Random Indexing and Modified Random Indexing based approach for extractive text summarization

Niladri Chatterjee a,*, Pramod Kumar Sahoo a,b

a Department of Mathematics, Indian Institute of Technology Delhi, Hauz Khas, New Delhi 110016, India
b Institute for Systems Studies & Analyses, Defence Research & Development Organisation, Metcalfe House Complex, Delhi 110054, India

Received 29 July 2012; received in revised form 15 April 2014; accepted 25 July 2014
Available online 8 August 2014

Abstract

Random Indexing based extractive text summarization has already been proposed in literature. This paper looks at the above technique in detail, and proposes several improvements. The improvements are both in terms of formation of index (word) vectors of the document, and construction of context vectors by using convolution instead of addition operation on the index vectors. Experiments have been conducted using both angular and linear distances as metrics for proximity. As a consequence, three improved versions of the algorithm, viz. RISUM, RISUM+ and MRISUM were obtained. These algorithms have been applied on DUC 2002 documents, and their comparative performance has been studied. Different ROUGE metrics have been used for performance evaluation. While RISUM and RISUM+ perform almost at par, MRISUM is found to outperform both RISUM and RISUM+ significantly. MRISUM also outperforms LSA+TRM based summarization approach. The study reveals that all the three Random Indexing based techniques proposed in this study produce consistent results when linear distance is used for measuring proximity.

© 2014 Elsevier Ltd. All rights reserved.

Keywords: Word Space Model; Random Indexing; PageRank; Convolution; Modified Random Indexing

1. Introduction

Extractive text summarization (Mani, 2001) aims at selecting the important sentences from the original text comprising single or multiple documents. Traditional extractive summarization techniques typically used simple heuristics, such as, word distribution in the source (Luhn, 1958), cue phrase method (Edmundson, 1969), location method (Edmundson, 1969; Hovy and Lin, 1999), discourse structure (Barzilay and Elhadad, 1999) with limited success. Subsequently, more comprehensive techniques have been developed by resorting to efficient representation of texts. Two major approaches in this regard are: Graph-based Model and Word Space Model (WSM).

Graph-based Model: Here a document is represented in the form of graphs, where sentences (or paragraphs) are represented as nodes; while weighted edges represent similarity between the nodes (Salton et al., 1997). Important
sentences (or paragraphs) are then extracted by analyzing the graph structure. TextRank (Mihalcea and Tarau, 2004) and LexRank (Erkan and Radev, 2004) are other graph-based methods proposed subsequently.

**Word Space Model (WSM):** Here words and sentences are represented in the form of vectors in a high-dimensional vector space, called Word Space (Schütze, 1993). The major WSM-based approaches include Latent Semantic Analysis (LSA) (Landauer and Dumais, 1997; Landauer et al., 1998), where orthogonal unary vectors\(^1\) are used to represent words of a document. The dimension of each such word vector is equal to the number of distinct words in the document. These word vectors are used to form the initial word-by-sentence matrix. For illustration, suppose the input text consists of \(m\) distinct words \(w_1, w_2, \ldots, w_m\), whose word vectors are \(m\)-dimensional unary vectors \(\vec{w}_1 = [1 \ 0 \ 0 \ \cdots \ 0]^T\), \(\vec{w}_2 = [0 \ 1 \ 0 \ \cdots \ 0]^T\), …, and \(\vec{w}_m = [0 \ 0 \ 0 \ \cdots \ 1]^T\), respectively. If the input text consists of \(n\) sentences \(S_1, S_2, \ldots, S_n\) and the weight assigned to the word \(w_i\) in sentence \(S_j\) is \(a_{ij}\), then the vector \(\vec{S}_j\) computed as \(\sum_{i=1}^{m} a_{ij} \vec{w}_i\) provides the vector representation of the sentence \(S_j\). In this way the document can be represented as a word-by-sentence matrix \(A = [\vec{S}_1 \ \vec{S}_2 \ \cdots \ \vec{S}_n]\) of dimension \(m \times n\). Since the number of words is much larger than the number of sentences, and normally each distinct word appears in few sentences only, \(A\) is a very high-dimensional sparse matrix. The weights \(a_{ij}, \quad i = 1, \ldots, m; j = 1, \ldots, n\) are usually composed of two components: a local weighting of the word \(w_i\) in the sentence \(S_j\), and a global weighting of the word \(w_i\) in the whole collection (Berry et al., 1995).

LSA is computationally expensive as it uses Singular Value Decomposition (SVD) (Klema and Laub, 1980) for dimension reduction of the huge and sparse word-by-sentence matrix \(A\). Random Indexing (RI) (Sahlgren, 2005, 2006) has been proposed as an alternative to LSA. RI allows incremental learning of context information by using near-orthogonal index vectors representing words of a given text. This helps in dimension reduction by resorting to index vectors of much lower dimension in comparison with the number of words in the given text. It thereby avoids the computationally expensive dimension reduction technique typically used in LSA.

The ability of these schemes to transform a linguistic model into a mathematical model made them amenable to rigorous mathematical treatments. Hassel and Sjöbergh (2006) presented a RI+Greedy Search based approach for extractive text summarization. Starting from a summary with lead sentences of the document, the algorithm at each step investigates all neighbours of the existing summary to create a better summary, where a neighbouring summary is one that can be created by removing one sentence of a given summary, and adding another. When all such summaries have been investigated, the one most similar to the original document is updated as the current best candidate. The process terminates when no summary better than the current candidate is obtained. In this approach, each text (original document/candidate summary) is assigned its own vector for semantic content, which is the (weighted) sum of all the context vectors (created using RI) of the words in the text. Similarity between two texts is measured in terms of the cosine angle between the semantic vectors of the texts. This approach, however, does not rank the individual sentences for their suitability to be included in the summary.

Hybridization of WSM and graph-based ranking has also been used for single document extractive summarization. Some of these approaches include:

- **LSA+TRM (Yeh et al., 2005):** This scheme uses LSA along with Text Relationship Map (TRM) for extracting the important sentences from the globally bushy path. The authors claim that when appropriate dimension reduction ratio is chosen, this approach outperforms keyword-based text summarization approaches.

- **RI+PageRank (Chatterjee and Mohan, 2007):** Here fixed size, near-orthogonal random index vectors have been used for implicit dimension reduction of WSM. Cosine similarity has been applied to measure similarity between sentences, and weighted PageRank algorithm has been applied to rank the sentences according to their importance. The authors claim that this scheme produces better summaries than commercially available summarizers, viz. Copernic and Word Summarizer. In the rest of the paper this system will be referred to as RI-Baseline.

However, this scheme suffers from two major drawbacks: abruptness in the generated summaries, and divergence of weighted PageRank algorithm (Smith and Jönsson, 2011). In our experiments with the RI-Baseline too we found the same problems.

\(^1\) A unary vector is a vector consisting of one 1 and rest entries 0 (Sahlgren, 2006, p. 43).
The abruptness in summaries produced by RI-Baseline can be attributed to the fact of complete randomness in the creation of index vectors (see Appendix A for an illustrative example). In this work we propose improvements to the RI-Baseline by working on the randomization of index vectors in a restricted way. In our approach we divide the index vectors in two halves. The ‘+1’s are placed randomly in the upper half, the ‘−1’ s are placed in the lower half. This in turn avoids mutual cancellation of ‘+1’s and ‘−1’s while summing the index vectors to create the context vectors of the constituent words.

The divergence of weighted PageRank algorithm in RI-Baseline system is due to the use of cosine similarity measure for computing proximity between two sentences. The cosine similarity measure between the sentences produces both positive and negative similarity values. A mixture of positive and negative similarity values often makes the weighted PageRank algorithm diverge (Appendix B explains this with an example). Smith and Jönsson (2011) tackled this problem using a much larger outside Word Space as a training set, which is expensive in terms of time and space. However, in this paper we handled this problem in a very simple and effective manner by using cosine dissimilarity measure instead of cosine similarity measure.

The above mentioned two modifications to RI-Baseline system results a new summarization system termed as RISUM. We made two further modifications to the scheme – which led to two successively improved versions of the scheme, termed RISUM+ and MRISUM, respectively. In RISUM+ we used the focus word as well while forming its context vector. In our experiments with DUC2 2002 documents we found both RISUM and RISUM+ to be better than RI-Baseline. In MRISUM we have used convolution (Borsellino and Poggio, 1973) as the operation to form context vectors instead of addition. We found MRISUM to be far superior to RISUM and RISUM+.

The paper is organized as follows. In Section 2 we discuss the RI scheme to some detail. Section 3 discusses RISUM and RISUM+; while Section 4 deals with the MRISUM scheme. Section 5 analyses their performance. Section 6 concludes the paper.

2 Random Indexing (RI)

In Random Indexing each distinct word in a document is assigned a unique, randomly generated vector called the index vector of the word. These index vectors are sparse, high-dimensional, and ternary. An index vector consists of a large number of 0s and a small number (ε) of ±1s. Each element of it is allocated one of these values with the following probabilities:

\[
\begin{align*}
   +1 & \text{ with probability } \frac{e}{2d} \\
   0 & \text{ with probability } \frac{d-e}{d} \\
   -1 & \text{ with probability } \frac{e}{2d}
\end{align*}
\]  

(1)

where \(d\) is the dimension of the index vector, and \(\varepsilon \ll d\). For a given document an appropriate value of \(d\) is to be chosen depending upon the number of distinct words in the document.

Using RI a document is represented as a set of vectors in the Word Space with the underlying assumption that the intended semantics of a word in a document can be found from its context words in the document. The context vector of each word in the document is created by adding the index vectors of all the words occurring within a specified context of the focus word. In this respect we found that random positioning of ‘+1’s and ‘−1’s may result in creating two index vectors having their respective ‘+1’ and ‘−1’ occurring at the same coordinate position. This in turn creates erroneous context vectors as the ‘+1’ and the ‘−1’ cancel each other. To avoid this situation we restrict the placement of ‘+1’s in the upper half positions, and ‘−1’s in the lower half positions of an index vector. This ensures that the semantics is not misrepresented in the context vector.

---

3. Random Indexing based summarization (RISUM)

Given a document the key steps for application of RISUM to create its extractive summary are the following:

- Representation of the sentences of the document in the Word Space.
- Construction of the Proximity Graph (of the sentences) for the document.
- Extraction of the summary from the graph.

Below we describe these three steps in detail.

3.1. Representing the document in the Word Space

The aim of this step is to create vectors representing the sentences of the document. This is done in four stages:

Pre-processing: At this stage the content words of the document are identified by using a list of stop words. We used ‘smart_common_words.txt’ file consisting of 598 stop words for this purpose. Then Porter Stemming Algorithm (Porter, 1980) is used to remove the common morphological and inflectional endings from the words of the document.

Forming index vectors: Each distinct word of the document is allotted an index vector in the \( d \)-dimensional vector space according to the scheme explained in Section 2. For the present work we have created index vectors with one ‘+1’ and one ‘−1’, and the dimension of the index vector is determined dynamically by using the formula \( d_{RI} = 2 \lceil \sqrt{m} \rceil \), where \( m \) is the number of distinct words in the document, and \( \lceil \rceil \) stands for the ceiling function. In fact \( d_{RI} \) is the minimum dimension that is required to accommodate all the distinct words of the document (Appendix C provides a justification of the formula). For illustration, if a document contains 500 distinct words, and the index vectors are constructed with one ‘+1’ and one ‘−1’, and the placement of ‘+1’ is restricted to first half positions and placement of ‘−1’ is restricted to second half positions of index vectors, then, the dimension of the index vectors will be \( d_{RI} = 2 \lceil \sqrt{500} \rceil = 46 \). In fact, with \( d = d_{RI} = 46 \), a maximum of 529 (23 \( \times \) 23) distinct words can be accommodated.

Construction of context vectors: Context vectors are formed for each content word occurring in the document. For RISUM, we define the context of a word with the help of a bi-directional window of size 2 (henceforth denoted by \( 2 + 2 \) window) restricted to the sentence in which the word has occurred. Initially the context vector of each content word in the document is initialized to \( d_{RI} \)-dimensional null vector. We denote \( i \)th content word of \( j \)th sentence as \( w_{i,j} \).

The context vector of \( w_{i,j} \) denoted by \( C(w_{i,j}) \) is computed using Eq. (2):

\[
C(w_{i,j}) := C(w_{i,j}) + \sum_{k = -2}^{2} 2^{1-|k|} I(w_{i+k,j})
\]

(2)

where \( I(w_{i,j}) \) denotes the index vector of the word \( w_{i,j} \); and \( 2^{1-|k|} \) is the weighting factor of a word based on its proximity (in a given sentence) to the content word under consideration. The context vector in a sense is a cue to the inherent semantics of a word derived from the words co-occurring with it in the sentence in which it occurs. Since a content word in a document co-occurs with a set of words in one sentence, and with a different set of words in another sentence, the context vector of the same content word may vary from sentence to sentence.

Mapping the sentences into the Word Space: All the sentences of the document are finally mapped into the Word Space with the help of the context vectors of their constituent content words. The sentence vector \( S(s_j) \) of the \( j \)th sentence is obtained by using Eq. (3):

\[
S(s_j) = \frac{1}{m_j} \sum_{i=1}^{m_j} (C(w_{i,j}) - O)
\]

(3)

where \( m_j \) is the number of content words in the \( j \)th sentence. Here \( O \) denotes the central theme of the document computed as the arithmetic mean of context vectors of all the content words of the document.

---

\footnote{ROUGE (Lin, 2004), the evaluation metric used by DUC also uses the same file for listing stop words.}
While forming the context vectors we have focussed on the content words only, as context vectors of stop words have negligible influence on the semantics derived from the words of a sentence. However, we found that stop words are important for determining the context of a word occurrence. Therefore, while forming the context vector of a content word, all the words within its context window (irrespective of whether it is a stop word or content word) are considered. This in turn calls for a need to subtract the mean vector from the context vectors. Subtraction of the mean vector reduces the magnitude of those context vectors which are close in direction to the central theme of the document, and increases the magnitude of context vectors which are almost in the opposite direction from the central theme. This reduces the influence of the commonly occurring words, such as the auxiliary verbs and articles, on the sentence vector (Higgins and Burstein, 2007).

3.2. Representing the document as a proximity graph

Once the sentences are mapped into the Word Space, the entire document is represented as a proximity graph. The nodes of the graph represent the sentences of the document, and weighted edges represent the proximity between sentences. In our scheme, we experimented separately with two fundamental ways of computing proximity between sentence vectors, viz. angular proximity and linear proximity.

(a) Angular proximity is measured in terms of the angle between the two sentence vectors in the $d_{RI}$-dimensional vector space. For angular proximity we have used cosine dissimilarity measure given by:

$$C_{ij} = 1 - \frac{\sum_{k=1}^{d_{RI}} s_{ik} \cdot s_{jk}}{\sqrt{\sum_{k=1}^{d_{RI}} s_{ik}^2} \cdot \sqrt{\sum_{k=1}^{d_{RI}} s_{jk}^2}}$$ (4)

where $C_{ij}$ is the cosine dissimilarity between $i$th and $j$th sentence and $S(s_p) = [s_{p1} s_{p2} \cdots s_{pd}]$ is the $p$th sentence vector.

(b) Linear proximity is measured in terms of the Euclidean distance between the two sentence vectors in the $d_{RI}$-dimensional vector space using the formula given by:

$$e_{ij} = \sqrt{\sum_{k=1}^{d_{RI}} (s_{ik} - s_{jk})^2}$$ (5)

where $e_{ij}$ is the Euclidean distance between $i$th and $j$th sentence.

3.3. Summary generation

The edge weight $\omega_{ij}$ of the weighted graph shows the relatedness between the $i$th and $j$th nodes of the document. Depending upon the proximity measure used (Cosine or Euclidean) one gets different edge weights, and hence different proximity graphs. Weighted PageRank algorithm (Mihalcea and Tarau, 2004) is now used to determine the importance of a sentence in the whole document. If $G = (V, E)$ is an undirected graph with the set of nodes $V$ and set of edges $E$, then the weighted PageRank of a node $v_i$, denoted by $PR^W(v_i)$ is defined as:

$$PR^W(v_i) = (1 - \tau) + \tau \sum_{v_j \in \{v_i : \{v_i, v_j\} \in E\}} \frac{\omega_{ij} PR^W(v_j)}{\sum_{v_k \in \{v_i : \{v_i, v_k\} \in E\}^G} \omega_{ik}}$$ (6)

where $\omega_{ij}$ is the weight associated with the undirected edge $\{v_i, v_j\}$ and $\omega_{ij} = \omega_{ji}$ for all $i, j$; and $\tau$ is a parameter chosen between 0 and 1. In our experiments we set $\tau$ at 0.85 as per the recommendation of Brin and Page (1998). Iterative application of the weighted PageRank algorithm on the proximity graph makes the node weights converge. In case of cosine dissimilarity measure, nodes with higher weights are considered for summary generation; while in case of Euclidean distance measure lighter nodes are picked up for the summary.

The summarization scheme RISUM has an inherent drawback. While computing the context vector of a word $w$, it does not take into consideration the word itself. This may result in some identical context vectors for words which are semantically very different from each other. For example consider the two sentences:
(i) Police arrested the man.
(ii) Police awarded the man.

Although the words ‘arrested’ and ‘awarded’ are semantically (almost) opposite, they have the same context vector in the standard formulation of RI. This problem occurs because the index vectors of ‘arrested’ and ‘awarded’ play no role in the construction of their respective context vectors. Therefore, for the scheme RISUM+ we initialize the context vector of a focus word to its index vector (instead of the \(d_{RI}\)-dimensional null vector), and update it by using Eq. (2). This allows us to distinguish between the context vectors of two words which are semantically very different but may occur in the same context.

Like RISUM, in RISUM+ too one can use both cosine and Euclidean distances. Table 1 gives a comparison of the outcomes of RISUM and RISUM+ on DUC 2002 documents. Bold numbers represent the best recall scores in each category (10% and 50%) with respect to the corresponding ROUGE evaluations. It is clear from Table 1 that RISUM+ performs only marginally better than RISUM. This prompts us to investigate with the other method for computing context vectors. As mentioned in Section 1, we have used convolution for this purpose. The resulting scheme MRISUM is discussed in Section 4.

### 4. Modified Random Indexing based summarization (MRISUM)

Our motivation to use convolution in text summarization is influenced by the successful application of this powerful mathematical technique in various scientific applications, such as: electrical engineering (Senior, 1986), speech acoustics (Harrington and Cassidy, 1999), image processing (Castleman, 1996). Moreover, convolution as a technique has already found its application in Natural Language Processing (NLP), most of these studies are on ‘holographic representation of lexicon’ (Metcalf, 1990; Plate, 1995; Jones et al., 2006; Jones and Mewhort, 2007; De Vine and Bruza, 2010). The convolution operation on finite vectors can be defined as follows.

If \(\mathcal{U} = [u_1 \ u_2 \ \cdots \ u_m]\) and \(\mathcal{V} = [v_1 \ v_2 \ \cdots \ v_n]\) are two vectors of dimension \(m\) and \(n\) respectively, then their convolution \(\mathcal{U} * \mathcal{V}\) is a vector \(\mathcal{W} = [w_1 \ w_2 \ \cdots \ w_{m+n-1}]\) of dimension \(m+n-1\) defined as follows:

\[
w_i = \sum_j u_j \cdot v_{i-j+1}
\]

where \(1 \leq i \leq m + n - 1\) and \(j\) ranges over all legal subscripts for \(u_j\) and \(v_{i-j+1}\), specifically \(\max(1, i + 1 - n) \leq j \leq \min(i, m)\).
In our proposed scheme MRISUM, initially context vector of each word in the document is initialized to the \((2d_{RI} - 1)\)-dimensional null vector. The context vector of \(i\)th content word of \(j\)th sentence is updated using Eq. (8):

\[
\mathcal{C}(w_{i,j}) := \mathcal{C}(w_{i,j}) + \sum_{k=2}^{2} 2^{-|k|} [I(w_{i,j}) \ast I(w_{i+k,j})]
\]

Once the context vectors are generated the summary generation proceeds in the same way as RISUM.

5. Experimental results

DUC 2002 corpus has been used as experimental data set for comparing the performance of the three systems (RISUM, RISUM+ and MRISUM) with the RI-Baseline and LSA+TRM\(^4\) systems. The DUC 2002 corpus consists of 567 newswire documents on 59 different topics. DUC 2002 is the last version of DUC that includes evaluation of single-document generic summaries. For our experiment we have chosen 90 documents to cover all different topics and sizes. Among the 90 documents 30 are very short (<16 sentences), 30 are of medium size (16–25 sentences), and rest 30 are large documents (more than 25 sentences). The smallest document consists of 7 sentences and the largest document consists of 64 sentences. We produced short summaries (10% of number of sentences) and long summaries (50% of number sentences) for each document. Each document of DUC 2002 corpus is accompanied with two different abstracts created manually by professionals. These summaries serve as the ‘gold’ summaries of the corresponding document.

RI-Baseline system uses cosine similarity measure for proximity computation. In order to circumvent the divergence that it would result in (as mentioned in Section 1) we replaced the negative edge weights with the smallest positive number \(eps(2.2204 \times 10^{-16})\) available in MATLAB.\(^5\) Further, for the sake of performance comparison, we used a variant of the RI-Baseline where the Euclidean distance measure is used instead of cosine similarity measure. In the subsequent discussions this resulting system has been denoted as RI-Euclidean. Table 1 shows the performance of RI-Baseline, RI-Euclidean, RISUM and RISUM+ under different ROUGE (Lin, 2004) measures for both 10% and 50% summaries. In particular, we used the following ROUGE metrics:

- ROUGE-1 (unigram recall measure),
- ROUGE-2 (bigram recall measure),
- ROUGE-3 (trigram recall measure),
- ROUGE-4 (4-gram recall measure),
- ROUGE-L (longest common subsequence recall measure),
- ROUGE-S\(^*\) (skip bigram recall measure), and
- ROUGE-SU\(^*\) (skip bigram with unigram recall measure).

For each of the three summarizers (RISUM, RISUM+ and MRISUM), we have considered both cosine dissimilarity and Euclidean distance measures. The observations are as follows:

- RI-Baseline is the poorest among all the summarizers in terms of all the above ROUGE measures.
- RI-Euclidean performs better than RI-Baseline both for 10% and 50% summaries.
- The performance of RISUM (Cosine) is much improved in comparison with RI-Baseline. Also, RISUM (Euclidean) performs better than RI-Euclidean.
- RISUM (Euclidean) performs much better than RISUM (Cosine) in 10% category. But RISUM (Cosine) performs marginally better in 50% category.
- RISUM+ (Cosine) and RISUM+ (Euclidean) perform slightly better than their corresponding RISUM scheme.

\(^4\)The dimension reduction ratio (DR) is set to 0.6 and 0.7 (i.e., if the rank of the singular value matrix is \(n\), then only the first \(DR \times n\) of diagonal entries are kept for semantic matrix reconstruction).

\(^5\)The summarization algorithms were implemented in MATLAB Version: 7.14.0.739 (R2012a).
Table 2 shows the performance of RISUM, RISUM+, MRISUM and LSA+TRM. Note that the dimension of a context vector in MRISUM is \((2d_R - 1)\), because of the underlying convolution operation. In order to nullify the effect of the increased dimension used in MRISUM, here we used RISUM and RISUM+ with dimension of index vectors \(2d_R\). Bold numbers represent the best recall scores in each category (10% and 50%) with respect to the corresponding ROUGE evaluations. Our observations are as follows:

- There are almost no improvements in recall scores of RISUM and RISUM+ when the dimension of index vectors is doubled.
- MRISUM gives significantly better performance than RISUM and RISUM+ even when their dimensions are at par with MRISUM.
- MRISUM (Euclidean) performs much better than MRISUM (Cosine) in 10% category, while in 50% category MRISUM (Cosine) performs marginally better.
- MRISUM (Euclidean) also outperforms LSA+TRM summarization technique in 10% category. In 50% category MRISUM (Cosine) performs best in all ROUGE evaluations except in ROUGE-1 and ROUGE-L, where LSA+TRM with DR = 0.7 performs marginally better.

In our experiments over 90 DUC 2002 documents the average number of words encountered per document is 252 (Max: 480, Min: 125). Therefore, in LSA the average dimension of unary vectors for construction of initial word-by-sentence matrix is 252. This results in the average dimension of initial word-by-sentence matrix to be 252 \(\times n_{avg}\), where \(n_{avg}\) is the average number of sentences per document. But in RISUM, RISUM+ and MRISUM the average dimension of index vectors is 32 (Max: 44, Min: 24). Due to convolution the average dimension of context vectors in MRISUM is increased to 63 (Max: 87, Min: 47), which is almost double of the dimension of the context vectors used in RISUM and RISUM+. Thus RISUM, RISUM+ and MRISUM deal with index vectors of dimension much smaller than the number of words in the document. This is in sharp contrast of techniques based on LSA where the number of rows of the initial word-by-sentence matrix is exactly same as the number of distinct words in the document. As LSA uses SVD which is computationally expensive, the larger is the document the costlier is the computation of LSA-based techniques. However, our experiments suggest that MRISUM has a superior performance in terms of the ROUGE metrics as given in Table 2. MRISUM therefore appears to be an efficient approach for single document summarization, and warrants further exploration.

Since in RI index vectors are created randomly, the output summary of a document may vary with every run of an algorithm. The results shown in Tables 1 and 2 for RI-Baseline, RI-Euclidean, RISUM, RISUM+ and MRISUM are the average ROUGE scores over 100 experiments with each of the schemes. Figs. 1 and 2 depict the standard deviation of
ROUGE scores in 10% and 50% summaries, respectively. RISUM gives better consistency than RI-Baseline; whereas RISUM+ gives marginally better consistency than RISUM. MRISUM has been found to be the most consistent one. Also, it has been observed that Euclidean distance measure, in general, performs more consistently in comparison with cosine similarity/dissimilarity measures. Quite expectedly long summaries are more consistent than short summaries. This is because a document may contain a number of equally good candidate sentences for a summary; a short summary can select only some of them, whereas most of the good candidate sentences qualify for a long summary.

Our experiments were conducted on a INTEL(R) Core i5 650@350 GHz Processor, RAM 8 GB, 64-bit operating system loaded with Windows 7 Professional Service Pack 1. The summarization algorithms were implemented in MATLAB Version: 7.14.0.739 (R2012a). The run time of each algorithm over 90 documents is shown in Table 3. Although convolution is generally perceived as a computation intensive algorithm, the run time of MRISUM is only marginally higher than the run times of other RI-based algorithms. The run time of LSA+TRM is slightly higher than the run time of MRISUM.
Fig. 2. Standard deviation of performance of summarizers in 50% category.

Table 3
Run time of summarizers.

<table>
<thead>
<tr>
<th>Summarizer</th>
<th>Run time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RI-Baseline</td>
<td>348.838</td>
</tr>
<tr>
<td>RI-Euclidean</td>
<td>348.813</td>
</tr>
<tr>
<td>RISUM (Cosine)</td>
<td>348.839</td>
</tr>
<tr>
<td>RISUM (Euclidean)</td>
<td>348.812</td>
</tr>
<tr>
<td>RISUM+ (Cosine)</td>
<td>348.841</td>
</tr>
<tr>
<td>RISUM+ (Euclidean)</td>
<td>348.815</td>
</tr>
<tr>
<td>MRISUM (Cosine)</td>
<td>365.413</td>
</tr>
<tr>
<td>MRISUM (Euclidean)</td>
<td>365.107</td>
</tr>
<tr>
<td>LSA+TRM</td>
<td>388.231</td>
</tr>
</tbody>
</table>
6. Conclusion and future work

This paper proposes three extractive summarization techniques, RISUM, RISUM+ and MRISUM for single document summarization. While in RISUM context vectors are constructed using weighted sum of index vectors of content words, in RISUM+ the index vector of the focus word is also taken into account to construct the context vector. In MRISUM the context vectors are formed by convolution of the index vectors of the content words with the focus word. For all the three techniques we used angular measure (cosine dissimilarity) and linear distance measure (Euclidean distance) separately. The outputs of all the three systems are found to be better than the RI-Baseline system. Although the performance of RISUM and RISUM+ are almost comparable, MRISUM gives a significantly better performance. However, MRISUM is computationally somewhat costlier than RISUM and RISUM+.

All the three summarizers perform more consistently with the Euclidean distance measure. For short summaries MRISUM (Euclidean) performs much better than MRISUM (Cosine). For long summaries although the average scores of MRISUM (Cosine) are marginally better than MRISUM (Euclidean), the latter’s performance is much more consistent when observed over 100 runs on the same document. Also, MRISUM’s performance is superior to computationally complex method LSA+TRM.

The consistent performance of RI-based summarization schemes with Euclidean distance measure over cosine dissimilarity measure encourages us do further research on the behaviour of these proximity measures in conjunction with RI-based schemes. In future, we would like to establish a mathematical justification of this behaviour. Further, we like to develop a scheme that will be an aggregation of both the proximity measures. We expect that such a scheme will perform effectively and efficiently for both long and short summary generation. We also like to extend the schemes for multi-document summarization.

Appendix A. Reason for using restricted index vectors

Consider the following sentence: “A wise man will make more opportunities than he finds”. Let the dimension $d$ of the index vector be 10, and the context be defined as one preceding and one succeeding word. Then the context of ‘man’ is ‘wise’ and ‘will’. If ‘wise’ and ‘will’ are assigned index vectors [0 0 1 0 0 −1 0 0 0] and [0 0 0 0 1 0 0 0 −1 0] respectively, then the context vector of ‘man’ is [0 0 1 0 1 −1 0 0 −1 0]. Instead, if ‘wise’ and ‘will’ are assigned index vectors [0 0 1 0 0 −1 0 0 0] and [0 −1 0 0 1 0 0 0 0], then the context vector of ‘man’ becomes [0 0 0 0 0 1 0 1 0 0]. Suppose ‘more’ and ‘than’ are assigned index vectors [−1 0 0 0 0 0 0 0 0 0] and [1 0 0 −1 0 0 0 0 0 0], then the context vector of ‘opportunities’ becomes [0 0 0 0 0 0 0 0 0 0]. Surprisingly, the words ‘man’ and ‘opportunities’ have different context words but their context vector is the same zero vector. This situation occurred because index vectors are generated randomly, and there is no restriction on placements of ‘+1’s and ‘−1’s. In a long text passage several such cases may arise where different words occurring in different contexts may be represented by the zero vector only, hence the semantic of words will be largely misrepresented. To overcome such a situation, in this paper we propose to restrict the placement of ‘+1’s in the first half positions and ‘−1’s in the second half positions of index vector, and thus the dimension of index vector may be suitable chosen as an even number. This will ensure that no context vector becomes the zero vector by any means. We would like to state that the original RI technique which is used by Chatterjee and Mohan (2007) has no such restriction in place for construction of index vectors.

Appendix B. Reasons for using dissimilarity measures instead of similarity

Although proximity is typically measured in terms of similarity, our preference for dissimilarity measure over similarity measure can be attributed to the following reasons.

Chatterjee and Mohan (2007) represented the proximity between two sentences using cosine similarity measure, which is defined as:

$$c_{ij} = \frac{\sum_{k=1}^{d}s_{ik} \cdot s_{jk}}{\sqrt{\sum_{k=1}^{d}s_{ik}^2} \cdot \sqrt{\sum_{k=1}^{d}s_{jk}^2}} \tag{B1}$$

If cosine similarity measure is used to compute proximity, the edge weight between any two nodes lies in the interval $[-1, 1]$. There are situations where due to a mixture of positive and negative edge weights, the PageRank value of
every node diverges to either $+\infty$ or $-\infty$. For example, if the weighted PageRank algorithm with $\tau = 0.85$ is applied iteratively on a 3-node undirected graph, whose

(i) edge weights are: $\omega_{12} = \omega_{21} = -0.5$, $\omega_{13} = \omega_{31} = 0.9$, and $\omega_{23} = \omega_{32} = 0.7$;

(ii) initial PageRank of all the three nodes equal to 0.1;

then \( PR^W(v_1) \rightarrow -\infty, \ PR^W(v_2) \rightarrow \infty, \) and \( PR^W(v_3) \rightarrow \infty. \) This inherent drawback is avoided by using cosine dissimilarity measure (Eq. (4)), which confines edge weights in the interval \([0, 2]\).

There is a pragmatic aspect of using cosine dissimilarity as well. Often it is found that a document contains several different aspects. As it is passed through a summarization process only some of these aspects are highlighted in the summary. The diversity of the contents is thus often ignored. Application of a similarity measure between sentences tends to capture sentences having commonality of themes. This in turn makes the extracted summary reflect one single aspect of the document. On the other hand, a dissimilarity measure is often able to collect all different facts lying in a single document.

Appendix C.

**Theorem:** Suppose random index vectors are constructed with one ‘+1’ and one ‘−1’; the ‘+1’ being restricted in the upper half positions and the ‘−1’ being restricted in the lower half positions of the index vector. Then to create \( m \) distinct index vectors, the minimum dimension of the index vector should be \( 2 \lceil \sqrt{m} \rceil \), where \( \lceil \cdot \rceil \) stands for the ceiling function.

**Proof.** Since ‘+1’ is restricted to the upper half positions and ‘−1’ restricted to the lower half positions of the index vector, the dimension of the index vector \( d \) can be taken as an even number. Suppose \( d = 2k \), for some \( k \in \mathbb{N} \). The ‘+1’ can be placed any of the \( k \) ways in the first \( k \) places of the index vector. Similarly, the ‘−1’ can be placed any of the \( k \) ways in the last \( k \) places of the index vector. Since the placements of the ‘+1’ and the ‘−1’ are independent of each other, the total number of index vectors can be formed in this kind of placements is \( k^2 \).

Suppose \( m \) distinct index vectors are to be created. Then \( k^2 = m \), implying \( k = \lceil \sqrt{m} \rceil \). Here \( \lceil \cdot \rceil \) is used to round up \( \sqrt{m} \) to a positive integer. Therefore, \( d = 2 \lceil \sqrt{m} \rceil \).

Suppose, if possible, \( d \) is not the minimum dimension of the index vectors. Then there exists a dimension \( d_1 \) which can create \( m \) distinct index vectors. Suppose \( d_1 = d - 2 \). Hence, \( d_1 = 2 \lceil \sqrt{m} \rceil - 2 \). The number of distinct index vectors can be created with dimension \( d_1 \) is \( (\lceil \sqrt{m} \rceil - 1)^2 \). Since \( (\lceil \sqrt{m} \rceil - 1)^2 < m \), \( d_1 \) cannot be the minimum dimension. Hence, \( d = 2 \lceil \sqrt{m} \rceil \) is the minimum dimension of the index vectors which can create \( m \) distinct index vectors, when the ‘+1’ is restricted to the upper half positions and the ‘−1’ restricted to the lower half positions of the index vector.

References


Niladri Chatterjee received his M.Stat. and M.Tech. degrees both from Indian Statistical Institute, Kolkata, India in 1984 and 1986 respectively. He received his Ph.D. degree in Computer Science from London University in 1995. During 1996–1998 he worked as a Postdoctoral Research Fellow in the Department of Computer Science, University College, London. Since 1999 he has been working as a faculty in the Department of Mathematics, Indian Institute of Technology Delhi, India. His research interests are natural language processing, statistical modelling, and semantic web.

Pramod Kumar Sahoo received his M.Sc. and M.Phil. degrees in Mathematics both from Sambalpur University, India in 1997 and 1998 respectively. Since 2003 he has been working as a Scientist in Institute for Systems Studies & Analyses, Delhi, India. Currently he is pursuing his Ph.D. in Mathematics at Indian Institute of Technology Delhi, India. His areas of interests are natural language processing, fuzzy logic, and systems analysis.