Procesamiento del Lenguaje Natural, Revista nº 40, marzo de 2008, pp. 115-119 recibido 08-02-08, aceptado 03-03-08

Test of Complementarity on Sentence Extraction Methods

Prueba de complementariedad para métodos de extracción de oraciones

Alberto Bañuelos Moro, José de Jesús Lavalle Martínez
B. Universidad Autónoma de Puebla
Fac. de Ciencias de la Computación
cabumtz@gmail.com, jlavallenator@gmail.com

Héctor Jiménez Salazar
Universidad Autónoma Metropolitana
Dept. de Tecnologías de la Información
hgimenezs@gmail.com

Resumen: Analizamos tres enfoques para la generación del extracto de un texto con el fin de saber si algún método provee a otro características complementarias. Se aplicaron los métodos en forma combinada para analizar sus resultados, en un marco teórico propuesto. Los tres enfoques tratados fueron los basados: en gráficas, en términos clave, y en la representatividad de las oraciones. Utilizando la colección DUC 2002, el método basado en representatividad fue el mejor. Sin embargo, no se encontraron características complementarias entre ellos, aunque a partir del análisis se identificaron algunos rasgos relevantes de los métodos.
Palabras clave: extracto automático, características complementarias de métodos

Abstract: In this work three approaches to sentence extraction methods are analyzed. We try to find if the used methods show some complementary features. In order to accomplish this goal, the methods of sentence extraction were applied and combined, analyzing the results in the theoretical framework that we propose. We test three approaches: graph-based, keyword-based and representation-based. The methods were tested using the text collection DUC 2002, obtaining the best performance for a very simple method based on representation index. Even though no complementary methods were found, the results allow to identify some relevant features of the methods.
Keywords: sentence extraction method, complementary features of methods

1 Introduction

The huge volume of available text in the web contrasts with the quantity of tools to make the growing information profitable to daily activities. Given a text, its title and its index, a summary of it could be the key to decide if the text contains valuable information. There have been many attempts to build a summary from a given text (Hovy, 2005). One of them considers to get an extract; i.e. a reduced set of sentences from the text that better represents it. This work considers the problem related to get an extract from a given text.

Getting the extract from a text has been focused on by different ways. For example, in order to choose the most representative sentences from a text, a score is assigned to each sentence based on the similarity between such a sentence and all the terms. This method gets good results and its complexity is in $O(n^2)$, where n is the number of text sentences. Another approach first identifies the "most important" terms used in the text, then the score of each sentence is computed according to the occurrence of those terms in the sentence. Finally, the score determines the representation degree of each sentence (Bueno-Tecpanecatl, Pinto, and Jiménez-Salazar, 2005). Lastly, we cite the method text-rank, which has had a high impact in Natural Language Processing (NLP) applications (Mihalcea, Tarau, and Figa, 2004). The text-rank method (Mihalcea, 2004) is derived from the algorithm page-rank, which has been used to determine the importance of a web page as a function of its relevance in the world wide web; i.e. according to the set of pages refering to, and the set of pages refered from it (Brin and Page, 1998).

Many NLP tasks have used multiple

ISSN 1135-5948

sources of information and several methods at the same time, obtaining improvements on the results. We belive it is possible to take advantage from the analysis of a set of methods, therefore we propose a methodology in order to identify strengths of the methods and the possible success of combining them. In this work we try to make clear if there are important differences among some sentence extraction methods.

In the rest of this work we deal with complementary methods concept, the used sentence extaction methods, a description of the tests carried out, and finally, a discussion on the given results.

2 Combination of methods

In this work, we apply three approaches to sentence extraction and combine some of them to observe possible relationships among them. Our goal is to analyse these approaches in order to strengthen a simple algorithm without losing their efficacy.

We stablish three possible levels to combine methods: (1) high level, joining the results of the methods; (2) middle level, combining partial results; and (3) low level, embedding one method in another one. Some examples of these levels follow. In (1), iterative algorithms which in each step refine their results may be considered; Brill's POS tagger may be seen (Brill, 1994), at least, as the application of two methods: tag assignment and correction assignment. For (2), combining of scores to choose a partial result; voting algorithms used, for instance, in text categorization (Montejo, Urena, and Steinberg, 2005). And in (3) some approaches are: *merging*, a clear example is quick-sort which can use another sort algorithm to end the recursive process; *resources*, each method works on some kind of data providing a step within the whole method, word sense disambiguation has some examples of this approach (Ng and Lee, 1996); *fusion*, in this class any improvement of an algorithm may be considered.

In our context, high level combination could require combining of sentences in a similar way to text generation do for summarization. Low level would imply formulating a new method. As we can see, middle level is the simplest one, and according to the results we can investigate other combining strategies. The power of a combination lies on the complementary character of the results of methods. Therefore, it is worthy to combine complementary methods.

We considered that two methods are complementary if each of them provides exclusive results to some extent. More formally, given two methods, M_1 and M_2, a dataset, D, and an evaluation function, E, it is considered that M_1 is better than M_2 if $E(M_1) > E(M_2)$. The result, $M_1(D)$, provided by applying M_1 to D may be compared, in a suitable scale, with the result of another method, $M_2(D)$. Considering a measure, $||$, of the results (score) we can state a combination, M of the methods: $M(D) = M_1(D)$ if $|M_1(D)| > |M_2(D)|$ or $M(D) = M_2(D)$ if $|M_1(D)| < |M_2(D)|$. Note that, in this case, M is not guarantee on the enhancement of both methods; nevertheless that $|M_1(D)| > |M_2(D)|$, we could obtain $E(M_1) < E(M_2)$. This is possible if M_1 has a decision criteria from which false positive cases take advantage. Besides, if congruence is obtained, $|M_1(D)| < |M_2(D)|$ then $E(M_1) < E(M_2)$, the methods would be complementary and, we can rise their performance: $E(M) \geq E(M_1)$ and $E(M) \geq E(M_2)$, i.e. a significative improvement, thus M_1 and M_2 are considered complementary.

We are interested in knowing if methods based on different strategies have inherently different results. This fact may conjecture if they are complementary, whenever combining their results there exists a significative improvement. Now, we give an overview of the applied techniques.

3 Sentence Extraction Methods

In this section we give some details on the used methods. Let T be a text and $[o_1, \ldots, o_n]$ the sentences that make up T.

3.1 Text-rank

The algorithm page-rank and its derivatives (Kleinberg, 1999) use a graph. Broadly speaking, at the beginning of page-rank a value is assigned to each node. Then, in an iterative fashion, it updates the values. After an $\epsilon-$convergency to its fixed point is reached, every node has a score; which means the importance degree of the node as a function of the role it played in the paths of the graph. These algorithms belong to the class of iterative algorithms that look for a fix point; similar to the Gauss-Seidel algorithm

to solve simultaneous equations.

The edges arrangement can be done in one of the following ways: a directed graph with forward edges (previous sentences pointing to posterior ones); a directed graph with backward edges (posterior sentences pointing to previous ones); or an undirected graph. Let $G = (V, E)$ be the graph that we have constructed, where V is the set of nodes, and $E \subset V \times V$ is the set of edges. For each $v_i \in V$, let $In(v_i)$ be the set of nodes pointing to v_i, and let $Out(v_i)$ be the set of nodes pointed by v_i (in the case of undirected graphs $In(v_i) = Out(v_i)$).

The weighting of the graph is done from a text: each sentence labels a node of the graph, the similarity between two sentences is the weight of the edge that links the corresponding nodes. The similarity between sentences is a measure computed in different ways; for example, by using the following formula:

$$sim(o_1, o_2) = \frac{inter(o_1, o_2)}{\log(|o_1|) + \log(|o_2|)} \qquad (1)$$

where o_1 and o_2 are the sentences under consideration, $inter(o_1, o_2)$ is the number of words belonging to both o_1 and o_2, and $|o_i|$ the number of words of o_i.

The text-rank method (TR) is convergent with margin of error ϵ. The score of each node is computed as follows:

$$TR(o_i) = (1-d) + d * \sum_{o_j \in In(o_i)} w_{ji} \frac{TR(o_j)}{\sum_{o_k \in Out(o_j)} w_{jk}}, \qquad (2)$$

where, w_{ij} is the weight of the vertex joining o_i and o_j ($sim(o_i, o_j)$), and d is a fix value between 0 and 1. After getting the initial scores, TR is iterated until a fix point is reached using ϵ; see (Mihalcea, 2004) for more details.

3.2 Extracting keywords

Two methods to get keywords from a text are presented. They get the sentences score by computing the similarity between the set of keywords of the text and the sentence (formula (1)). The next code may clear the previous statement:

Algorithm: Ordering of sentences;
input T : list of sentences;
$kywr$: list of words;
output T' : list of sentences; // ordered
begin
foreach $o_i \in T$ **do**
$s_i = sim(o_i, kywr)$
$T' = project_2(sort([(s_1, o_1), \ldots, (s_n, o_n)]))$
end

Now, we will see two methods which obtain an input of the algorithm, namely $kywr$.

3.2.1 Text-rank

In this case (Kw), an undirected and not weighed graph is constructed taking lexical units as nodes. To define the edges between nodes the co-occurrence criteria, of both terms in a window of N units (Mihalcea and Tarau, 2004) is used. We select the 10 first terms with highest score.

3.2.2 Transition rank

Another method used in this work takes terms of mid-frequency as the base to get an extract. It has been seen (Urbizagásteguí-Alvarado, 1999) that such terms have high semantical contents. We use the *transition point* (TP) method to get terms of mid-frequency. The TP is a frequency that divides the vocabulary of a text into words of high and low frequency. In this way, the terms with a frequency around the TP are candidates for important terms; therefore, to choose mid-frequencies, a threshold must be given. This method was used in (Bueno-Tecpanecatl, Pinto, and Jiménez-Salazar, 2005) to get extracts. Also TP has been used in text clustering (Jiménez-Salazar, Pinto, and Rosso, 2005). In the present work, we use the transition rank method (see (Pérez et al., 2006)) because it does not need to define a threshold around the TP in order to select terms. When the terms of mid-frequency have been found, they are used to compute the score of each sentence accounting the mid-frequency terms contained in the sentence. An analog procedure may followed taking the keywords provided by text-rank algorithm (Mihalcea, 2004).

Essentially the procedure (TPR) is to choose terms with a frequency in a rank from the lowest not repeated frequencies to the highest repeated frequencies. The terms with such frequencies presumably have high semantical contents, and they are taken as the keywords of the text.

3.3 Representation index

In (Marcu, 1999) a simple method to generate the extract of a text was proposed. The key idea of this method is the representativeness index of a sentence, which in turn, the index is determined in the following way: the importance degree of a sentence o_i is determined inversely to the similarity between the text T removing o_i and T; since if o_i is important, and removing it from T make less similar this text to T. Then, the sentences are ordered according to its index: $o_1, \ldots, o_n$, where $sim(T - [o_i], T) \leq sim(T - [o_{i+1}], T), 1 \leq i < n$. We made a little variant to this method using the sentence instead of text diminished by the sentence: $o_1, \ldots, o_n$, where $sim([o_i], T) \geq sim([o_{i+1}], T), 1 \leq i < n$.

This method (RI) directly computes the score of each sentence o_i applying the formula (1) to the sentence and the full text: $sim(o_i, T)$. RI uses the same code as above (Odering of sentences) replacing T instead *kywr* in the similarity function.

4 Experiments

A description of the used data, its preprocessing, and an evaluation of the results is now given.

4.1 Dataset

The experiments were made on 533 articles, about news in the English language, from the DUC 2002 collection[1] they have no format at all.

Each text was converted to lower-case, spaces were inserted to separate punctuation symbols. The texts were divided into sentences (taking the period as a separator), empty lines and stopwords were deleted.

4.2 Applied procedure

The methods described above were applied: TPR, transition rank; Kw, keywords using text-rank; RI, representation index; and TR, text-rank.

In the case of the text-rank algorithm, having the text already preprocessed, a graph was constructed applying the formula (2) with $d = 0.85$ (Mihalcea, 2004). The initial value assigned to each node was 1, and the convergency error was $\epsilon = 0.001$. It took an average of 18 iterations to reach the fixed point.

To produce the extract from each text the 7 sentences with the highest score were taken, independently of the method considered.

Some method combinations were made in order to know the possible relationship between them. The combination consisted of getting the score of each sentence, by computing the maximum between the score of two methods M_1, M_2: $\max(score(M_1), score(M_2))$.

[1] Document Understanding Conference, `http://duc.nist.gov/`.

Method	Score	Method	Score
TR	0.5761	max(TR,TPR)	0.5416
TPR	0.4711	max(TR,Kw)	0.5498
Kw	0.4969	max(Kw,TPR)	0.4813
RI	**0.6284**	max(TR,RI)	0.6148

Table 1: Evaluation of the methods and some combinations.

4.3 Evaluation

To evaluate the results, the automatic summaries evaluation package, ROUGE was used, it is based on statistics of N-grams. ROUGE was used with: ROUGE-L, confidence interval of 95%, without reserved words, score formula model average, assigning the same importance to precision and recall, and averaging the score of the units.

Table 1 shows the values gotten in evaluating the results by ROUGE. The representation index method had the highest value (0.6284).

5 Discussion

Three approaches to sentence extraction were applied to the collection DUC 2002: keyword-based (TPR, Kw), representation-based (RI) and, graph-based (TR). The best method was RI. Combining its results, through score maximization, the evaluation revealed they are not complementary; one of them can not help the other. Since they share score function and data from the text, the combination improved only one method: $E(M_1) < E(M) < E(M_2)$.

In Table 1 we can see higher scores are shown by methods which use the full sentences in order to determine the score. Those methods whose parameters were a reduced set of words, i.e. keyword-based, got the lowest evaluation. And how they calculate the keywords was not important because the difference between score values was very small.

This result is explained by the lose of information, since they only worked with isolated terms.

For high score, the differences among the methods are mainly given by the parameters used in the similarity function. RI method used as a parameter the whole text to calculate the score, while TR method extends the similarity between sentences to all sentences indirectly through iteration. In spite of using the whole text, RI could introduce noise in the computation of similarity, when it was used the highest performance was obtained. It seems that used information in graph-based method cannot be incorporated throughout iteration as it was done in the representation-based method.

The strength of TR is the iteration[2], which refine scores of sentences, whilst the strength for RI is the use of full text. These features may help to formulate a better algorithm considering a deeper representation of the text sentence, for instance using relative position of terms in the sentence; and a richer class of nodes in the graph-based method, as the application of TR to connected components instead of nodes. These issues as well as test of combination at high or low level, varying the dataset and evaluation system will be considered as future work.

References

Brill, Erick. 1994. Some advances in rule-based part of speech tagging. In AAAI, editor, *Proceedings of the AAAI Conference*.

Brin, Sergey and Lawrence Page. 1998. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30:1–7.

Bueno-Tecpanecatl, Claudia, D. Pinto, and Héctor Jiménez-Salazar. 2005. El párrafo virtual en la generación de extractos. *Research on Computing Science*, 13:85–90.

Hovy, Eduard. 2005. Text summarization. In R. Mitkov, editor, *The Oxford Handbook of Computational Linguistics*. Oxford University Press, 1st edition, pages 583–598.

[2]Actually TR outperformed (HITS, 0.5023) the top systems of DUC 2002 (0.5011) (Mihalcea and Tarau, 2004).

Jiménez-Salazar, Héctor, David Pinto, and Paolo Rosso. 2005. Uso del punto de transición en la selección de términos índice para agrupamiento de textos cortos. *Procesamiento del Lenguaje Natural*, 35:383–390.

Kleinberg, J.M. 1999. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632.

Marcu, Daniel. 1999. The automatic construction of large-scale corpora for summarization research. In *Proceedings of the SIGIR of ACM 99*, pages 137–144.

Mihalcea, Rada. 2004. Graph-based ranking algorithms for sentence extraction, applied to text summarization. In *The Companion Volume to the Proc. of 42st Annual Meeting of the ACL*, pages 170–173, Barcelona, Spain, July. Association for Computational Linguistics.

Mihalcea, Rada and Paul Tarau. 2004. Textrank: bringing order into text. In *The Companion Volume to the Proc. of 42st Annual Meeting of the ACL*, pages 190–193, Barcelona, Spain, July. Association for Computational Linguistics.

Mihalcea, Rada, Paul Tarau, and Elizabeth Figa. 2004. PageRank on Semantic Networks, with application to Word Sense Disambiguation. In *Proc. of the 20st International Conference on Computational Linguistics*.

Montejo, Arturo Ráez, Alfonso Urena, and Ralf Steinberg. 2005. Text categorization using bibliographic records: beyond document content. *Procesamiento del Lenguaje Natural*, 35:119–126.

Ng, Hwee Tou and Hian Beng Lee. 1996. Integrating Multiple Knowledge Sources to Disambiguate Word Sense: An Exemplar Based Approach. In *Proc. the 34th Annual Meeting of the ACL*.

Pérez, David, José Tepacuacho, Héctor Jiménez, and Grigori Sidorov. 2006. A term frequency range for text representation. *Research on Computing Science*, 20:113–118.

Urbizagástegui-Alvarado, Rubén. 1999. Las posibilidades de la ley de Zipf en la indización automática. Technical report, Universidad de California Riverside, California, USA.