International Journal of Electrical and Computer Engineering (IJECE)
Vol. 6, No. 3, June 2016, pp. 945 – 954
ISSN: 2088-8708, DOI: 10.11591/ijece.v6i3.9030

Fast Document Summarization using Locality Sensitive Hashing and Memory Access Efficient Node Ranking

Ercan Canhasi*
*Faculty of Computer Science, University of Prizren

Article Info

Article history:
Received Feb 5, 2016
Revised May 7, 2016
Accepted May 19, 2016

Keyword:
Document summarization
Locality Sensitive Hashing
I/O Access Efficient Node Ranking
Min-Hashing
Timeline summarization
Comparative summarization

ABSTRACT

Text modeling and sentence selection are the fundamental steps of a typical extractive document summarization algorithm. The common text modeling method connects a pair of sentences based on their similarities. Even thought it can effectively represent the sentence similarity graph of given document(s) its big drawback is a large time complexity of $O(n^2)$, where n represents the number of sentences. The quadratic time complexity makes it impractical for large documents. In this paper we propose the fast approximation algorithms for the text modeling and the sentence selection. Our text modeling algorithm reduces the time complexity to near-linear time by rapidly finding the most similar sentences to form the sentences similarity graph. In doing so we utilized Locality-Sensitive Hashing, a fast algorithm for the approximate nearest neighbor search. For the sentence selection step we propose a simple memory-access-efficient node ranking method based on the idea of scanning sequentially only the neighborhood arrays. Experimentally, we show that sacrificing a rather small percentage of recall and precision in the quality of the produced summary can reduce the quadratic to sub-linear time complexity. We see the big potential of proposed method in text summarization for mobile devices and big text data summarization for internet of things on cloud. In our experiments, beside evaluating the presented method on the standard general and query multi-document summarization tasks, we also tested it on few alternative summarization tasks including general and query, timeline, and comparative summarization.

Corresponding Author:
Ercan Canhasi
Faculty of Computer Science, University of Prizren
Rruga e Shkronjave, nr. 1 20000 Prizren, Kosove
ercan.canhasi@uni-prizren.com

1. INTRODUCTION

The world is floating on data. These data are mainly coming from the word wide web which is expanding exponentially making massive volume of the online information available for users. For years we have been affected by the quantity of data streaming through and produced by our systems. Automatic document summarization as the complementary tool to regular web search engines can be used to scale down this problem of information overload. Since the most of mobile and interactive ubiquitous multimedia devices have restricted hardware such as CPU, memory, and display screen it is essential to compress an document collection to a brief summary before it is delivered to the user of these devices. Other technology trends which can largely benefit from a scalable document summarization methods are the Internet of Things (IoT) on Cloud and Big data. The later is about the processing and analysis of large data repositories on Cloud computing. Big document summarization method is an important technique for data management of IoT. Traditional document summarization methods are restricted to summarize suitable information from the exploding IoT big data on Cloud.

The main task of the extraction based multi-document summarization is to extract the most important sentences from multiple documents and format them into a summary. According to the number of documents to be summarized, the summary can be a single document or a multi-document. Query-focused multi-document summarization is a special case of multi-document summarization. Given a query, the task is to produce a summary which can respond to the information required by the query. Two other specific document summarization tasks treated in

Table 1. Observed execution time spent on calculations (in seconds). The time elapsed in computing the summaries are measured on processor with following specifications: Intel(R) Core(TM) i5 CPU M 450 @ 2.45GHz with 4GB RAM memory. The first two columns, document(s) length in KB, and the total number of sentences, represent the input values. While the rest four columns, text modeling by means of LHS and conventional all-to-all comparing methods, sentece selection with I/O access efficient node ranking and archetypal analysis based sentence selection, are the measured times or the output values of the experiment.

Doc.(s) length	# of sentences	Text modeling		Sentence selection	
		LSH	all to all	node ranking	AA
1KB	13	0.008	0.09	0.021	1.73
20KB	160	0.054	1.40	0.056	1.11
45KB	366	0.138	2.28	0.139	4.82
100KB	744	0.303	5.89	0.308	9.21
644KB	5112	4.110	191.06	4.123	207.89

this work are timeline and comparative summarization. Timeline summarization aims at producing a sequence of compact summaries for news sets broadcast at various periods. Comparative News Summarization aims to outline the mutualities and contrasts between comparable news subjects.

In this paper, we propose a scalable solution to multi-document summarization based on the randomized algorithms. Many sentence similarity graph generation algorithms make use of some distance similarity (e.g., cosine similarity) to measure pairwise distance between sets of vectors representing corresponding sentences. Assume that we are given n sentences with a maximum of m terms. Calculating the full similarity matrix would take time complexity n^2m. Many novel summarization tasks such as the comparative, update and time-line summarization require processing the large number of sentences. Having an n^2m algorithm in such setups would be very impractical. Fortunately, we can borrow some ideas from the Math and Theoretical Computer Science to develop a scalable document summarization algorithm proportional to nm. The essence of our methods lies in defining Locality Sensitive Hash (LSH) functions. LSH functions involve the creation of short signatures (fingerprints) for each vector in space such that those vectors that are closer to each other are more likely to have similar fingerprints. LSH functions are generally based on randomized algorithms and are probabilistic.

The contribution of this paper is fourfold: 1) Paper presents a new fast sentence selection algorithm able to scale effortlessly; 2) We describe the method for sub-linear time text modeling by means of sentence similarity graph and very efficient node ranking in those graphs; 3) No individual part of our method is new or revolutionary. Locality sensitive hashing has been done before, as have node ranking and its usage in summarization. The novelty is in the combination of these individually useful parts into a single, coherent, real-time summarization system. We have not seen LSH nor our node ranking implementation applied to summarization in this way before; 4) We extensively evaluated our method on few different summarization tasks.

The remainder of the paper is organized as follows: Section 2 first briefly presents the related work. In Section 3 we describe the centerpiece of this work namely the fast document summarization method. Section 4 gives the description of the test environment and data sets we used for testing. The results are also presented in Section 5. We conclude the paper and set guidelines for further work in Section 6.

2. RELATED WORK

Our work is related to various research fields including general and query focused summarization [1, 2, 3, 4, 6, 5, 7], timeline summarization [8, 9], comparative summarization [10], sampling-based document summarization algorithms [11], node ranking of the sentence similarity graph [12, 13, 14] and similarity search for high dimensional data objects [15, 16]. Following paragraphs give a brief survey of these works.

In [11], authors use Random Indexing for text modeling. Random indexing presents a computationally efficient way of implicit dimensionality reduction. It involves inexpensive vector computations such as addition and thus provides an efficient way to compute similarities between words, sentences and documents.

The algorithm that we use in this paper, min-hash [17], was originally developed for returning only the authoritative documents in search results. Another closely-related problem is one know as the text reuse [18]. In contrast to near-duplicate detection, the focus is usually on smaller segments of text as opposed to entire documents. Other similar formulations of the problem are what the data mining community calls pairwise similarity search or all pairs search [19] and what the database community calls set similarity join [20].

Our previous work [3, 4, 5] present an extractive summarization framework based on three alternative models to integrate the archetypal analysis based sentence selection: (1) the plain archetypal analysis sentence clustering and ranking for general; (2) the weighted archetypal analysis sentence selection for the query focused document summarization and (3) the weighted hierarchical archetypal analysis sentence selection for 4 different summarization tasks.

Timeline summarization (TS for short) has become a widely adopted, natural way to present long news stories in a compact manner. Existing approaches for TS aim to generate a good daily summary for each of these dates (e.g, [8, 9]). In this study, we set our focus on showing how the presented method can directly without extra effort be used in TS problem.

Comparative multi document summarization (CDS) is first proposed in [10] to summarize differences between comparable document groups. [10] presents a sentence selection strategy modeled by means of conditional entropy, which precisely discriminates the documents in different groups.

Graph-based methods like TextRank [12] and PageRank [13] model a document or a set of documents as a text similarity graph, constructed by taking sentences as vertices and the similarity between sentences as edge weights. They take into account the global information and recursively calculate the sentence significance from the entire text graph rather than simply relying on unconnected individual sentences. From an NLP perspective, extractive summarization embodies two criteria: sentence relevance and sentence redundancy. Graph-based sentence ranking algorithms successfully merge both of these criteria into a single framework, by utilizing the so-called graph-based lexical centrality principle. Graph-based ranking algorithms were also used in query-focused summarization when it became a popular research topic. For instance, a topic-sensitive version of LexRank is proposed by [14]. It integrates the relevance of a sentence to the query into LexRank to get a biased PageRank ranking.

Similar work to ours [21] presents a new principled and versatile summarization framework MDS using the submodal function. This framework can deal with different summarization tasks, including generic, query-focused, updated, comparative summarization. The empirical results show that this framework outperforms the other rivals in the generic summarization and is competitive in other summarization tasks. In [22] authors have investigated the use of maximum entropy, naive-Bayes, support vector machine models and a hybrid machine model for multi-document automatic text summarization.

3. FAST DOCUMENT SUMMARIZATION

3.1. Motivation

The trend in automatic document summarization approaches found in state of the art systems proposes a general summarization methods which consists of the following sub-tasks: 1. Text modeling: convert the text into, for instance graph representation 2. Sentence ranking: identify the salient sentences from given text model 3. Summary generation: extract selected sentences into final summary.

In order to obtain the sentences similarity graph one needs to compute the similarity values for all possible pairs of sentences in order to connect them in the sentence similarity graph. Mainly the vector space model is used to represent sentences from given documents. The vector space model is an algebraic model for representing sentences as vectors of terms. The cosine similarity is a measure of similarity between two vectors of an inner product space that measures the cosine of the angle between them. Assuming that multiplication and addition are constant-time operations, the time complexity of computing the cosine similarity where m is the biggest number of terms is therefore $O(m) + O(m) = O(m)$. But since we need to compute the sentence similarity for every pair of sentences then the time and space complexity of generating the sentence similarity graph becomes $O(n(n-1)/2)$, here n is the number of sentences. In order to give a better gist of the time complexity we report the elapsed time in producing the summary for different document(s) lengths in in Table 1. Not only the similarity graph calculation is time expensive, but usually sentence selection methods are also very time consuming.

Hence, this paper presents the way for using the fast randomized approximation algorithm (i.e., LSH and min-hash), to deal with the quadratic complexity of the conventional text modeling techniques. Our approximation algorithm utilizes Locality-Sensitive Hashing, abbreviated as LSH hereafter, which is a probabilistic approximation algorithm for the nearest neighbor search. We do not only try to increase the speed and the scalability of the summary production system on the text modeling level but we also present our contribution on the sentence selection/extraction level.

For text modeling we propose the following method (Essential Steps in similarity graph computation):1. N-gram extraction: Convert sentences into sets 2. Min-Hashing: Convert wide sets to short signatures, while preserving similarity 3. Locality-Sensitive Hashing: Concentrate on couples of signatures probable to originate from similar sentences

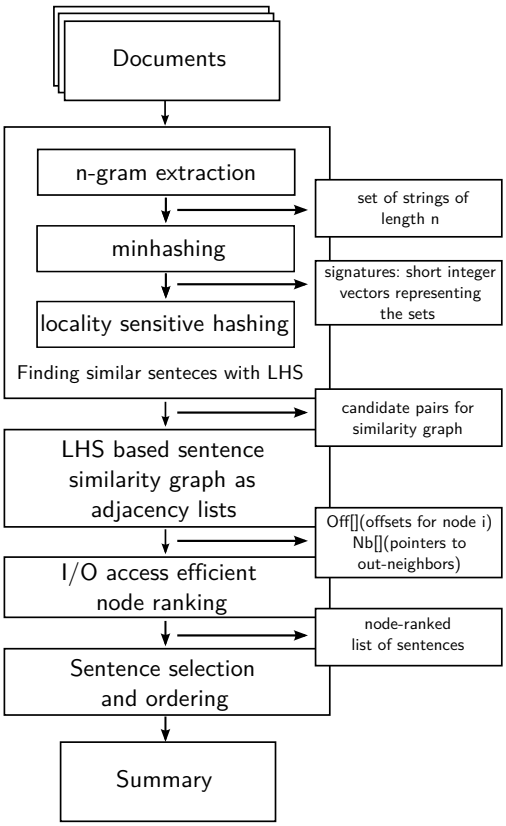


Figure 1. Method overview

Given the sets of similar sentences we can very efficiently compute the sentence ranking by mapping the problem of sentence selection to node ranking in the graph of similar sentence sets. For sentence modeling we propose the following method: Essential steps in I/O efficient node ranking 1. Get the input sentence similarity graph represented as a set of three arrays 2. Produce the sentence ranking by recursively computing the eigenvector decompositions 3. Return the sentence ranking

In the rest of the section we describe these steps in more details.

3.2. Fast similarity graph computing

In this subsection, the problem of sentence similarity is fist described as search for the sets with a approximately big intersection. We then show how the problem of finding textually similar sentences can be turned into such a set problem by representing given text entities as a set of n-grams. Then, we present how method known as min-hashing can be used for shortening these huge n-gram sets while preserving the similarity information of the underlying sets. And finally we utilize the locality-sensitive hashing for adjusting our search on couple of sentences that are most probable to be similar.

Let us assume the similarity of the pair of sentences can be deduced by barely looking at the relative size of their intersection. This is the similarity measure know as Jaccard similarity. If the Jaccard similarity of sets W and Z is $|W \cap Z|/|W \cup Z|$, than the Jaccard similarity of sentences S_1 and S_2 can be denoted as $SIM(S_1, S_2)$. The form of similarity we are utilizing here is character-level similarity.

A very simple but productive method for representing sentences as sets is to describe them as the sets of very short strings that occur within sentences. In this way sentences that share pieces as short as words or even syllable will have many common elements in their sets, even if those common entities appear in different orders in the two sentences. Define a n-gram for a sentence to be any substring of length n found within the sentence. Then, correlate each sentence with the set of n-grams that appear one or more times within that sentence. Instead of manipulating the sub-strings as n-grams, we choose a hash function that maps them to some number of buckets and handle the final bucket number as the n-gram. The set defining a sentence eventually becomes a set of integers that are bucket numbers of one or more n-grams that appear in the sentence. In this way we drastically compress the original textual data.

Algorithm 1 Fast min-hashing algorithm

procedure MINHASHING($S[], H[], n, k$)
  Input: S (Set of n-grams), H (N random hash functions), n (number of n-grams), k (number of hash functions);
  Output: c set of min-hash signatures of the input set of n-grams S ;
  $c[] \leftarrow$ new int$[n]$
  for $i = 0$ to n **do**
    $c[i] \leftarrow \infty$
  for $i = 1$ to n **do**
    if S(i)==1 **then**
      for $i = 0$ to k **do**
        if $h_j(i) == c_j$ **then**
          $c[i] \leftarrow \infty$
end procedure

Algorithm 2 Approximate nearest neighbor search

procedure LSH($M[,], s, b, r$)
  Input: M (min-hash signature matrix), s (similarity threshold), b the number of bands, r the number of rows;
  Output: F set of documents with jaccard similarity at least s;
  Divide matrix M into b bands of r rows
  for each b in band **do** hash b portion of each column to a hash table with k buckets; make k as large as possible
  end for
  Candidate column pairs are those that hash to the same bucket for ≥ 1 band
  Tune b and r to catch similar sentences
end procedure

Since the sets of n-grams are usually large, one can replace them by much smaller representations called signatures. Signatures can be calculated using the method known as the min-hashing, briefly given in Algorithm 1. This technique is developed to guarantee that two similar objects generate hashes that are themselves similar. In fact, the similarity of the hashes has a direct relationship to the similarity of the sentences they were generated from. This ratio temps to approximate the Jaccard Similarity.

Although we use min-hashing to compress large sets into small signatures while yet preserving the expected similarity of any pair of sentences, there is still another very important issue. Finding the pairs of sentences with greatest similarity efficiently can be very time consuming. The reason is that the number of pairs of sentences may be too large. The brute-force approach would be to compare each sentence with each other sentence, using MinHash, which obviously has the quadratic time complexity. A faster solution is to use Locality Sensitive Hashing (LSH). This takes the MinHash values for sentences and hashes the MinHash values so they hash into the same bucket if they are similar. The brief algorithm is described in 2. Note that the computation time for LSH with MinHash depends only on the number of sentences and number of MinHash functions used and not on the length of the sentences.

We can now give an approach to finding the set of candidate pairs for similar sentences and then discovering the truly similar sentences among them:

1. Pick a value of n and construct from each sentence the set of n-grams.
2. hash the n-grams to shorter bucket numbers.
3. Sort the sentence and n-grams pairs to order them by latter.
4. Pick a length n for the minhash signatures. Feed the sorted list to the algorithm 1 to compute the minhash signatures for all the sentences.
5. Choose a threshold t that defines how similar sentences have to be in order for them to be regarded as a desired "similar pair." Pick a number of bands b and a number of rows r such that $b * r = n$, and the threshold t is approximately $(1/b)1/r$.
6. Construct candidate pairs by applying the LSH technique described in algorithm 2.

7. Connect the sentences in the similarity graph based on the candidate pairs calculated by LHS.

3.3. Sentence selection

In this subsection we describe an I/O efficient graph based ranking method for sentence selection from the graph of sentences. The construction methodology of graph was presented in previous subsection. The idea has been vastly used in document summarization since the pioneering works known as PageRank [pagerank], and textrank [textrank]. To efficiently compute the PageRank scores for a big graphs, the input sentence similarity graph has to be represented as a binary link structure, more specifically as a set of three arrays: $SenL$ (list of the n sentences), Off (array of integers which denotes the offsets of list for node i) Nb (array of integers which denotes the pointers to out-neighbors); Using the above structure, a simple I/O efficient PageRank algorithm can be written in Algorithm 3. Note that except for $newpr[]$ array, which represents the PageRank values, all arrays are scanned only once sequentially from front to end.

Given the node ranking our summarization approach will extract the most important nodes, i.e sentences, to include in the summary. Here an additional sentence penalization step is applied. Suppose x_i is the highest ranked sentence. Sentence x_i is moved to set of sentences representing the final summary, and then the redundancy penalty is imposed to the overall rank score of each sentence linked with x_i as follows: for each sentence x_j, its rank score $RScore(x_j)$ is computed by:

$$RScore(x_j) = RScore(x_j) * (1 - Sim_{ji})^t, t > 0$$

is the exponent decay factor. The larger t is, the greater penalty is imposed to the overall rank score. If $t = 0$, no diversity penalty is imposed at all; In our experiments we set $t = 3$; Presented penalization algorithm is based on the idea that extracting the overall rank score of less informative sentences overlapping with the sentences in summary is iteratively decreased. Here, redundancy removal is also the key step of content selection. Finally, the sentence with the highest rank score is chosen to produce the summary until satisfying the summary length limit.

Algorithm 3 I/O efficient node ranking

```
1:  procedure SENTENCERANKING(SenL, Off, Nb)
2:      Input: SenL (list of the n sentences), Off (array of integers which denotes the offsets of list for node i) Nb
        (array of integers which denotes the pointers to out-neighbors);
3:      Output: node-ranked list of sentences;
4:      n ← SenL.Count
5:      β ← 0.15
6:      m ← 10
7:      pr[] ← newfloat[n]
8:      newpr[] ← newfloat[n]
9:      for i = 0 to n do
10:         pr[i] ← 1/n
11:         newpr[i] ← (1 − β)/n
12:     for k = 0 to m do
13:         for i = 0 to Off.Count − 1 do
14:             outd ← Off[i + 1] − Off[i]
15:             for j = Off[i] to (Off[i + 1] − 1) do
16:                 newpr[NB[j]]+ = (beta ∗ (pr[i]/outd))
17:         newpr[i] ← (1 − β)/n
18: end procedure
```

4. EXPERIMENTS

In this section, we conduct experiments to evaluate the effectiveness and possible positive contributions of the proposed method compared with other existing summarization systems on few different summarization tasks including General/Query, Comparative, and Timeline summarization.

4.1. Evaluation Metric

We used the metric based on the ROUGE scores that are widely used in traditional summarization tasks. Recall Oriented Understudy for Gisting Evaluation (ROUGE) evaluation package [23], compares various summary results from several summarization methods with summaries generated by humans. In timeline evaluation tasks, the quality of different TSs are compared via F-measure of the ROUGE-1, ROUGE-2. In this paper, we adopted the same metrics, plus the additional ROUGE-S*. Technically, ROGUE-S* is computed the same as bigram-based ROGUE-2 scores, but it allows the words in the bigram to be aparted by a window. This makes ROGUE-S* capture better the global distributional semantics, while traditional ROGUE-Ns capture better the local semantics, i.e. sentence to sentence matching.

The set of input parameters for FastSum namely 1) the number of ngrams; 2) the number of bands; 3) the number of rows; 4) and the number of elements are separately defined for each kind of treated summarization task as reported in Table1. The rational for picking up these values are purely empirical and are based on the experiments presented in the rest of the paper.

Table 2. FastSum Input parameters

Task	#ngrams	#bands	#rows	#elements
General/Query	4	20	2	40
Timeline	4	20	3	60
Comparative	6	20	5	100

4.2. General summarization

We use the DUC05 and DUC06 data sets to evaluate our proposed method empirically on general and query focused summarization tasks. Benchmark data sets are from DUC[1] for automatic summarization evaluation. DUC05 and DUC06 data sets consist of 50 topics. The task is to create a summary of no more than 250 words. In those document data sets, stop words were removed using the stop list[2] and the terms were stemmed using the Porter's scheme[3], which is a commonly used algorithm for word stemming in English.

Table 3. Evaluation of the methods on the DUC2005 dataset.

Summarizers	ROUGE-1	ROUGE-2	ROUGE-SU4
Avg-Human	0.4417 (1)	0.1023 (1)	0.1622 (1)
Avg-DUC05	0.3434 (7)	0.0602 (6)	0.1148 (7)
System-15	0.3751 (4)	0.0725 (4)	0.1316 (4)
System-4	0.3748 (5)	0.0685 (5)	0.1277 (5)
Biased-Lex	0.3861 (3)	0.0753 (3)	0.1363 (3)
wAASum	0.3945 (2)	0.0797 (2)	0.1420 (2)
FastSum	0.3697 (6)	0.0506 (7)	0.1168 (6)

We work with the following methods for general/query-focused summarization as the baseline systems to compare with our proposed method:

(1) Avg-Human: the average human summarizer on DUC05(06); (2) Avg-DUC05(06): the average system summarizer; (3) System-15(24): The best system-summarizer from DUC05(06); (4) System-4(12): The second best system summarizer from DUC04(05); (5) Lex-PageRank: by calculating the eigenvector centrality given the sentence to sentence similarity graph the method extracts the most significant sentences; (6) wAASum: weighted Archetypal analysis summarization system of the sentence similarity graph; (7) FastSum: the method presented by this paper.

Although there are, for each year, more than 30 systems that have participated in DUC competition, here we only compare with the DUC human best, the DUC human average, the DUC system best and the DUC system average result.

[1] http://duc.nist.gov
[2] ftp://ftp.cs.cornell.edu/pub/smart/english.stop
[3] http://www.tartarus.org/martin/PorterStemmer/

Table 4. Evaluation of the methods on the DUC2006 dataset.

Summarizers	ROUGE-1	ROUGE-2	ROUGE-SU4
Avg-Human	0.4576 (1)	0.1149 (1)	0.1706 (1)
Avg-DUC06	0.3795 (7)	0.0754 (6)	0.1321 (7)
System-24	0.4102 (3)	0.0951 (2)	0.1546 (4)
System-12	0.4049 (5)	0.0899 (4)	0.1476 (5)
Biased-Lex	0.3899 (6)	0.0856 (5)	0.1394 (6)
wAASum	0.4238 (2)	0.0917 (3)	0.1671 (2)
FastSum	0.4086 (4)	0.0710 (7)	0.1616 (3)

Table 5. Results in comparative summarization: Sentences selected by our proposed FastSum approach.

ID	Selected sentence
1	At the Madrid summit last December, leaders of EU member nations agreed unanimously that the European single currency will be formally launched on January 1, 1999.
2	The Wa National Organization, the Palaung State Liberation Front and the Lahu Democratic Front said the arrest were an insulting act of shameless, barbaric arrogance against the people of Burma.
3	ETA, which stands for Basque Homeland and Freedom, has killed nearly 800 people in its 30-year campaign for an independent Basque nation carved out of parts of northern Spain and southern France.
4	Unemployment in France fell to 11.6 percent in October from 11.7 percent, reflecting a slow but steady improvement in France's high jobless rate, long one of the nation's knottiest problems.
5	Researchers evaluate overweight and obesity using a measure called body mass index BMI , which divides weight in kilograms by the square of height in meters.

Tables 3 and 4 show the ROUGE scores of different methods using DUC05 and DUC06 data sets, respectively. The higher ROUGE score indicates the better summarization performance. The number in parentheses in each table slot shows the ranking of each method on a specific data set. Even thought our results are not among the best we show that by sacrificing a rather small percentage of recall and precision in the quality of the produced summary can reduce the quadratic to sub-linear time complexity of other typical summarization systems.

4.3. Comparative summarization

In this section we investigate one of the recent summarization tasks, first proposed by [10].

We model the comparative summarization as follows: Extract the summaries $S_1, S_2, ..., S_N$ from the given N groups of documents $G_1, G_2, ..., G_N$. Extracted summaries should be as divergent as possible from one another on the topic level while still expressing central themes of corresponding groups.

We propose a following function for the comparative summarization to generate the discriminant summary for each group of documents:

$$Cs = \left[(i,j) \mid \begin{cases} sim_{norm}(s_i, s_j) & \text{if } G(s_i) = G(s_j) \\ -sim_{norm}(s_i, s_j) & \text{if } G(s_i) \neq G(s_j) \end{cases} \right]_{t \times t} \quad (1)$$

where $G(s_i)$ is the document group containing sentence s_i, $sim_{norm}(s_i, s_j)$ is the normalized sentence similarity.

Evaluation: Since there is currently no dataset/methodology available to carry out a quantitative evaluation of comparative summarization we used five clusters of documents from the DUC07 corpora to generate comparative summaries using the FastSum method. The data set contains five clusters as follows: 1. Steps toward introduction of the Euro; 2. Burma government change 1988; 3. Basque separatist movement 1996-2000. 4. Unemployment in France in the 1990s; 5. Obesity in the United States and possible causes for US obesity;

Looking at the results by FastSum sentence selection method in Table 5, each of the sentences represents one cluster respectively and summarizes well specific topics of each cluster. In Table 5, we also highlight some keywords

Table 6. Average results on 17 timelines, the reported results are computed 95% confidence interval

Summarizers	ROUGE-1	ROUGE-2	ROUGE-SU4
Random	0.128	0.021	0.026
Chieu et al.	0.202	0.037	0.041
Tran et al.	0.230	0.053	0.050
FastSum	0.197	0.032	0.039

representing the unique features of each topic. Note that the sentence extracted by FastSum for each topic are not just discriminative but they also present the essence of the topic.

4.4. Timeline summarization

In order to evaluate out method on timeline summarization task we used Timeline17 dataset [8]. Briefly, the dataset consists of 17 manual-created timelines and their associated news articles. Data set are published online [4].

We evaluate our system against traditional multi document summarization and timeline generation systems. Following is the list of those systems: *Random:* The system generates day summary for each day by randomly selecting sentences for particular day. *[9]* is multi-document summarizer which utilizes the popularity of a sentence as TFIDF similarity with other sentences to estimate its importance. *[8]* They use SVM-rank to demonstrate the performance of their system, which is one of the most common learn to rank implementa‘tions.

Result: The average results of TS generation on given dataset are represented in Table 6. Although when compared to other two systems ours seems to perform more poorly this is mainly due to its simplicity which is payed by its scalability.

5. CONCLUSION AND FUTURE WORK

A particular challenge for graph based multi-document summarization methods is a large time complexity of at least $O(n^2)$ for text modeling and some additional complexity of sentence selection. Hence we need effective summarization methods to reduce this high time complexity. In this paper we have formalized the problem of the fast and scalable document summarization method as combination of (1) the text modeling sub-problem of calculating the similarity graph based on locality sensitive hashing and (2) the sentence selection sub-problem of I/O access efficient node ranking.

The contribution of the work can be summarized as: 1. The paper presents a new fast sentence selection algorithm able to scale effortlessly. 2. We describe the method for sub-linear time text modeling by means of sentence similarity graph and very efficient node ranking in those graphs. 3. No individual part of our method is new or revolutionary. Locality sensitive hashing has been done before, as have node ranking and its usage in summarization. The novelty is in the combination of these individually useful parts into a single, coherent, real-time summarization system. We have not seen LSH nor our node ranking implementation applied to summarization in this way before; 4. We extensively evaluated our method on few different summarization tasks.

In future the FastSum may be further improved. There are many potential directions for improvements, such as: 1. improving the FastSum into a distributed real-time multi-document summarization system 2. adopting FastSum to and testing it as the system capable of scaling to many servers and huge size of documents 3. in order to improve the quality of produced summaries one can enhance the sentence similarity calculation by using the wordnet and by adopting the LSH to fast semantic similarity calculation.

6. SOURCE CODE

All the source codes can be downloaded as SVN checkout at:
`https://github.com/ErcanCanhasi/FastDocumentSummarization.git`

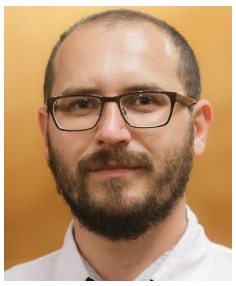
REFERENCES

[1] Pankaj B, Agrawal AJ. (2014) Extractive Based Single Document Text Summarization Using Clustering Approach. In: *IAES International Journal of Artificial Intelligence (IJ-AI) 2014; 3(2).*

[4] http://www.l3s.de/˜gtran/timeline/

[2] Pedram VA, Omid SSh. Scientific Documents clustering based on Text Summarization. In: *International Journal of Electrical and Computer Engineering (IJECE) 2015; 5(4): 782–787.*
[3] Canhasi E, Kononenko I. Multi-document summarization via Archetypal Analysis of the content-graph joint model. *Knowledge and Information Systems (KAIS)*, 2014; 41(3): 821–842.
[4] Canhasi E, Kononenko I. Weighted archetypal analysis of the multi-element graph for query-focused multi-document summarization. *Expert Systems with Applications (ESWA)*, 2014; 41(2): 535–543.
[5] Canhasi, E., Kononenko, I., Weighted hierarchical archetypal analysis for multi-document summarization. Comput. Speech Lang. (2015), http://dx.doi.org/10.1016/j.csl.2015.11.004
[6] Canhasi E, Kononenko I. Automatic Extractive Multi-document Summarization Based on Archetypal Analysis. *Non-negative Matrix Factorization Techniques. Springer Berlin Heidelberg*, 2016; 75-88.
[7] Dipti YS. Effect of feature selection on small and large document summarization. In: *IAES International Journal of Artificial Intelligence (IJ-AI) 2014; 3(3).*
[8] Tran GB, Tran AT, Tran NK, Alrifai M, Kanhabua N. Leveraging Learning To Rank in an Optimization Framework for Timeline Summarization. 2013
[9] Chieu HL, Lee YK. Query based event extraction along a timeline. In: *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval, ACM.* 2004; 425–432.
[10] Wang D, ZhuS L, Gong TY. Comparative document summarization via discriminative sentence selection, TKDD 2012; 6(3): 1–12.
[11] Chatterjee N, Mohan S. Extraction-based single-document summarization using random indexing. In: *Tools with Artificial Intelligence, ICTAI 19th IEEE International Conference* 2007; 448–455.
[12] Mihalcea R, Tarau P. TextRank: Bringing Order into Texts In: *Proceedings of Conference on Empirical Methods in Natural Language Processing*, EMNLP, ACL 2004; 404–411.
[13] Erkan G, Radev DR. LexRank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 2004; 457–479.
[14] Otterbacher J, Erkan G, Radev DR. Biased LexRank: Passage retrieval using random walks with question-based priors. *Information Processing and Management*, 2009; 45(1): 42-54.
[15] Andoni A, Indyk P. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In: *Foundations of Computer Science FOCS'06. 47th Annual IEEE Symposium on, (IEEE) 2006;* 459–468.
[16] Henzinger M. Finding near-duplicate web pages: a large-scale evaluation of algorithms. In: *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, (ACM SIGIR) 2006;* 284–291.
[17] Broder AZ, On the resemblance and containment of documents. In: *Compression and Complexity of Sequences 1997. Proceedings, (IEEE) 1997;* 21–29.
[18] Bendersky M, Croft WB. Finding text reuse on the web. In: *Proceedings of the Second ACM International Conference on Web Search and Data Mining, (ACM)* 2009; 262–271.
[19] Bayardo RJ, Ma Y, Srikant R. Scaling up all pairs similarity search. In: *Proceedings of the 16th international conference on World Wide Web, (ACM) 2007;* 131–140.
[20] Vernica R, Carey MJ, Li C. Efficient parallel set-similarity joins using MapReduce In: *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, ACM 2010; 495–506.
[21] Li J, Li L, Li T, Multi-document summarization via submodularity. Applied Intelligence, 2014; 37(3); 420–430.
[22] Fattah MA. A hybrid machine learning model for multi-document summarization. Applied intelligence 2014; 40(4): 592–600.
[23] Lin CY. Rouge: a package for automatic evaluation of summaries. In: *Text summarization branches out: proceedings of the ACL-04 workshop of ACL 2004;* 74-81.

BIOGRAPHY OF AUTHOR



Ercan Canhasi received his Ph.D. in 2014 from University of Ljubljana, Slovenia. He is a assistant professor at the Faculty of Computer Science in Prizren. His research interests include text mining, natural language processing and text summarization. He is the (co)author of few papers. Further info on his homepage: https://sites.google.com/a/uni-prizren.com/ercancanhasi/