Improving Text Segmentation with Clustering Cohesion

Raúl Abella Pérez and José Eladio Medina Pagola

Advanced Technologies Application Centre
Ciudad de la Habana, Cuba
1 Introduction

Text segmentation is the task of splitting a document into syntactical units (paragraphs, sentences, words, etc.) or semantic blocks, usually based on topics. The difficulty of text segmentation mainly depends on the characteristics of documents to be segmented (i.e. scientific texts, news, etc.) and the segmentation outputs (e.g. topics, paragraphs, sentences, etc.).

This work is concentrated on topic text segmentation. Topic segmentation tries to identify the boundaries in a document for capturing the latent topical structure. The automatic detection of appropriate subtopic boundaries in a document is a very useful task in text processing. For example, in information retrieval and in passages retrieval, to return documents, segments or passages closer to the user’s queries. Another application of topic segmentation is in summarization, where it can be used to select segments of texts containing the main ideas for the summary requested (Abella-Pérez & Medina-Pagola, 2010b; Hearst, 1997; Hernández & Pagola, 2007; Ken & Granitzer, 2009).

In the literature there are many methods of texts segmentation by topic. There are different approaches to solve this problem; one is a linear segmentation, where the document is split into a linear sequence of adjacent segments (Abella-Pérez & Medina-Pagola, 2010b; Hernández & Pagola, 2007; Ken & Granitzer, 2009). Another approach is a hierarchical segmentation; the outputs of these algorithms try to identify the document structure, usually chapters and multiple levels of sub-chapters.

When analyzing the performance of different methods of text segmentation by topic we observed some difficulties as, for instance, wrong interruptions of segments, omission of sentences or paragraphs which belong to other segments, and the generation of segments with incomplete information (Hearst, 1997; Hernández & Pagola, 2007; Ken & Granitzer, 2009). Another difficulty we observed is that those methods are not able to identify the true relations amongst paragraphs of each segment and between segments considering natural topic cohesion. Many of these methods are dependent on different parameters, such as the threshold to decide whether two textual units are cohesive. The dependence of an algorithm on threshold definition makes it hard to use, and knowing the threshold for deciding whether two paragraphs are related is difficult and varies depending on document characteristics.

At present there are different methods of linear segmentation by topic; most of them use lexical cohesion to identify the boundaries between textual units. One of the most complete works was proposed by Hearst in 1994, which uses a sliding window to scan the document and to determine topic boundaries (Hearst, 1997). Heinonen's proposal in 1998 is another example of text segmentation by topic (Heinonen, 1998). The author uses a dynamic programming algorithm to identify the boundaries between the topics. In 2010, Abella and Medina proposed a linear segmentation algorithm by topic (ClustSeg) (Abella-Pérez & Medina-Pagola, 2010b). In this proposal, the identification of the topic boundaries is based on the results of a clustering algorithm using a sliding window approach.

ClustSeg, when tries to identify topic boundaries, requires two parameters defined by the user: window size and threshold. The window is used to process each group resulting from the application of a clustering algorithm in order to obtain all the segments which can be formed from each cluster. The second parameter, the threshold, is one of the parameters used by the clustering algorithm to decide which paragraphs should belong to the same cluster.

In this work we propose a strategy to automatically calculate the threshold for deciding the cohesiveness between textual units. Besides, we evaluate the impact of different thresholds in the ClustSeg method, comparing them with our proposal.
We have structured the present work as follows. In Section 2 we briefly explain different works to solve the segmentation problem and their drawbacks. In Section 3 we describe the proposals that were evaluated to automatically calculate the threshold. In the last section we present the experimental results in different corpora.

## 2 Related Work

About the problem of topic segmentation, there are different aspects to be considered, both in the linguistic and computational field; for example, the distinction between topics and subtopics. Linguistic signals are used to computationally identify topic changes, and are also necessary to consider inherent computational preprocessing of the original text.

Many segmentation methods by topic use linguistic signs to identify the changes of topics between textual units (Abella-Pérez & Medina-Pagola, 2010b; Choi, 2000; Hearst, 1997; Heinonen, 1998; Hernández & Pagola, 2007; Ken & Granitzer, 2009). In order to select an adequate sign, it is very important to take into account the text types which are going to be segmented. Lexical cohesion is one of the linguistic signs more used in segmentation methods.

The term Lexical Cohesion was defined by Halliday and Hasan in 1976 as a sense relationship that exists among the textual units in a text. Text segmentation by topic is one of the signals more used to determine the boundaries between textual units, assuming that the textual units that are closely related by lexical cohesion are usually in the same topic. According to Halliday and Hasan, lexical cohesion includes two different aspects, the repetition and placement. In the task of text segmentation the most used is reiteration.

Below we will describe some segmentation methods that are focused on the identification of sub-topic structures in documents; they are based on lexical reiteration to detect relationship among textual units.

### 2.1 TextTiling

Hearst’s work is an example that uses lexical cohesion as a mechanism to identify the topic boundaries. Hearst proposed a method which tries to split texts into discourse units of multiple paragraphs, called TextTiling (Hearst, 1997). This algorithm uses a sliding window approach and, for each position, two blocks are built; one preceding and the second succeeding each position. To determine the lexical punctuation between these two blocks, it uses the term repetition as a lexical cohesion mechanism. These blocks are formed by a specified amount of pseudo-sentences which are represented by the vector space model and the cosine as the similarity measure. Considering the lexical values calculated, this method splits the text from the valleys, or points with low lexical scores. Where a low lexical score is preceded and succeeded by a high score is taken as an indicator of lexical change of terminology and therefore coincides with a change of topic. This method uses another window, usually smaller, to decide the minimum size of a valid segment.

TextTiling algorithm is simple to implement and does not require data training to identify topics boundaries. Also, does not use dictionaries or thesauri. This algorithm maintains a good performance, but it presents a drawback that causes the interruption of a segment that contains a simple subtopic; this problem also produces many segments that surpass the considered valid amount. This occurs when there is a
short paragraph or paraphrase which interrupts a cohesive text chain. TextTiling does not detect this behavior. In these anomalous cases, TextTiling gets a notable low score and, then, assigns a segment boundary.

### 2.2 Heinone’s Method

Unlike Hearst’s, Heinone proposed a method which uses a sliding window to determine, for each paragraph, which is the most similar paragraph inside the window (Heinonen, 1998). The sliding window is formed by several paragraphs on both sides (above and below) of every processed paragraph. In this proposal, the ideal segment size is identified using a cost function. This function uses a value (and not a window) \( p \) (the author experiments with 600 words) to decide the size of a segment, penalizing the lesser and greater than this value. This segmentation method is especially useful when it is important to control the segment length. The author uses a dynamic programming technique which guarantees getting segments of minimum cost. In this method, each paragraphs are represented using the vector space model and similarities between them are calculated using the cosine measure.

Next, a cohesion vector is built for the document, where each paragraph is associated with the highest similarity value inside its window. The window is composed of several surrounding paragraphs, paragraphs on top and paragraphs underneath. The paragraph vectors are weighted based on their distance from the boundary in question with immediate paragraphs having the highest weight. In order to determine the limits of topics, Heinonen used a programming method.

This method achieves good results among the segment length, the preference length and the similarity value associated with each paragraph. Also, it did not use thesauri or dictionaries, so it is independent of the text language. A drawback that presents this method is that it requires an approximate subtopic length, which is an unpredictable value that is not always the same for all the subtopics. Besides, this algorithm, to decide the inclusion of a paragraph in a segment, takes into account the higher value of similarity in its window, but this value can belong to a paragraph that is on top or underneath. This is not detected by the algorithm and it can decide incorrectly the inclusion of a paragraph in a segment.

### 2.3 C99

The C99 algorithm proposed by Choi is another example that uses lexical cohesion as a mechanism to identify the topic boundaries (Choi, 2000). This method determines the frequency of each word in the sentence and calculates the similarity between each pair of sentences using the cosine measure. With these values, it builds a matrix of similarities between sentences. More recently, Choi improved C99 by using the Latent Semantic Analysis (LSA) achievements to reduce the size of the word vector space (Choi, 2001). Then it builds a second matrix, called matrix of rank. This matrix is obtained by replacing each value of the similarity matrix by rank in the local region, using an 11x11 rank mask. The author calls range the number of neighbors who have a lower value of similarity in the local region.

Finally, C99 determines the topics boundaries using a divisive clustering process. It uses linear text segmentation by topic based on lexical cohesion as a strategy to identify boundaries between the segments. Also, it does not use thesauri either dictionaries, so it is independent of the text language. In this proposal, Choi introduces a special mechanism to determine the boundaries of topics by using the matrix rank. However, it has some difficulties. First, Choi represents each sentence as a vector, using the vector space model; this causes unnecessary calculations, making the size of matrices (similarity matrix and rank matrix) quite large. On the other hand, the probability that a given term appears more than once
in a sentence is almost zero, so that the weight of each term is nearly boolean. In addition, to build the matrix range, Choi uses a fixed rank of size 11x11; this value needs not to be the same for each text.

2.4 TextLec

Another approach of linear text segmentation by topic is TextLec, proposed in 2007 (Hernández & Pagola, 2007). This method uses word repetition as a lexical cohesion mechanism. Each paragraph is represented by the vector space model. The authors of this work assume that all the sentences which belong to a paragraph are about a same topic. This method also uses a sliding window approach but, unlike Hearst’s, it only uses a window of paragraphs which are below each position.

TextLec, after preprocessing, has two main stages: In the first stage, it identifies the farthest cohesive paragraph inside the window for each paragraph and finally identifies the boundary segments.

In order to decide if a paragraph has cohesion inside the window, they calculate the similarity using the cosine measure and, then, they take the farthest cohesive paragraph. The threshold is used to determine whether two paragraphs are similar. TextLec finally forms the topic segments. This process involves the inclusion of paragraphs located between the beginning of the segment and the farthest cohesive paragraph.

This algorithm is able to eliminate some deficiencies presented in Hearst and Heinone’s work. For example, by using an inferior window for each paragraph, formed only with paragraphs underneath the paragraph in question, it is able to decrease the effect of short paragraphs or others that interrupt cohesiveness. Besides, the use of a window allows reducing unnecessary calculations.

2.5 TSF

TSF is a linear topic segmentation algorithm proposed by Roman Kern and Michael Granitz in 2009 (Ken & Granitzer, 2009). It assumes that lexical cohesion is an indicator of topic change in a document and it uses an approach of sliding window. Each sentence is represented using the vector space model. To calculate the similarity between two sentences, it uses the cosine similarity.

Similar to TextTiling, TSF constructs for each position \( i \) two blocks, formed by a set of sentences that precede and succeed the current position \( \mu(B_i^{pre}, B_i^{post}) \). The block size is one of the parameters defined by the user, which reflects the minimum size for a segment to be considered valid.

For each of the two blocks it calculates the similarity between each sentence pair. Then it calculates the arithmetic mean of the similarity values to determine the internal similarity; this value is interpreted as the level of similarity of the sentences around the current position. The parameter \( \mu \) denotes the average of similarities between two blocks.

\[
\text{sim}^{inner}_i = \frac{\mu(B_i^{pre}, B_i^{pre}) + \mu(B_i^{post}, B_i^{post})}{2}. \tag{1}
\]

Then, it is calculated the similarity of the sentences of a block with regard to the sentences of another block, these similarities are averaged to obtain a global value, expressing how similar are two blocks.

\[
\text{sim}^{outer}_i = \mu(B_i^{pre}, B_i^{post}). \tag{2}
\]

With these two similarity values, the dissimilarity between the two blocks around the current position is calculated as:
disimilarity_i = \frac{\text{sim}_i^{\text{inner}} - \text{sim}_i^{\text{outer}}}{\text{sim}_i^{\text{inner}}}. \quad (3)

This value is positive if the average similarity between the sentences of both blocks is less than the average similarities within blocks, which means that the current position is a real limit topic. If the maximum is reached outside (1) this means that the external similarity is zero which means that the blocks do not have terms in common. If a dissimilarity value exceeds the threshold value (second parameter input by the user), the corresponding position is marked as restricted candidate and this candidate is chosen as the limit if there is no higher value of dissimilarity of the following positions of sentences.

TSF considers linear topic segmentation; it does not use either dictionaries or thesauri and does not need training data. This method, for identifying the boundaries of topics, takes two parameters. The first is the size of the block, which defines the minimum size considered for a valid segment. The other parameter that this method requires is a threshold, which is really difficult to know in order to determine if two text units are related.

2.6 ClustSeg

ClustSeg, proposed by Abella and Medina in 2010, is another proposal of linear topic segmentation, which aims to identify topics boundaries in text, using the term repetition as lexical cohesion mechanism (Abella-Pérez & Medina-Pagola, 2010b). In this work it is assumed that all the sentences belonging to a same paragraph are about a same topic. But, unlike what is proposed in TextLec, it is considered that a paragraph can be related to more than one topical segment. Each paragraph is represented by the vector space model and cohesion among them is calculated using the cosine measure. Boundary identification of topics is defined on the results of applying a clustering algorithm using a sliding window strategy.

In this proposal it is assumed that changes in the vocabulary match with the subtopic changes in a document. Thus, the paragraphs about the same topics have a significant lexical cohesion among themselves. But, unlike the proposals of TextLec and TextTiling, the distance between them does not have any importance at a first stage. It is due to the fact that a paragraph would refer to a previous topic of the document, since paragraphs can be related to more than one topic. This might occur when the author refers to a segment of topic which has already been closed. Based on these assumptions, it is uses an overlap clustering algorithm to identify topic boundaries.

The ClustSeg algorithm, after preprocessing the text and eliminating the stopwords (prepositions, conjunctions, articles, and pronouns), has three main stages: In the first stage, it searches for topic cohesion using an overlapping clustering algorithm. The algorithm used by the authors is ICSD (Pérez-Suárez, Martinez-Trinidad, & Carrasco-Ochoa, 2009). This algorithm has, as input, the vector of each paragraph and a threshold (defined by the user), returning a cluster set, where each cluster represents a set of cohesive paragraphs belonging (presumably) to an independent topic.

After obtaining the clustering from the static version of ICSD algorithm, ClustSeg processes all clusters in order to obtain all the segments which can be formed from each cluster. As in other methods, it uses a window to define if two adjacent paragraphs in a cluster are close. Each segment is obtained by joining adjacent paragraphs according to the user predefined window, which also defines the minimum size for a segment to be selected as valid.

As a result of the previous stage, a set of segment limits of topics is obtained. As these segments were obtained from different groups, and are overlapped, these segments could also be overlapped. For
this reason, in a later stage, ClustSeg concatenates all segments that have at least one paragraph in common. At the end of this stage, it identifies the cohesive segments. This method considers as valid limit the paragraph previous to the start of every segment.

ClustSeg assumes that a paragraph can be related to various topics, and although it does not identify relationship existing between topics, it is a first step to determine it.

On the other hand, it does not require training data to identify the boundaries of topics. It is able to reduce incorrect ending of segments. This method has several shortcomings such as the fact that it depends on several parameters to identify topic segments (threshold, window).

3  Improving ClustSeg

As mentioned above, ClustSeg requires that the user defines a threshold. This threshold is used to define whether two paragraphs are cohesive with each other. The dependence of an algorithm on threshold definition makes it hard to use, and knowing the threshold for deciding whether two paragraphs are related is difficult and varies depending on document characteristics. For these reasons, it is not recommended to use a predefined threshold as a global parameter for every document.

To calculate the threshold, we considered several measures of central tendency to decide which one is the optimal minimum-variance unbiased estimator. In our proposals, we analyzed average and median like measures. We analyzed other measures, like simple mid-range, mid-hinge, trimean and others, but they showed poorer behaviors.

In this work we calculate the threshold automatically using the same window defined by the user. First, for each window it is calculated a set of similarities between every pair of paragraphs, using the:

- **Set 1**: The average similarity amongst all the values in the window similarity set.
- **Set 2**: The greatest similarity amongst all the values in the window similarity set.
- **Set 3**: The lowest similarity amongst all the values in the window similarity set.

In order to find the best threshold we evaluate four proposals in different texts, using the sets of similarity described above. The proposals considered are:

- **Proposal 1**: Average of similarity values of the Set 1 \( (P) \).
- **Proposal 2**: Median of similarity values of the Set 1 \( (MP) \);
- **Proposal 3**: Average of \( MP, MG \) and \( ML \).
  
  \( MG \): median of similarity values of the Set 2.
  
  \( ML \): median of similarity values of the Set 3.
- **Proposal 4**: Average of \( P, MaxP \) and \( MinP \).
  
  \( MaxP \): highest value of set 1.
  
  \( MinP \): minimum value of set 1

Next, we describe the steps taken to select the best proposal for threshold calculation:
For each test text, we made several runs of the method (varying the window size) considering each proposal.

For each segmentation obtained, the evaluation measure (WindowDiff) values are calculated.

We analyze the evaluation measure values obtained by each proposal.

We included the average as Proposal 1 because it is the most commonly used measure. Also, as Proposal 4, we considered an average including a mid-range approach. Nevertheless, we knew that average measures are very sensitive to outliers. Knowing this, we included median measures: the simple median, as Proposal 2, and the average of the simple median and the maximum-minimum medians, as Proposal 3.

In the experiments, we observed that averaging median and minimum-maximum medians (Proposal 3) gets the best results. Perhaps, it catches better the variability of a local similarity set in a relative small sample and probably uniform distributed on each document. For this reason, we consider as valid threshold the obtained by the Proposal 3, Proposal 3 was the one chosen. These experiments are shown in the following section.

4 Evaluation

There are two main problems concerning the evaluation of text segmentation algorithms. The first one is given by the subjective nature when detecting the right boundaries of topics and sub-topics into texts; it turns the selection of reference segmentation from a fair and objective comparison into a very difficult task (Pevzner & Hearst, 2000). In order to solve this problem, usually artificial documents are created, concatenating different real documents, on the assumption that the limits between these documents are good breaking points (Abella-Pérez & Medina-Pagola, 2010b; Hernández & Pagola, 2007). Another way is to compare the results against a manual segmentation based on human judgments, which makes a “gold standard” (Stokes, Carthy, & Smeaton, 2004).

The second problem is the selection of a measure to be used in the evaluation; because, for different applications of text segmentation, different types of mistakes become more or less important. For example, in Information Retrieval it can be accepted that the segment boundaries differ in a few sentences from the real segment boundaries. However, in order to segment news stories from broadcast news, the accuracy of boundaries location is very important.

In this section, we describe the different evaluation measures that exist, and the corpus used to evaluate the efficacy of Clustseg, with the proposal for calculating the threshold. Finally, we can see the experimental results.

4.1 Evaluation measures

Two of the evaluation measures that have been used by many authors are Precision and Recall, which are standard measures in Information Retrieval experiments. In the estimation of the segmentation accuracy, the Precision and the Recall are defined like this.

- **Precision**: The percentage that represents the segment boundaries correctly detected by the algorithm from all boundaries detected by it.
Recall: The percentage that represents the segment boundaries correctly detected by the algorithm from all boundaries in the reference segmentation.

Precision and Recall are usually very convenient in applications where the accuracy of boundaries location is very important. But in applications where it is not very necessary they have some problems. These measures strongly penalize the algorithm when boundaries that do not agree exactly with the reference segmentation are detected, because they are not sensitive to the proximity between the boundaries of both segmentations. Another difficulty with Precision and Recall is that there is inherent tradeoff between precision and recall; improving one tends to cause the score for the other to decline (Pevzner & Hearst, 2000). This difficulty is usually solved in Information Retrieval with F-measure; it is defined as:

\[
F - \text{measure} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}
\] 

F-measure has been used in segmentation as well. Nevertheless, we should note that as F-measure depends on Precision and Recall it shows the first problem, i.e., it is not sensitive to the proximity between the boundaries of both segmentations.

\(P_k\) is another measure used to evaluate segmentation methods proposed in 1997 and 1999 by Beeferman, Berger and Lafferty. This measure is the probability that two sentences taken randomly from the text are correctly classified as belonging or not to the same segment. This measure, as opposed to Precision and Recall, considers the proximity of the limits of the reference segmentation and the segmentation obtained. Despite this, the measure presents some shortcomings. One of these is that the algorithm penalizes stronger when it ignores a segment limit when you put it incorrectly.

WindowDiff is a measure proposed by Pevzner and Hearst in 2000 (Pevzner & Hearst, 2000). This measure uses a sliding window of length \(k\) to process the whole document and find the discrepancies between the reference segmentation and the segmentation obtained by the algorithm. As proposed by Beeferman, Berger and Lafferty, the authors maintain \(k\) equal to half the average size of the segments with reference segmentation.

The amount of boundaries inside the window of both segmentations is determined for each window position; it is penalized if the amount of boundaries disagrees. Later, all penalizations found are added. This value is normalized and the metric takes a value between 0 and 1. WindowDiff takes a score of 0 if all boundaries are correctly assigned and it takes a score of 1 if there is a complete difference between the automatic segmentation and the reference one. Let \(s_{ref}\) and \(s_{alg}\) be the reference and algorithm segmentations, WindowDiff is defined as:

\[
\text{WindowDiff}(s_{ref}, s_{alg}) = \frac{1}{N - K} \sum_{i=1}^{N-k} \left( |b(s_{ref}, s_{ref}^{j+k}) - b(s_{alg}, s_{alg}^{j+k})| > 0 \right)
\]

where \(b(i, j)\) represents the number of boundaries between the position \(i\) and \(j\) in the text and \(N\) is the total number of text units in the document.
4.2 Corpus

In order to check the performance of ClustSeg (with each proposal) we made several experiments with different texts sizes. In Table 1 the different corpora used in the experiments are shown.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Number of paragraphs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text 1</td>
<td>The same corpus used by the ClustSeg authors (Abella-Pérez &amp; Medina-Pagola, 2010b).</td>
<td>305</td>
</tr>
<tr>
<td>Text 2</td>
<td>Joint of two different topics papers: “Background” of (Jia, Huan, Buhr, Zhang, &amp; Carayannopoulos, 2009) and “An Introduction to Latent Semantic Analysis” of (Landauer, Foltz, &amp; Laham, 1998).</td>
<td>22</td>
</tr>
<tr>
<td>Text 3</td>
<td>This corpus was formed with the work published in (Abella-Pérez &amp; Medina-Pagola, 2010a). Regarded as valid the boundaries between sections. Abstract, conclusions and references were discarded</td>
<td>31</td>
</tr>
<tr>
<td>Text 4</td>
<td>TDT2 corpus file (19980104_1337_1411_NYT_NYT.sgm).</td>
<td>317</td>
</tr>
<tr>
<td>Text 5</td>
<td>TDT2 corpus file (19980601_1201_1217_APW_ENG.sgm).</td>
<td>58</td>
</tr>
<tr>
<td>Text 6</td>
<td>Concatenation of different AFP news (AF941122_0126.noti, AF940804_0203.noti, AF940622_0001.noti, AF941121_0103.noti, AF941225_0037.noti, AF941018_0167.noti, AF940524_0311.noti, AF940729_0100.noti, AF941228_0261.noti)</td>
<td>109</td>
</tr>
<tr>
<td>Text 7</td>
<td>Concatenation of different AFP news (AF940830_0110.noti, AF941230_0016.noti, AF940911_0024.noti, AF940622_0060.noti)</td>
<td>52</td>
</tr>
</tbody>
</table>

Table 1: Corpora used in the experimentation.

4.3 Selection of the threshold

The measure we use to evaluate the performance of the method is WindowDiff since, in our proposal, the accuracy of the boundaries is not important, because we are trying to automatically discover topic segmentations.

In Figure 1 we can see the results obtained by ClustSeg with each proposal and different intervals of window.

Finally, in Figure 2 we can see a graph with the best values of WindowDiff obtained by the method with each proposal in different texts.

Analyzing the performance of ClustSeg through the behavior of WindowDiff values, which are shown in the previous figures, we can see that in general the method gets better performance with Proposal 3, where in all texts, except in Text 5, this proposal outperforms the rest.

This experimentation, posing no conclusive demonstration, suggests that, according to the assumptions, the value more representative of the lexical cohesion between paragraphs is the one calculated by the Proposal 3, which is an averaging of the simple median and the minimum-maximum medians of the similarities. This proposal is considered in the next experiments to calculate the threshold automatically. Hereinafter we refer to the method with the improved of calculation of the threshold as ClustSegU.
<table>
<thead>
<tr>
<th>Proposal 1</th>
<th>Proposal 2</th>
<th>Proposal 3</th>
<th>Proposal 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.44 0.44</td>
<td>0.40 0.44</td>
<td>0.13 0.13</td>
<td>0.16 0.13</td>
</tr>
<tr>
<td>0.44 0.44</td>
<td>0.44 0.44</td>
<td>0.12 0.11</td>
<td>0.16 0.13</td>
</tr>
<tr>
<td>0.16 0.16</td>
<td>0.16 0.16</td>
<td>0.16 0.16</td>
<td>0.16 0.16</td>
</tr>
<tr>
<td>0.16 0.16</td>
<td>0.16 0.16</td>
<td>0.16 0.16</td>
<td>0.16 0.16</td>
</tr>
<tr>
<td>0.21 0.21</td>
<td>0.21 0.21</td>
<td>0.21 0.21</td>
<td>0.21 0.21</td>
</tr>
<tr>
<td>0.21 0.21</td>
<td>0.21 0.21</td>
<td>0.21 0.21</td>
<td>0.21 0.21</td>
</tr>
<tr>
<td>0.32 0.32</td>
<td>0.32 0.32</td>
<td>0.32 0.32</td>
<td>0.32 0.32</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Proposal 1</th>
<th>Proposal 2</th>
<th>Proposal 3</th>
<th>Proposal 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.36 0.39</td>
<td>0.39 0.43</td>
<td>0.40 0.40</td>
<td>0.31 0.33</td>
</tr>
<tr>
<td>0.39 0.39</td>
<td>0.43 0.43</td>
<td>0.40 0.44</td>
<td>0.22 0.26</td>
</tr>
<tr>
<td>0.26 0.30</td>
<td>0.25 0.30</td>
<td>0.26 0.26</td>
<td>0.22 0.26</td>
</tr>
<tr>
<td>0.18 0.18</td>
<td>0.18 0.18</td>
<td>0.18 0.18</td>
<td>0.18 0.18</td>
</tr>
<tr>
<td>0.38 0.29</td>
<td>0.29 0.29</td>
<td>0.32 0.32</td>
<td>0.32 0.32</td>
</tr>
<tr>
<td>0.28 0.28</td>
<td>0.26 0.26</td>
<td>0.30 0.30</td>
<td>0.24 0.24</td>
</tr>
<tr>
<td>0.24 0.24</td>
<td>0.30 0.30</td>
<td>0.26 0.26</td>
<td>0.30 0.30</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Proposal 1</th>
<th>Proposal 2</th>
<th>Proposal 3</th>
<th>Proposal 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.19 0.19</td>
<td>0.21 0.21</td>
<td>0.23 0.23</td>
<td>0.28 0.28</td>
</tr>
<tr>
<td>0.21 0.21</td>
<td>0.32 0.32</td>
<td>0.19 0.19</td>
<td>0.23 0.23</td>
</tr>
<tr>
<td>0.19 0.19</td>
<td>0.19 0.19</td>
<td>0.23 0.23</td>
<td>0.23 0.23</td>
</tr>
</tbody>
</table>

**Figure 1:** WindowDiff values obtained by ClustSeg with each proposal in different texts and different values of window.

**Figure 2:** WindowDiff values obtained by ClustSeg with each proposal in different texts.

### 4.4 Comparison with other algorithms

This section provides a comparison of the performance of ClustSegU with other methods that attempt to solve problems similar to those presented in this paper. As mentioned above, the performance of the
methods is evaluated by the *WindowDiff* metric in different texts. For a fair comparison, we evaluate ClustSegU with other methods using the parameters which show the best performances.

The results of the first experiment can be seen in Table 2. This experiment was done in the same corpus used by the ClustSeg authors, Text 1, a text which is relatively large. The result obtained in this corpus by ClustSegU was with a window equal to 13 and 14. In this table we see that ClustSegU improves the results achieved by other methods, without having to define the threshold.

In this experimentation, we did not include the TSF method because we do not have its code. We include some results of TSF method, particularly in Table 4, thanks to the authors of this work, who sent us several runs of the method using Text 2 and Text 3. As we do not have TextTiling and Heinone’s codes, they could not be included in the experimentation, except with Text1 (Table 2) taking the results appearing in the references with the same corpus. The C99 algorithm used was downloaded from the following URL: http://sourceforge.net/projects/textsegfault/files/c99/C99-1.2-release.tgz/download. While the TextLec method utilized is the same used by the authors.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>ClustSegU</th>
<th>TextLec</th>
<th>TextTiling</th>
<th>Heinone’s</th>
<th>C99</th>
</tr>
</thead>
<tbody>
<tr>
<td>WindowDiff</td>
<td>0.11</td>
<td>0.21</td>
<td>0.33</td>
<td>0.26</td>
<td>0.21</td>
</tr>
</tbody>
</table>

**Table 2:** WindowDiff values in Text 1.

We also made other evaluations in this corpus, varying the window size in order to see the behavior of the efficacy of ClustSegU. In Table 3, we see these results, where ClustSegU with window sizes from 7 to 14 improves the results achieved by the other methods.

<table>
<thead>
<tr>
<th>Window</th>
<th>ClustSegU</th>
</tr>
</thead>
<tbody>
<tr>
<td>W=15</td>
<td>0.23</td>
</tr>
<tr>
<td>W=14</td>
<td>0.11</td>
</tr>
<tr>
<td>W=13</td>
<td>0.11</td>
</tr>
<tr>
<td>W=12</td>
<td>0.12</td>
</tr>
<tr>
<td>W=11</td>
<td>0.13</td>
</tr>
<tr>
<td>W=10</td>
<td>0.13</td>
</tr>
<tr>
<td>W=9</td>
<td>0.14</td>
</tr>
<tr>
<td>W=8</td>
<td>0.14</td>
</tr>
<tr>
<td>W=7</td>
<td>0.19</td>
</tr>
</tbody>
</table>

**Table 3:** WindowDiff values of ClustSegU in the Text 1 by varying the window size.

As mentioned above, Text 1 is relatively large (has 305 paragraphs). For this reason, we accomplished other experiments using smaller texts (Text 2 and Text 3) to see if ClustSegU improves the results obtained by other methods. On the other hand, we included the best results of TSF algorithm in these corpora. In Table 4 we can see these results.
Table 4: WindowDiff values in Text 2 and Text 3.

The results of ClustSegU in Text 2 were obtained with a window size of 3 and 4, which correctly identified the topic boundaries. On the other hand, the result obtained in Text 3 was with window sizes from 5 to 8 and the threshold value calculated automatically was 0.34. Meanwhile, the best results of TSF, in Text 2 and Text 3, were obtained with a window size of 8 and a threshold of 0.7.

Moreover, we evaluated the effectiveness of ClustSegU considering a wide range of threshold values, to see the thresholds with better results and the obtained by the automatic calculation. In Table 5 we can see these results using Text 3.

Table 5: WindowDiff values of ClustSegU in the Text 3 varying the threshold and window size (W).

In Table 5 we can see that ClustSegU obtains the best results with a threshold of 0.34, where, in the worst case, the method with a window size 11, 10, 9 and 4 produces the same result as the method C99 and TSF while with values of window 8, 7, 6, 5 and 3 obtains better results than the other methods. These results show that the proposal made for the automatic generation achieves better or similar results than any other possible strategy could be defined. So, our proposal seems to be a good (in this case the best) approach.
The previous experiments were accomplished in personal corpora. Next we will carry out several experiments using international corpora (TDT and AFP) used to evaluate different tasks in natural language processing.

The TDT2 collection was downloaded from the site http://www.nist.gov/speech/tests/tdt.html; it contains English news from different news agencies published in 1998 during the period January to June. This corpus consists of several files where each file contains a set of news. In order to experiment in this corpus we selected files at random and took the news limits as valid topic boundaries.

The AFP collection can be downloaded from http://trec.nist.gov and it contains news in Spanish that were published by the AFP news agency during 1994. The news stories from each collection were chosen and concatenated to form two different corpora.

<table>
<thead>
<tr>
<th>Corpus</th>
<th>Algorithm</th>
<th>ClustSegU</th>
<th>TextLec</th>
<th>C99</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text 4</td>
<td></td>
<td>0.17</td>
<td>0.27</td>
<td>0.16</td>
</tr>
<tr>
<td>Text 5</td>
<td></td>
<td>0.22</td>
<td>0.22</td>
<td>0.30</td>
</tr>
<tr>
<td>Text 6</td>
<td></td>
<td>0.18</td>
<td>0.25</td>
<td>0.49</td>
</tr>
<tr>
<td>Text 7</td>
<td></td>
<td>0.19</td>
<td>0.11</td>
<td>0.32</td>
</tr>
</tbody>
</table>

**Table 6:** WindowDiff values in TDT2 and AFP corpora.

In Table 6 we can see, that in these corpora, ClustSegU achieves a better performance than the other methods, except in the Text 7.

## 5 Conclusion

Text segmentation is an important task for many preprocessing tools of natural language. At present, although there are several methods of segmentation, the effectiveness of these methods is not entirely acceptable. Moreover, many of these methods are dependent on different parameters, such as the threshold to decide whether two textual units are cohesive. The dependence of an algorithm on threshold definition makes it hard to use, and to knowing the threshold for deciding whether two paragraphs are related is difficult and varies depending on document characteristics.

In this work we propose a strategy to automatically calculate the threshold for deciding the cohesiveness between textual units. Besides, we evaluate the impact of different thresholds in the ClustSeg method, comparing them with our proposal.

As future work, in order to improve the results of ClustSeg we will evaluate other clustering methods and will analyze a better integration of both strategies. We will consider other ways of segmentation from overlapped segments. Besides, we will evaluate different strategies to identify the minimum size of valid segment.

**References**


