cup \{S_K\}$

Reducer

Integrate all the filtered sentences and Produce a single document presenting the

$Summary = Summary \cup \{S_{ik}\}$

Fig. 10 Mapper and reducer for document filtering

text mining best practices, more than 15 terms will minimally improve the performance of summarizer and less than 15 terms will drastically reduce the performance of the summarizer.

Three major requirements for multi-document summarization [51] are clustering, coverage and anti-redundancy. Clustering is the ability to cluster similar documents and passages to find related information, coverage is the ability to find and extract the main points across documents and anti-redundancy is the ability to minimize redundancy between passages in the summary. Clustering requirement is achieved with the help of K-Means algorithm to group the similar documents with the common themes and also is the part of proposed technique. Coverage and anti-redundancy is achieved with the help of sentence filtering while generating the final summary.

Summarization evaluation

Text summarization process is majorly evaluated using performance parameters namely, Compression Ratio (CR), Retention Ratio (RR), ROUGE score and Pyramid score.

Compression and retention ratio

The Compression Ratio (CR) is the ratio of size of the summarized text document to the total size of the original text documents. Retention Ratio (RR) is the ratio of the information available in the summarized document to the information available in the original text collections. The expression of calculating the CR and RR are given in the equation (6) and equation (7) respectively.

$$CR = \frac{|d|}{|D|} \tag{6}$$

$$RR = \frac{Info(d)}{Info(D)} \tag{7}$$

Where $|d|$ represents the size of the summarized text is document and $|D|$ is the total size of the original text collection. $Info(d)$ represents the information available in the summarized text document and $Info(D)$ is the information present in the original text collection.

Rouge and pyramid score

Rouge score or rouge-N score [52] and pyramid score [53, 54] are the two major parameters used for evaluating the summarization tasks and are used in a number of studies. Qazvinian and Radev [55] have used these parameters in summarizing scientific papers using citation summary networks. Galgani *et al.* [48] used these parameters while building an incremental summarizer.

ROUGE stands for Recall-Oriented Understudy for Gisting Evaluation, it includes several measures to quantitatively compare system-generated to human-generated summaries, counting the number of overlapping n-grams of various lengths, word pairs and word sequences between the summaries. A standard ROUGE evaluation would compare the whole block of catchphrases to the whole block of extracted sentences. ROUGE-N is an n-gram recall between a candidate summary and a set of reference summaries. ROUGE measure includes several automatic evaluations such as ROUGE-N,

ROUGE-L, ROUGE-W and ROUGE-SU. ROUGE-N is an n-gram recall computed as given in the Eq. (8).

$$ROUGE\text{–}N = \frac{\sum_{Se\{ReferenceSummaries\}} \sum_{gram_n eS} Count_{match}(gram_n)}{\sum_{Se\{ReferenceSummaries\}} \sum_{gram_n eS} Count(gram_n)} \tag{8}$$

where n represents the length of the n-gram, and ref represents the reference summaries. $Count_{match}(gram_n)$ represents the number of n-grams co-occurring in a candidate and the reference summaries, and Count ($gram_n$) represents the number of n-grams in the reference summaries. ROUGE-L measure uses the longest common subsequence (LCS), ROUGE-W measure is derived using weighted LCS and ROUGE-SU measure uses skip-bigram plus unigram for measuring the generated summaries [56]. In this work the average precision, recall and F-measure scores generated by ROUGE-1, ROUGE-2, and ROUGE-L are used to measure the performance of the summaries and to compare the presented algorithm over the MapReduce framework.

The Pyramid evaluation combines both a precision measure (as the score is a function of the size of the summary) and of a recall measure (as the score is also a function of the weights of the optimal SCUs or Summarization Content Units). The score given by the pyramid method for a summary is a ratio of the sum of the weights of its facts to the sum of the weights of an optimal summary. The pyramid score ranges from 0 to 1, and high scores show the summary content contain more heavily weighted facts. The *n* tiers pyramid refers to "pyramid of order *n*". Given a pyramid of order *n*, the optimal summary content can be predicted which contain all the SCUs from the top tier and then from the next tier and so on. In other words an SCU from tier ($n - 1$) should not be expressed until the SCUs in tier *n* have been expressed. The score assigned is a ratio of the sum of the weights of its SCUs to the sum of the weights of the optimal summary with the same number of SCUs. It ranges from 0 to 1, with higher scores indicating that relatively more of the content is as highly weighted as possible [53, 54].

The exact formula we use is computed as follows [53, 54]. Suppose the pyramid has n tiers, T_i, with tier T_n on top and T1 on the bottom. The weights of SCUs in tier Ti will be i. Let $|T_i|$ denote the number of SCUs in the tier T_i. Let D_i be the number of SCUs in the summary that appear in T_i. SCUs in a summary that do not appear in a pyramid are assigned weight zero. The total SCU weight D is $D = \sum_{1}^{n} i \times D_i$ .The optimal content score for a summary with X SCUs is:

$$Max = \sum_{i=j+1}^{n} i \times |T_i| + j \times \left(X - \sum_{i=j+1}^{n} |T_i| \right) \text{ Where } j = \max_i \left(\sum_{t=i}^{n} |T_t| \geq X \right) \tag{9}$$

In the equation above, j is equal to the index of the lowest tier an optimally informative summary will draw from. This tier is the first one top down such that the sum of its cardinality and the cardinalities of tiers above it is greater than or equal to X (summary size in SCUs). For example, if X is less than the cardinality of the most highly weighted tier, then j = n and Max is simply Xxn (the product of X and the highest weighting factor). Then the pyramid score *P* is the ratio of D to Max. Because P compares the actual distribution of SCUs to an empirically determined weighting, it

provides a direct correlate of the way human summarizers select information from source texts.

Result analysis

The scalability of the proposed work in MapReduce framework up to four nodes is shown in the Fig. 11. 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 $S_N = \frac{1}{\left(f+\frac{1-f}{N}\right)}$.

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. 12. 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.

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.

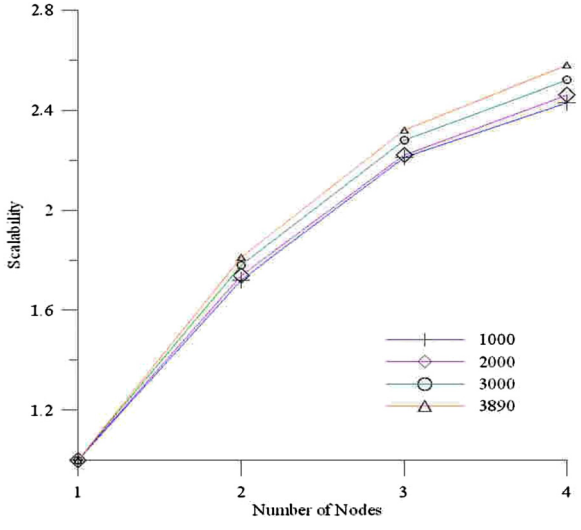

Fig. 11 Scalability of MapReducer based summarizer

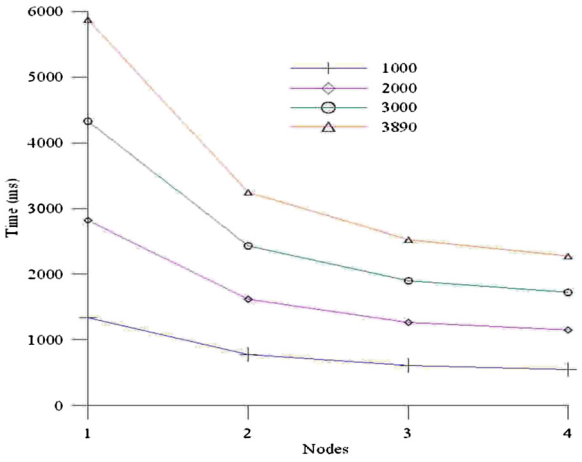

Fig. 12 Time in ms for summarizing the text reports

- 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. 13. Similarly, the retention for the possible three cases is presented in the Fig. 14. 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.

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

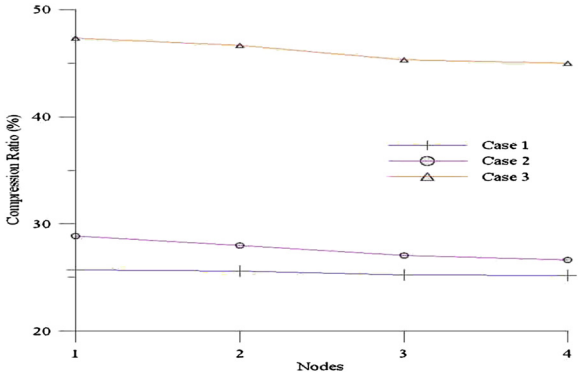

Fig. 13 Compression ratio for different cases

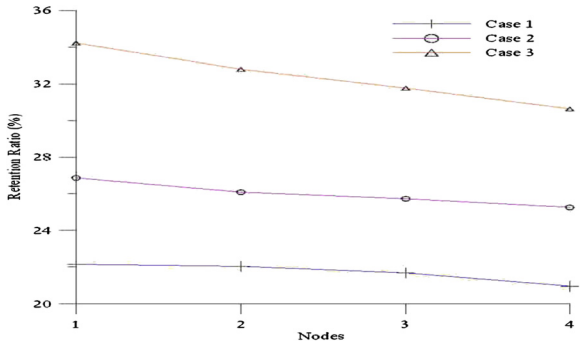

Fig. 14 Retention ratio for different cases

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. 15 visually illustrate the comparison. Figure 15 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.

Table 1 ROUGE and pyramid scores for the three different cases

	ROUGE −1			ROUGE −2			ROUGE −L			
	Precision	Recall	F	Precision	Recall	F	Precision	Recall	F	Pyramid score
Case – I	0.488	0.566	0.276	0.438	0.487	0.213	0.473	0.508	0.244	0.528
Case – II	0.637	0.616	0.392	0.434	0.332	0.144	0.590	0.610	0.360	0.634
Case – III	0.710	0. 620	0.440	0.685	0.536	0.367	0.691	0.622	0.428	0.780

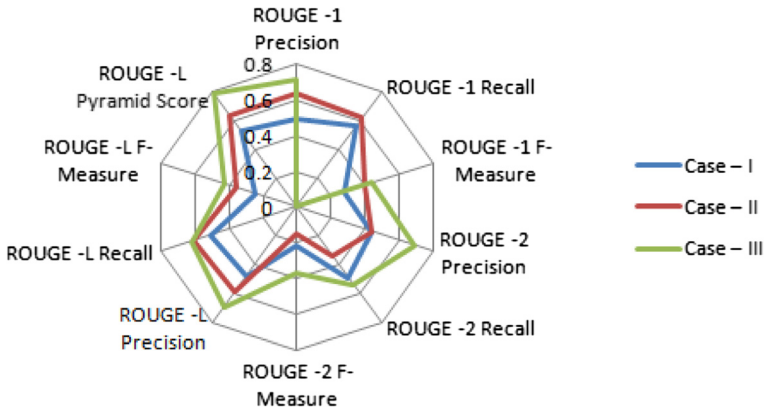

Fig. 15 Comparison of ROUGE and pyramid scores

Conclusions 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.

Competing interests

The author declares that he has no competing interests.

Authors' contribution

NKN is the sole author of this manuscript. NKN developed the analytical model presented in this work and created figure of methodology of the paper. NKN implemented the proposed model and prepared the graphs for indicating the results. NKN wrote the text of the paper, read and approved the final manuscript.

Authors' information

Naresh Kumar Nagwani, Ph.D.,
Assistant Professor,
Computer Science & Engineering,
National Institute of Technology Raipur, India.
Email:- nknagwani.cs@nitrr.ac.in

Acknowledgments

The authors want to thank National Institute of Technology Raipur, India for providing infrastructure and facilities to carry out this research work.

Received: 2 April 2015 Accepted: 17 June 2015
Published online: 26 June 2015

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 127–134

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 208–212
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 137–150
4. Dean J, Ghemawat S (2010) MapReduce: A flexible data processing tool. Commun ACM 53(1):72–77
5. Borthakur, D. (2007) The hadoop distributed file system: Architecture and design. Hadoop Project Website (Available online at - 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 1–5
7. Russom P (2011) Big Data Analytics. TDWI Research Report, US, pp 1–38
8. McAfee A, Brynjolfsson E (2012) Big Data: The Management Revolution. Harv Bus Rev 90(10):60–68
9. Li F, Ooi BC, Özsu MT, Wu S (2013) Distributed Data Management Using MapReduce. ACM Computing Surveys 46:1–41
10. Shim K (2013) MapReduce Algorithms for Big Data Analysis. Databases in Networked Information Systems, Springer, Berlin, Heidelberg, Germany, pp 44–48
11. Shim K (2012) MapReduce Algorithms for Big Data Analysis, Framework. Proceedings of the VLDB Endowment 5(12):2016–2017
12. Lee K-H, Lee Y-J, Choi H, Chung YD, Moon B (2011) Parallel Data Processing with MapReduce: A Survey. ACM SIGMOD Record 40(4):11–20
13. Yang J, Li X (2013) MapReduce Based Method for Big Data Semantic Clustering. In Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference. Manchester, England, pp 2814–2819
14. Ene A, Im S, Moseley B (2011) Fast Clustering using MapReduce. Proc. of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, New York, USA, pp 681–689
15. 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 1–5
16. Esteves RM, Rong C (2011) Using Mahout for clustering Wikipedia's latest articles: a comparison between K-means and fuzzy C-means in the cloud. In Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference. Athens, Greece, pp 565–569
17. Li HG, Wu GQ, Hu XG, Zhang J, Li L, Wu X (2011) K-means clustering with bagging and mapreduce. Proc. 2011 44th Hawaii International Conference on IEEE System Sciences (HICSS). Kauai/Hawaii, US, pp 1–8
18. Zhang G, Zhang M (2013) The Algorithm of Data Preprocessing in Web Log Mining Based on Cloud Computing. In 2012 International Conference on Information Technology and Management Science (ICITMS 2012) Proceedings Springer. Berlin, Heidelberg, Germany, pp 467–474
19. Morales GDF, Gionis A, Sozio M (2011) Social content matching in mapreduce. Proceedings of the VLDB Endowment 4(7):460–469
20. Verma A, Llora X, Goldberg DE, Campbell RH (2009) Scaling Genetic algorithms using MapReduce. Intelligent Systems Design and Application(ISDA). Ninth International Conference, Pisa, Italy, pp 13–18
21. Cambria E, Rajagopal D, Olsher D, Das D (2013) Big Social Data Analysis. Big Data Computing Chapter 13:401–414
22. Lieberman M (2014) Visualizing Big Data: Social Network Analysis. Digital Research Conference, San Antonio, Texas, pp 1–23
23. López V, Río SD, Benítez JM, Herrera F (2014) Cost-sensitive linguistic fuzzy rule based classification systems under the MapReduce framework for imbalanced big data. Fuzzy Sets Syst 1:1–34
24. Blanas S, Patel JM, Ercegovac V, Rao J, Shekita EJ, Tian Y (2010) A Comparison of Join Algorithms for Log Processing in MapReduce. Proc. of the 2010 ACM SIGMOD International Conference on Management of data. New York, USA, pp 975–986
25. Hoi SCH, Wang J, Zhao P, Jin R (2012) Online Feature Selection for Mining Big Data. Proc. of the 1st International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms. Systems, Programming Models and Applications, ACM, New York, USA, pp 93–100
26. Chen S-Y, Li J-H, Lin K-C, Chen H-M, Chen T-S (2013) Using MapReduce Framework for Mining Association Rules. In Information Technology Convergence Springer, Netherlands, pp 723–731
27. Urbani J, Maassen J, Bal H (2010) Massive Semantic Web data compression with MapReduce. Proc. of the 19th ACM International Symposium on High Performance Distributed Computing. New York, USA, pp 795–802
28. Rajdho A, Biba M (2013) Plugging Text Processing and Mining in a Cloud Computing Framework. In Internet of Things and Inter-cooperative Computational Technologies for Collective Intelligence Springer, Berlin, Heidelberg, Germany, pp 369–390
29. Balkir AS, Foster I, Rzhetsky A (2011) A Distributed Look-up Architecture for Text Mining Applications using MapReduce. High Performance Computing, Networking, Storage and Analysis (SC), 2011 International Conference. Seattle, US, pp 1–11
30. Zongzhen H, Weina Z, Xiaojuan D (2013) A fuzzy approach to clustering of text documents based on MapReduce. In Computational and Information Sciences (ICCIS), 2013 Fifth International Conference on IEEE. Shiyang, China, pp 666–669
31. Chen F, Hsu M (2013) A Performance Comparison of Parallel DBMSs and MapReduce on Large-Scale Text Analytics. Proc. of the 16th International Conference on Extending Database Technology ACM. New York, USA, pp 613–624
32. Das TK, Kumar PM (2013) BIG Data Analytics: A Framework for Unstructured Data Analysis. International Journal of Engineering and Technology (IJET) 5(1):153–156
33. Momtaz A, Amreen S (2012) Detecting Document Similarity in Large Document Collection using MapReduce and the Hadoop Framework . BS Thesis. BRAC University, Dhaka, Bangladesh, pp 1–54
34. Lin J, Dyer C (2010) Data-Intensive Text Processing with MapReduce. Morgan & Claypool Publishers 3(1):1–177
35. Elsayed T, Lin J, Oard DW (2008) Pairwise Document Similarity in Large Collections with MapReduce. Proc. of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies. Stroudsburg, US, pp 265–268
36. Galgani F, Compton P, Hoffmann A (2012) Citation based summarisation of legal texts. Proc. of 12th Pacific Rim International Conference on Artificial Intelligence. Kuching, Malaysia, pp 40–52

37. Hassel M (2004) Evaluation of Automatic Text Summarization. Licentiate Thesis, Stockholm, Sweden, pp 1–75
38. Hu Q, Zou X (2011) Design and implementation of multi-document automatic summarization using MapReduce. Computer Engineering and Applications 47(35):67–70
39. Lai C, Renals S (2014) Incorporating Lexical and Prosodic Information at Different Levels for Meeting Summarization, Proceedings of the 15th Annual Conference of the International Speech Communication Association, INTERSPEECH 2014. ISCA, Singapore, pp 1875–1879
40. Fowkes J, Ranca R, Allamanis M, Lapata M, Sutton C (2014) Autofolding for Source Code Summarization. Computing Research Repository 1403(4503):1–12
41. Tzouridis E, Nasir JA, Lahore LUMS, Brefeld U (2014) Learning to Summarise Related Sentences. The 25th International Conference on Computational Linguistics (COLING'14), Dublin, Ireland, pp 1–12, ACL
42. Wang Y, Bai H, Stanton M, Chen WY, Chang EY (2009) Plda: Parallel latent dirichlet allocation for large-scale applications. 5th International Conference, AAIM (Algorithmic Aspects in Information and Management), San Francisco, CA, USA, pp 309–322
43. Miller GA (1995) WordNet: a lexical database for English. Commun ACM 38(11):39–41
44. Blei DM, Ng AY, Jordan MI (2003) Latent Dirichlet Allocation. The Journal of Machine Learning Research 3:993–1022
45. Feldman R, Sanger J (2007) The Text Mining Handbook - Advanced Approaches In Analyzing Unstructured Data. Press, Cambridge University. ISBN 978-0-521-83657-9
46. McCallum A K (2002) Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu/. Accessed 10 May 2014
47. Galgani F, Compton P, Hoffmann A (2012) Combining Different Summarization Techniques for Legal Text. Proc. of the Workshop on Innovative Hybrid Approaches to the Processing of Textual Data. Association for Computational Linguistics, Avignon, France, pp 115–123
48. Galgani F, Compton P, Hoffmann A (2014) HAUSS: Incrementally building a summarizer combining multiple techniques, Int. J. Human-Computer Studies 72:584–605
49. Li W (1992) Random Texts Exhibit Zipf's-Law-Like Word Frequency Distribution. IEEE Trans Inf Theory 38(6):1842–1845
50. Reed WJ (2001) The Pareto, Zipf and other power laws. Econ Lett 74(1):15–19
51. Goldstein J, Mittal V, Carbonell JG, Kantrowitz M (2000) Multi-Document Summarization By Sentence Extraction. School of Computer Science, Carnegie Mellon University, Research Showcase, pp 40–48
52. Lin CY (2004) Rouge: a package for automatic evaluation of summaries. In: Out TSB (ed) Proceedings of the ACL-04 Workshop. Association for Computational Linguistics, Barcelona, Spain, pp 74–81
53. Nenkova A, Passonneau R (2004) Evaluating Content Selection in Summarization: The Pyramid Method. Proc. Human Language Technology Conf. North Am. Chapter of the Assoc. for Computational Linguistics (HLT-NAACL), Boston, Massachusetts, pp 145–152
54. Harnly A, Nenkova A, Passonneau R, Rambow O (2005) Automation of Summary Evaluation by the Pyramid Method, In Recent Advances in Natural Language Processing (RANLP). Borovets, Bulgaria, pp 226–232
55. Qazvinian V, Radev DR (2008) Scientific Paper Summarization Using Citation Summary Networks. Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1, Stroudsburg, PA, pp 689–696
56. Wang D, Li T (2012) Weighted Consensus Multi-document Summarization. Inf Process Manag 48:513–523
57. Amdahl GM (1967) Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities. Proceedings of the April 18–20, 1967, spring joint computer conference. Atlantic City, New Jersey, USA, pp 483–485