ISSN 1870-4069

A Formal Technique for Text Summarization from Web Pages by using Latent Semantic Analysis

J. Guadalupe Ramos[1], Isela Navarro-Alatorre[1], Georgina Flores Becerra[2], Omar Flores-Sánchez[2]

[1] Tecnológico Nacional de México, Instituto Tecnológico de La Piedad, Michoacan, Mexico

[2] Tecnológico Nacional de México, Instituto Tecnológico de Puebla, Puebla, Mexico

jgramos@pricemining.com, isela.na@piedad.tecnm.mx, kremhilda@gmail.com, omar.flores@itpuebla.edu.mx

Abstract. Web is the more attractive media for information consulting of, practically, whatever theme; humanity considers the Web, in the facts, the standard source of information. However as content grows, effort for discriminating and filtering increases too. Orthogonally, users employ each time smaller devices with reduced screens for web reviewing. Both considerations suggest the neediness of software tools for information acquiring and reduction, i.e., text summarization. There are several methods for text summarization, however, majority of them are based on techniques who considere plain documents in contrast with tree like structures of web pages, other are settled on the existence of keywords ignoring relations among words. In this work we present a formal method for the preparation of text summaries based on latent semantic analysis (LSA), which exploits the implicit relationships between the words that appear in a common context. In this way, text summaries are enriched with a certain semantic flavor incorporated by LSA. Furthermore we prepare the text summary induced by the query of an user and retrieving text excerpts more semantically similar to user's interest. Additionally we define a formula called semantic similarity which encapsulates the properties of LSA and determines the best text web page node for producing summaries.

Keywords. Latent semantic analysis, LSA, summarization, web pages.

1 Introduction

Automatic summarization from web pages has many clear motivations, by one side, available information in the Web is in constant growing, and in the other side, the most popular device for web consulting is the smartphone, which disposes of very small screens as the main interface for user interaction, this is obviously a non comfortable interface. In this sense, summarization, filtering and reduction of information techniques are required in order to choose only the meaningful information to be presented to an user. For text reduction from web pages there are not many approaches, we could consider that mature techniques for automatic summarization (such as those surveyed in [3]) could be employed. However those approaches lack of some aspects of our interest. For instance [3] considers procedures which do not use the tree like structure implicit in a web page.

We consider than tree like structure is important because can be employed for transforming a web page in a fragment of itself (by modifying tree structure), what could be read easily in a small device, this is called *filtering* [8]. Scheme of [8], employs techniques which are more based on keywords appearing in documents, i.e., the statistical information of co-occurrences between terms in a collection of documents, which can be gathered by formal methods like for instance *Latent Semantic Analysis (LSA)*, is not taken into account.

Certainly there are many works in the literature like [11, 5] and cited there, who produce automatic text summaries, inclusive by using LSA, those methods treat the text source as plain documents. However, web text is formatted using (X)HTML, i.e., the information is organized in formatting nodes which frequently provide certain implicit unity (all the information in a node is related) imposed by the web designer. In this way, main inspiration of this work is going to be established in the context of approaches that exploit the text fragments within a web page, for instance those devoted to web filtering such as [2, 8]. Regardless this focus, formulas here developed for calculus of semantic similarity among text fragments can be applied in a seamless way to any pair of text fragments, and thereby we could produce summaries from sentences instead of web page text fragments.

We consider that methods preserving the unity of (X)HTML nodes are more acceptable. Some works following this idea are, for instance [6, 15, 13, 2]. Indeed, in [2] a method for information extraction from web pages considering the distance between (X)HTML nodes is introduced. However standard tests of similarity as a basis for producing web summarization are not employed in [2], we believe that it is necessary to test classical techniques in order to establish an adequate comparison framework.

In this work we present a formal method for automatic creation of text summaries from a set of URLs, considering a web query, based on techniques of similarity employed in natural language processing and inspired on the notion of latent semantic analysis. Our formal method requires only once the calculations of LSA and then it can prepare a summary based on any web user query.

The rest of the paper is organized as follows. In Section 2, we overview standard concepts of vector representation of text in natural language and latent semantic analysis. In Section 3, we introduce a formal technique for text extraction based on semantic similarity. Then, in Section 4, we describe a set of experiments. Finally, we present our conclusions in Section 5.

2 Vector Space and Latent Semantic Analysis

In this subsection we introduce a formal standard representation for text documents written in natural language. The vector space model for automatic indexing was originally introduced by Salton et al., in [14] and it is considered a standard representation technique in information retrieval setting, where stored entities (documents) are compared with each other.

Given a text document d, a dictionary of *terms* is a set whose elements are the different *words* in the document d. Formally:

Definition 1 (dictionary). *Given a text document* d *let the set* $dict(d) = \{t_0, \ldots, t_{n-1}\}$ *be the dictionary of* d*, where* $\{t_0, \ldots, t_{n-1}\}$ *are the* n *terms in* d*.*

$\overrightarrow{V}(d)$ denotes the vector associated to document d, whose components are the weights for each element in the dictionary. It is assumed that element weights are computed using the *tf* weighting scheme, i.e., the value of a particular component is given according to the number of times the corresponding word occurs in document d. The set of documents in a collection then may be viewed as a set of vectors in a vector space, in which there is one axis for each term. This representation loses the relative ordering of the terms in each document [9]. In this view of a document, known in the literature as the *bag of words model*, the ordering of the terms in a document is ignored but the number of occurrences of each term is considered. Nevertheless, it seems intuitive that two documents with similar bag of words representations are similar in content [9].

Example 1. Given $text$ = "Web sites, Web services or Web-based applications", the dictionary (without lower-upper case distinction) is composed by {web, sites, services, based, applications}, then $\overrightarrow{V}(text)$ is $\langle 3, 1, 1, 1, 1\rangle$. Here the term Web appears 3 times, sites 1, and so on.

In Example 1 there are typographic symbols such as "," or "-", regularly this kind of text elements are ignored when a vectorial representation is prepared. A convenient set of particular terms is treated in the same way, in the example the term "or" was ignored, since words in a document are not equally important, i.e., some extremely common words provide little value in helping to distinguish the meaning of a text. These words are called *stop words*. The steps in natural language processing for analyzing a text document are the following.

- Filtering: the subject text must be filtered, dropping typographic symbols, for instance: ",.:;?".
- Removing stop words: those terms which do not provide meaning must be removed.
- Preparing the bag of words: the set of different terms is gathered, i.e., the dictionary properly.
- Vector representation: following the terms in the dictionary a vector is computed for each document.

Additionally, for web documents a step for removing format labels is mandatory.

2.1 Latent Semantic Analysis

Latent Semantic Analysis (LSA) is a theory and technical method for extracting and representing the contextual-usage meaning of words by means of statistical computations applied to a large corpus of text [4]. Hence, the underlying idea is that the aggregate of all the word contexts in which a given word does and does not appear, provides a set of mutual constraints that largely determines the similarity of meaning of words and sets of words to each other.

LSA transports the statistical findings from heap of documents and gathers numerical information in a convenient model to be exploited in methods of natural language processing. Basically numerical information is useful to reason about the likeness of terms in the same textual space.

First step of LSA consists of the construction of a matrix representation of text, i.e., the matrix M, in which columns are employed for modeling documents and rows for terms (words). Each row i represents a specific term as well as each column j represents a document.

Thus, each cell $M_{i,j}$ stands for the frequency in which every term i appears in the document j. Term frequency tf can be substituted by some other scheme for measuring the significance of each term in a document, which is called term weighting scheme. A common weighting scheme for terms is for instance $tf.idf$ [9].

Next, LSA peforms the *Singular Value Decomposition* process (SVD) on the matrix M. SVD is the core of LSA, is a standard technique which is applied in linear algebra over matrixes, it is a specific form of factorial analysis.

In the original matrix M terms and documents are mutually dependent between them. In SVD, a rectangular matrix M is decomposed in the product of other three matrixes, i.e., $M = USV^T$. New matrixes will be formed by *singular vectors* or *singular values*. Resultant matrix U will contain a vector representation of the terms, which will have linear independency w.r.t. the relationship with the documents, while V will contain the vector representation of the documents whose components will be linearly independent w.r.t. the relationships with terms in M. Finally S is a diagonal matrix in which singular values are found in descendent order, and they represent the relationships between the other matrixes. The highest values in S represent the relations with major variance among terms and documents.

After SVD decomposition, the original matrix M can be rebuilt as of the matrix product of the resultant three matrixes. When a reconstruction over matrixes is performed it is possible to choose only the first k elements of the matrixes, i.e., $M' = U_k S_k {V_k}^T$, with this, a new matrix M' is obtained, in which the noise introduced by irrelevant relations is eliminated. Thus, the new values $M'_{i,j}$ unveil latent relationships among terms and documents, the so called human cognitive relations in [4]. The reason is that the reconstructed matrix employs the singular values representing the major variance, in this way, we gain a matrix that models the best relationships in data. In this work SVD is considered a core tool for LSA, the goal of this paper is to exploit the advantages of these formal techniques for producing technology. Hence the paper's main focus is presentation of summarization method. For a deep explanation of SVD the reader can consult, for instance [7, 4].

Example 2. Let us consider the following four sentences.

1. $d0 =$ My computer has branded software
2. $d1 =$ A PC is useful only with branded software
3. $d2 =$ PC (as computer) hardware can be generic

4. $d3 =$ Branded software and generic hardware go well with my computer

Hence, the dictionary of the document collection is {computer, software, branded, PC, hardware, generic}. According to above speech, the first row in M is for the representation of the term $computer$ (second one for software, and so on w.r.t. the dictionary) and the column 0 will be for the first document, then $M_{0,3}$ stands for the number of times that computer appears in document 3, and so on. By applying the technique SVD $M = USV^T$ is obtained:

$$M = \begin{bmatrix} 1&0&1&1 \\ 1&1&0&1 \\ 1&1&0&1 \\ 0&1&1&0 \\ 0&0&1&1 \\ 0&0&1&1 \end{bmatrix} \quad U = \begin{bmatrix} 0.49 & 0.21 & 0.35 & 0.77 \\ 0.47 & -0.50 & 0.02 & -0.17 \\ 0.47 & -0.50 & 0.02 & -0.17 \\ 0.26 & 0.14 & -0.93 & 0.22 \\ 0.35 & 0.47 & 0.08 & -0.39 \\ 0.35 & 0.47 & 0.08 & -0.39 \end{bmatrix}$$

$$S = \begin{bmatrix} 3.19 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1.74 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.19 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.6 \end{bmatrix} \quad V^T = \begin{bmatrix} 0.45 & 0.38 & 0.46 & 0.67 \\ -0.46 & -0.50 & 0.73 & 0.08 \\ 0.32 & -0.75 & -0.36 & 0.45 \\ 0.70 & -0.22 & 0.35 & -0.58 \end{bmatrix}$$

Reconstructing M and considering only the first $k = 2$ elements we obtain:

$$M' = \begin{bmatrix} 0.542 & 0.413 & 0.986 & 1.085 \\ 1.067 & 0.993 & 0.044 & 0.929 \\ 1.067 & 0.993 & 0.044 & 0.929 \\ 0.265 & 0.195 & 0.557 & 0.577 \\ 0.133 & 0.019 & 1.115 & 0.821 \\ 0.133 & 0.019 & 1.115 & 0.821 \end{bmatrix}$$

If the similarity between vectors representing the rows 0 and 3 of M is calculated, i.e., the comparison of similarity among terms "PC" and "computer", result is 0.4, while, calculation of the same operation of both vectors in M' there is a result of 0.99. This new value is interesting due to the collection uses in a seamless way "PC" and "computer". Although the coincidence of both terms is given only in one document, correlations in the rest of documents allow to unveil the major *latent* similarity. This insights of relationships will be exploited later by the method exposed in this document.

3 A Formal Technique for Text Summarization

Now, we describe the main contribution of this work, i.e., a formal extraction technique useful for producing text summaries from a set of URLs.

The goal of the technique is to find out and then to extract web page fragments (from several URLs) that present the best semantic similarity w.r.t. a given web user query. Hence a *web user query* is a fragment of text in natural language, for instance *techniques applied to Web sites, Web services or Web-based applications*, which is regularly sent towards a web search engine.

3.1 Cosine Similarity

Similarity is a typical measurement between fragments of text in natural language. The standard way of quantifying the similarity between two texts $t1$ and $t2$ is to compute the cosine similarity of their vector representations $\overrightarrow{V}(t1)$ and $\overrightarrow{V}(t2)$ [9].

Definition 2 (similarity [9]). *The similarity between fragments of text* $t1$, $t2$ *is defined by*
$sim(t1, t2) = \frac{\overrightarrow{V}(t1)\cdot\overrightarrow{V}(t2)}{|\overrightarrow{V}(t1)||\overrightarrow{V}(t2)|}$, *where the numerator represents the dot product of the vectors* $\overrightarrow{V}(t1)$ *and* $\overrightarrow{V}(t2)$, *and the denominator is the product of their Euclidean lengths. The dot product of the two vectors* v, w *is* $\sum_{i=1}^{n} v_i w_i$ *while the Euclidean length of* $t1$ *is* $\sqrt{\sum_{i=1}^{n} \overrightarrow{V}_i^2(t1)}$. n *is the maximum number of different words between* $t1$ *and* $t2$. *A total similarity is 1.*

Example 3. If $t1 =$"World Wide Web" and $t2 =$"Web sites, Web services or Web-based applications", then the dictionary for $t1$ and $t2$ as a whole is "world, wide, web, sites, services, based, applications". The vector representation is: $\overrightarrow{V}(t1) = \langle 1, 1, 1, 0, 0, 0, 0\rangle$, $\overrightarrow{V}(t2) = \langle 0, 0, 3, 1, 1, 1, 1\rangle$. Therefore $sim(t1, t2)$ is $\frac{3}{\sqrt{3}\sqrt{13}}$, i.e., 0.48.

Roughly speaking our method will take a web user query and then will compute many similarity tests between the query and the text into web pages, searching by similar information in order to produce an automatic summary.

3.2 Semantic Similarity Induced by LSA

Certainly LSA provides valuable information w.r.t. similarity between whether documents or terms. For instance, in order to attend a query, [10] applies a similarity calculus by considering a vector from the query and a vector from the document. There, a vector document is each one of the columns of M'. However a summary of web pages requires being constructed from many text pieces from different web pages. In this setting a procedure for calculation of similarity among text excerpts instead of documents is needed. Granularity is a property commonly employed for referencing the size of a piece of data. Here, granularity is going to be used for talking about the number of terms in a text fragment. Methods for automatic summarization composed by text fragments of different granularity are required. A first approach for taking into account LSA through M' and multi-granularity in text fragments is the following:

Definition 3 (relative similarity). *The relative similarity between web user query text* Q *and* f_d *, a fragment of text from a document d, is defined by :*
$relsim(Q, f_d) = \frac{\overrightarrow{V}(Q)\cdot\overrightarrow{V}(f_d)}{|\overrightarrow{V}(Q)||\overrightarrow{V}(f_d)|}$,
with $\overrightarrow{V}(f_d) = \langle v_0, \ldots, v_i, \ldots, v_{n-1}\rangle$, *for* n *terms in the collection, and* v_i *is obtained from*

$$v_i = \begin{cases} M'[i][d] & t_i \in dict(collection) \text{ and} \\ & t_i \in dict(f_d) \\ 0 & \text{otherwise,} \end{cases}$$

where t_i *is a term in the collection of documents.*

In simple words, if a text fragment f_d from document d is being analyzed, for each term i in collection we put in vector $\overrightarrow{V}(f_d)$ value 0 if that word does not appear in f_d, and we put value $M'[i][d]$ if the word appears in f_d. Practically, the corresponding values in vector $\overrightarrow{V}(f_d)$ are mapped from column d in M'.

This mapping must be pointed out, because values from M' have been adjusted by LSA and then, are more appropriate. Indeed standard cosine similarity does not produce directly proportional values than relative similarity, for instance, similarity of fragment 21 in Table 1 is the lowest w.r.t. the rest of similarity measurements, nevertheless its relative similarity is the second one ranked in that column. Let us remember that we are comparing similarities among text fragments of several granularity, this is the reason why we apply a mapping of values from document in M' to $\overrightarrow{V}(f_d)$, i.e., we do not take the whole document terms.

So far, we could use relative similarity as a main tool for comparing text fragments, however, when there are not common terms in vectors, then result produced is 0, look for instance column $relsim$ in Table1 of tests. Hence, the strategy of only relative similarity is not completely well.

We must consider that LSA gathers information w.r.t. correlations among terms in a collection. A certain kind of measurement of term behavior should be conveniently incorporated into a new scheme of comparisons among text fragments. This is going to be treated in the rest of section.

First, we require a data structure which collects measurements of similarities among terms depicted in M'. This we call *Mutual similarity matrix* $\mathcal{M}$.

Intuitively speaking $\mathcal{M}$ is a data structure which contains the similarity values between all the terms in the reconstructed matrix M'. Hence $\mathcal{M}_{i,j}$ contains the similarity value among the term i and the term j of M', and the vector of term i is composed by the values in the row i in M' and so on.

Example 4. Let us consider the Example 2, the mutual similarity matrix of M' is as follows:

$$\mathcal{M} = \begin{bmatrix} 1.000 & 0.730 & 0.730 & 0.999 & 0.920 & 0.920 \\ 0.730 & 1.000 & 1.000 & 0.692 & 0.405 & 0.405 \\ 0.730 & 1.000 & 1.000 & 0.692 & 0.405 & 0.405 \\ 0.999 & 0.692 & 0.692 & 1.000 & 0.940 & 0.940 \\ 0.920 & 0.405 & 0.405 & 0.940 & 1.000 & 1.000 \\ 0.920 & 0.405 & 0.405 & 0.940 & 1.000 & 1.000 \end{bmatrix}$$

Here $\mathcal{M}[0][3] = 0.999$ which is the result of similarity among term 0 and 3 in M', i.e., the similarity between *computer*, row 0, $\langle 0.542, 0.413, 0.986, 1.085\rangle$ and *PC*, row 3, $\langle 0.265, 0.195, 0.557, 0.577\rangle$ of M'. These numbers, referenced particularly in the collection of Example 2. Obviously the diagonal matrix is composed by only ones.

Now, in the mutual similarity matrix the information that shows explicitly the numerical similarities among terms in the collection of documents is located. Quantities in $\mathcal{M}$ are the numerical measures of relations of terms w.r.t. a whole collection of documents, however, in order to prepare a text summary, composed from diverse text pieces we require the processing of smaller fragments of text instead of complete documents. Hence our method will perform comparisons among pieces of text (queries and web page text blocks) rather than documents and including the information gathered from LSA.

To compute text comparisons we should not apply only the $relsim$ calculation because we would not take advantage of the likeness between terms. Indeed in [12] a report is prepared by considering only the presence of literal words in text fragments and ignoring relations between them, we consider this approach loses a certain level of information for summary production.

In this way, it is necessary to exploit the measurements of relations contained in $\mathcal{M}$ to compute the calculus of similarity of small fragments of text and guaranteeing better summaries. For this, we apply a simple idea: if a term $t0$ is related with term $t4$ according to $\mathcal{M}$ and, if we have a text fragment f which contains $t0$ and does not contain $t4$ then, in order to execute the calculation of similarity of f we will aggregate to the vector $\overrightarrow{V}(f)$ the values for $t0$ and also for $t4$, i.e., $\overrightarrow{V}(f) = \langle 1, 0, 0, 0, threshold \rangle$ where $threshold$ is an arbitrary value in order to identify the limit for considering a *meaningful relation* between terms.

Basically, threshold define the least numerical limit of similarity measure for considering a relation between terms as important. Each row in M' represents a term in the collection of documents, and the numerical relation between the terms t_i, t_j is gathered in $\mathcal{M}_{i,j}$. In order to incorporate meaningful relations in the vectors of text fragments we define the concept of inflated vector.

Definition 4 (inflated vector). *Let $\overrightarrow{V\triangle}(f)$ be the inflated vector of a text fragment f in the setting of a collection of documents, such that $\overrightarrow{V\triangle}(f) = \langle v_0, \ldots, v_{m-1} \rangle$ where v_i is the corresponding weight of $t_i \in dict(collection)$ in the fragment f, which is obtained from:*

$$v_i = \begin{cases} weight(t_i, f) & t_i \in dict(f) \\ threshold & t_i \notin dict(f) \text{ but there is a} \\ & \text{meaningful relation with} \\ & \text{some } t_j \in dict(f) \\ 0 & \text{otherwise}, \end{cases} \quad \text{where } i, j \in \{0, \ldots, m-1\}, m \text{ is}$$

the number of rows in M' and $weight(t_i, f)$ is a function that returns the corresponding weight of t_i in f.

When a vector is inflated, following the dimensions of the collection, not only appears the weight of those terms present in a fragment of text, threshold is incorporated in the position corresponding for terms, which are not present in the fragment, but they maintain a meaningful relation (according to the collection) with other terms present in the fragment. In other words, in an inflated vector there appear the weight values of their terms and threshold value for their meaningful relationships. Hence, let us call *inflated similarity* $inflsim(t_1, t_2)$ to the similarity between two inflated vectors.

Here, inflated similarity returns the similarity calculus of two inflated vectors, where vectors were augmented from mutual closeness among terms. However the numerical importance of cosine similarity or relative similarity computations cannot be discarded, the reason is that cosine similarity and relative similarity are affected by text dispersion, i.e., big text fragments with few coincidences of terms have as a consequence low values of similarity, while inflated similarity is affected by the number of relationships between terms in the mutual similarity matrix. It appears more coincidences of terms. In this way, we introduce semantic similarity in order to take into account properties of previous calculus.

Particularly, relative similarity presents best properties because it performs the calculus by considering LSA.

Definition 5 (semantic similarity). *Given* $t1$ *and* $t2$*, let* $semsim(t_1, t_2) = relsim(t_1, t_2) \times inflsim(t_1, t_2)$ *be the semantic similarity between text fragments* t_1 *and* t_2*.*

Fundamentally we apply the product of relative similarity as a factor of correction for inflated similarity, in two ways, in one hand, it takes advantage of LSA and in the other hand, it computes greater values in the case of coinciding terms. When relative similarity value is 0, semantic similarity result is 0 too, in this case we can substitute product by a sum of logarithms, i.e., $semsim(t_1, t_2) = log(relsim(t_1, t_2)) + log(inflsim(t_1, t_2))$, and instead of a 0 argument we can put 0.001. In this way, semantic similarity combines properties of both measurements, relative similarity is more affected by coincidence of terms in the query and analyzed fragment, takes advantage of LSA and is sensible to fragment size.

3.3 Treatment of Text Fragments from Web Pages

A convenient web page representation is needed. DOM model is an adequate scheme, it stands for Document Object Model, which is a W3C standard platform—and language—neutral interface that allows programs and scripts to dynamically access and update the content, structure and style of documents [1].

Example 5. Let us consider the following toy web page:

```
<html>
 <head>   <title> Information Retrieval </title>  </head>
 <body>
   IR stands for information retrieval
  <div>
    There are two approaches for information retrieving:
    <span> a) When metadata is present.   </span>
    <span> b) By using soft-computing techniques.
             One of them is <strong>natural language processing.</strong></span>
  </div>
 </body>
</html>
```

Its corresponding DOM representation is visualized in a graphical mode in Figure 1. We can see a web page as a tree-like data structure where each node is an (X)HTML element, i.e., a (X)HTML tag with its contained text and its attributes, furthermore, its children are the embedded (X)HTML labels. If a node contains nested (X)HTML labels, there is a relation of *embedding* between the node and its children. For instance, `div` node embeds two `span` node children.

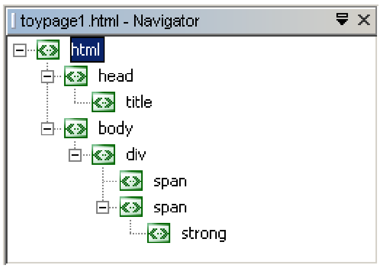

Fig. 1. The DOM tree of Example 5, visualized in a graphical way.

Table 1. Measurements for `http://lsa.colorado.edu/whatis.html` text fragments.

f	sim	$relsim$	$inflsim$	$comp$	***semsim***	terms	% terms	sem term	% sem term	text of f
1	0.14	0.137	0.336	0.046	-1.332	508	30.71	1654	100	What is LSA? What is ...
2	0	0	0.091	0	-4.039	1	0.06	123	7.43	What is LSA?
3	0	0	0.862	0	-3.064	17	1.02	1605	97.03	Note: If you linked ...
4	0	0	0.892	0	-3.050	9	0.54	1515	91.59	click here to open ...
5	0	0	0.997	0	-3.001	3	0.18	1221	73.82	The information on this page is based
6	0	0	0.867	0	-3.062	9	0.54	1605	97.03	Landauer, T. K., Foltz, ...
7	0	0	0.888	0	-3.051	4	0.24	1532	92.62	which is available for ...
8	0	0	0.780	0	-3.108	68	4.11	1654	100	Latent Semantic Analysis (LSA) is a ...
9	0	0	0.995	0	-3.002	3	0.18	1226	74.12	Latent Semantic Analysis
10	0	0	0.091	0	-4.039	1	0.06	123	7.43	(LSA)
11	0.21	0.154	0.804	0.124	-0.905	70	4.23	1654	100	Research reported in, and ...
12	0	0	0.997	0	-3.001	2	0.12	1219	73.70	semantic space
13	0	0	0.833	0	-3.079	33	1.99	1654	100	LSA can be construed ...
14	0	0	0.753	0	-3.123	72	4.35	1654	100	As a practical method ...
15	0	0	0.735	0	-3.134	76	4.59	1605	97.03	Of course, LSA, as ...
16	0	0	0.853	0	-3.069	24	1.45	1605	97.03	However, LSA as currently...
17	0	0	0.750	0	-3.125	98	5.92	1654	100	LSA differs from other...
18	0	0	0.729	0	-3.137	105	6.34	1605	97.03	However, as stated above...
19	0	0	0.874	0	-3.058	6	0.36	1579	95.46	Preliminary Details about ...
20	0	0	0.842	0	-3.075	47	2.84	1605	97.03	Latent Semantic Analysis is ...
21	0.05	0.163	0.822	0.134	-0.869	29	1.75	1605	97.03	The first step is to ...
22	0.45	0.234	0.754	0.176	**-0.752**	62	3.74	1654	100	**Next, LSA applies singular ...**
23	0	0	0.859	0	-3.066	17	1.02	1605	97.03	Landauer, T. K., ...
24	0	0	0.867	0	-3.062	4	0.24	1605	97.03	Basic and applied memory...
25	0	0	0.863	0	-3.064	15	0.90	1605	97.03	Landauer, T. K., & Dumais...
26	0	0	0.182	0	-3.740	2	0.12	400	24.18	Psychological Review, ...

4 Experiments

In this section, we describe an experiment performed upon a prototype, and correspondingly upon the formal technique. The collection was composed by text documents from a set of web pages whose links are the following.

1. `http://lsa.colorado.edu/whatis.html`
2. `https://en.wikipedia.org/wiki/Latent_semantic_analysis`
3. `http://recommender-systems.org/latent-semantic-indexing/`

Each web page was downloaded, and their DOM nodes were extracted by means of a DOM parser, then a text file with the set of nodes from each URL was prepared. The whole process of the formal technique introduced in Section 3 was computed in order to determine semantic similarity of every fragment for producing text summarization.

The launched query was: "singular value decomposition (SVD) to the matrix", for this we developed a tool that receives a query and returns each text fragment and its corresponding measurements. Summarization is constructed by taking fragments with higher semantic similarity. In the Table 1 a set of measurements are presented, there, a series of 26 text fragments f from `http://lsa.colorado.edu/whatis.html` are put through testing.

The first calculus shown is `sim` which represents the standard cosine similarity between the web user query an the analyzed fragment. The second one is the result of the relative similarity `relsim`, the third one represents the inflated similarity `inflsem`. Next column presents results of composed `comp` similarity, i.e., the product of `inflsem` and `relative`, then `semsim` is calculated by using the logarithmic approach. `semsim` unveils the result of the query, i.e., fragment 22 from the `http://lsa.colorado.edu/whatis.html` URL. For each fragment of web page the maximum semantic similarity w.r.t. the query is calculated and then the summary is produced.

A previous determination of a threshold of 0.95 was done, i.e., those relationships in the mutual similarity matrix equal or greater than 0.95 were considered important for the method. Value of threshold is thoroughly related with the experiment. Ideally, LSA should be fed with a huge stack of documents in order to harvest the more representative relationships between documents.

In the rest of columns, number of terms in the fragment is presented and their corresponding percent representation w.r.t. total terms in the collection. Next, the number of semantic terms is shown and its corresponding percent in the collection. Let us observe the percent increasing of semantic terms w.r.t. the real number of terms, this illustrates the great quantity of relationships that words implies in a text. Hence, methods which take into account the occurrence of relationships among terms are welcomed. Finally the last column shows the beginning text of the fragment.

In Table 1 text fragments with higher semantic similarity are chosen for composing the text summary.

5 Conclusions

We developed an extractive, multi web page, query-based, unsupervised technique for automatic summarization of web documents. We focused in approaches based on DOM tree structure. In this way previous works of [2, 12] were improved, for instance [12] computations would have returned 0 in the absence of common terms, here, semantic relationships provide numerical information for answer producing.

We have produced a formal technique which presents several advantages: always returns a value, is independent of the size of text fragment, privileges (numerically) the existence of common words in text fragment and query, outperforms results of cosine similarity, only once calculation of LSA is required to produce any number of summaries from a query. The formula discovered of semantic similarity can be applied to sentences of language which is useful for other kind of source documents.

The potential applications of the technique are the following: Summarization of web pages, summarization of documents, filtering of web pages (since we rank DOM nodes with semantic similarity), transformation of web pages, determining of hot sections in a web page (hot sections), production of industrial tools, and other more.

Acknowledgements. This work was partially supported by Tecnológico Nacional de México.

References

1. Arquitecture Domain, W.: Document Object Model (DOM) (2018), "Available at http://www.w3.org/DOM/"
2. Castillo, C., Valero, H., Ramos, J., Silva, J.: Information Extraction from Webpages Based on DOM Distances. In: CICLing (2). pp. 181–193 (2012)
3. Das, D., Martins, A.: A Survey on Automatic Text Summarization. Engineering and Technology 4, 192–195 (2007)
4. Furnas, G.W., Deerwester, S.C., Dumais, S.T., Landauer, T.K., Harshman, R.A., Streeter, L.A., Lochbaum, K.E.: Information retrieval using a singular value decomposition model of latent semantic structure. SIGIR Forum 51(2), 90–105 (2017)
5. Gambhir, M., Gupta, V.: Recent automatic text summarization techniques: A survey. Artif. Intell. Rev. 47(1), 1–66 (Jan 2017)
6. Gupta, S., Kaiser, G., Neistadt, D., Grimm, P.: DOM-based Content Extraction of HTML Documents. In: Proceedings of the 12th International Conference on World Wide Web. pp. 207–214. WWW '03, ACM (2003)
7. Landauer, T.K., McNamara, D.S., Dennis, S., Kintsch, W. (eds.): Handbook of Latent Semantic Analysis. Lawrence Erlbaum Associates (2007)
8. López, S., Silva, J.: A new information filtering method for webpages. In: Database and Expert Systems Applications, DEXA, International Workshops, 2010. pp. 32–36 (2010)
9. Manning, C., Raghavan, P., Schütze, H.: An Introduction to Information Retrieval. Cambridge University Press (2008)
10. N. Underhill, T.: An introduction to information retrieval using singular value decomposition and principal component analysis. `http://buzzard.ups.edu/courses/2007spring/projects/underhill-paper.pdf` (01 2007)
11. Ozsoy, M.G., Alpaslan, F.N., Cicekli, I.: Text summarization using latent semantic analysis. J. Inf. Sci. 37(4), 405–417 (2011)
12. Ramos, J.G., Campoy, L., Ruiz, S., Jasso, N., Solorio, J.: Preparing text reports from web pages employing similarity tests. In: Mexican International Conference on Computer Science, ENC 2013. pp. 13–19 (2013)
13. Ramos, J.G., Silva, J., Arroyo, G., Solorio, J.: A Technique for Information Retrieval from Microformatted Websites. Lecture Notes in Computer Science 5947/2010, 344–351 (2010)
14. Salton, G., Wong, A., Yang, C.S.: A Vector Space Model for Automatic Indexing. Commun. ACM 18(11), 613–620 (1975)
15. Silva, J.: A Program Slicing Based Method to Filter XML/DTD Documents. In: SOFSEM (1). pp. 771–782 (2007)