Journal of Universal Computer Science, vol. 14, no. 2 (2008), 178-192
submitted: 15/6/06, accepted: 17/11/07, appeared: 28/1/08

Using an Evolving Thematic Clustering in a Text Segmentation Process

Sylvain Lamprier
(LERIA - University of Angers, France
lamprier@info.univ-angers.fr)

Tassadit Amghar
(LERIA - University of Angers, France
amghar@info.univ-angers.fr)

Bernard Levrat
(LERIA - University of Angers, France
levrat@info.univ-angers.fr)

Frederic Saubion
(LERIA - University of Angers, France
saubion@info.univ-angers.fr)

Abstract: The thematic text segmentation task consists in identifying the most important thematic breaks in a document in order to cut it into homogeneous passages. We propose in this paper an algorithm for linear text segmentation on general corpuses. It relies on an initial clustering of the sentences of the text. This preliminary partitioning provides a global view on the sentences relations existing in the text, considering the similarities in a group rather than individually. The method, so-called ClassStruggle, is based on the distribution of the occurrences of the members of each class. During the process, the clusters then evolve, by considering a notion of proximity and of layout in the text, in the aim to create groups that contain only sentences related to a same topic development. Finally, boundaries are created between sentences belonging to two different classes. First experimental results are promising, ClassStruggle appears to be very competitive compared with existing methods.
Key Words: Text Segmentation, Clustering
Category: I.7, I.2.7

1 Introduction

The purpose of automatic text segmentation is to identify the most important thematic breaks in a document in order to cut it into homogeneous parts. In [Salton et al. 1996], these parts, called document units, are defined as parts of text with strong intrinsic relationships, disconnected from other adjacent parts. More precisely, the segmentation task consists in partitioning a text into contiguous areas, by determining boundaries between them. These areas of text, hereafter called thematic segments, are not precisely defined. Roughly said, they form semantically coherent parts of text, large enough to expose some aspect of a given topic. For example, paragraphs or selected passages are classical types of segments usually used in text summarization.

From another point of view, thematic segmentation of texts can be seen as the result of a gathering process which aggregates small pieces of text, in fact document units, into larger units in order to highlight local semantical coherences [Kozima 1993]. The granularity level of the segmentation depends on the size of the units to be grouped. Here, sentences have been chosen as discourse units for the experiments, but other units, such as words or paragraphs, could have been considered as well.

Increasing interest in thematic segmentation of texts is mainly explained by the number of its applications [Baeza-Yates and Ribeiro-Neto 1999], such as text alignment, document summarization, information extraction, or information retrieval. Text alignment algorithms, such as those used in multi-lingual alignment tasks, may rely on discourse units resulting from a preliminary segmentation process to establish correspondences between finer units. Thematic text segmentation could be used in information retrieval to extract relevant parts in the documents returned as responses to a user's query, in the context of document summarization [McDonald and Chen 2002] to detect the general structure of documents and, at last, in information streaming to select interesting data.

Many segmentation methods have been proposed and most of them rely on statistical approaches such as Text Tiling [Hearst 1997], C99 [Choi 2000], Dot-Plotting [Reynar 2000] or Segmenter [Kan et al. 1998]. They are based on the distribution of the words in the text, in order to determine the thematic changes by the way of lexical inventory variations in fixed size windows. On the other hand, particular linguistic methods introduce specific rules with regard to a given corpus and use semantic knowledge such as ontologies or thesauri. Good results could be obtained so but their portability to other domains is dependent on the existence of intrinsic semantic resources [Utiyama and Isahara 2001]. At last, some other methods are based on learning such as [Amini et al. 2000], which relies on hidden Markov's models or [Caillet et al. 2004], which proposes a term clustering. They are indeed efficient but depend on the thematic domain concerned by the learning process, should it be supervised or not.

This paper describes ClassStruggle, a new linear segmentation method, which is based on a initial clustering of the sentences of the text. Section 2 presents main motivations and basis of our work, section 3 consists in a description of the algorithm, section 4 describes some experiments of parameters tunning and, at last, section 5 evaluates ClassStruggle by comparison with competitive systems.

2 Motivations and related works

ClassStruggle belongs to the family of methods addressing segmentation by lexical chains such as [Galley et al. 2003] or [Utiyama and Isahara 2001]. These methods are mainly based on the repetitions of the terms in order to segment a text (and occasionally on synonymy, generalization/specialization or part-whole/whole-part relationships [Morris and Hirst 1991]). When a term is repeated in a more or less short distance (called a hiatus), a lexical chain is created between these two occurrences. Thematic boundaries are set in the text at places where the number of chains is minimal. These methods then attempt to segment texts at places where the local cohesion is the lowest. However, these methods do not consider the context of the repeated terms occurrences and a question remains: Is the simultaneity of active chains meaningful?

Figure 1: Example of lexical chains

[Fig. 1] depicts a configuration where lexical chains are searched for (each letter represents a part of text and the lines stand for the lexical chains). This highlights the limits of these methods: since lexical chains methods cut the text at the places where the number of chains is minimal, they set thematic frontiers before C and before G. However, according to the fact that C and D have nothing in common with E and F, it would also certainly be appropriate to set a boundary before E.

In order to obtain a more global view of the text and a finer analysis of the relations between sentences — and then eliminate this kind of problems — we propose to introduce an initial clustering process in the segmentation method.

In [Bellot 2000], P. Bellot has shown that there exists a strong relationship between textual clustering and segmentation. His hypothesis states that if two adjacent sentences belong to two different classes (In the following, the terms class and cluster indifferently refer to the same concept: a group of sentences), then it is possible to set a boundary between them. This assumption appears too strong since text segmentation not only depends on the similarity of the sentences, but must also consider their sequential organisation in the text. Indeed, a part of a text related to a particular topic may contain sentences deviating somewhat from it. These sentences may be classified differently from their neighbors. However, this certainly does not imply that a boundary has necessarily to be set there. Moreover, this assumption is too dependent on the chosen clustering mechanism, no existing clustering method being fully reliable. To overcome these biases and to improve robustness, we attempt to give more flexibility to the segmentation process w.r.t. the initial clustering step.

3 The segmentation method

Given a text, our segmentation method includes, as a first step, a clustering of the sentences w.r.t. their similarities. After having described this process, the remaining of this section is devoted to the segmentation algorithm itself.

3.1 The clustering method

In order to group sentences in clusters w.r.t. their similarity, sentences are first encoded in vectors, following the vectorial model (i.e., a vector of weights w.r.t. the set of meaningful terms) [Baeza-Yates and Ribeiro-Neto 1999].

The proximity between two vectors $\mathbf{p_1}$ and $\mathbf{p_2}$ of two parts of text p_1 and p_2 can be computed as a cosine measure:

$$\cos(\mathbf{p_1}, \mathbf{p_2}) = \frac{\sum_{i=1}^{t} w_{i,\mathbf{p_1}} \times w_{i,\mathbf{p_2}}}{\sqrt{\sum_{i=1}^{t} w_{i,\mathbf{p_1}}^2 \times \sum_{i=1}^{t} w_{i,\mathbf{p_2}}^2}} \tag{1}$$

where t is the number of meaningful terms, $w_{i,\mathbf{p}}$ the weight of the term i in the vector $\mathbf{p}$. Here, the weights of the terms in the vectors are simply their frequency within the text in concern. Earlier works weighted the terms according to their frequency times their inverse document frequency but simple term frequencies appear more efficient [Hearst 1997]. This formula computes a score corresponding to the similarity of the two concerned parts of text.

The clustering method used here is a variant of the "Single-Pass" algorithm [Frakes and Baeza-Yates 1992]. Let $\mathcal{S_R}$ be the set of the sentences being not assigned yet to any cluster. For each cluster creation, as long as $\mathcal{S_R}$ is not empty, we operate as follows:

1. **Cluster initialisation:** The two closest sentences of $\mathcal{S_R}$ are grouped together in a new cluster C.
2. **Sentence research:** Research of the sentence $smax$ in $\mathcal{S_R}$ being the closest to the sentences of C:
   $smax = \arg\max_{u \in \mathcal{S_R}} \sum_{s \in C} \cos(\mathbf{u}, \mathbf{s})$.
3. **Sentence adding:** Computation of the average of the cosine scores of every pairs of sentences belonging to C. If this score decreases after the adding of $smax$ in C, this sentence is removed from the cluster. In that case, the

creation of C is finished. In the other case, the process returns to stage 2 in order to seek another sentence that could belong to C.

This method is not optimal since the resulting clusters strongly depend on the order in which the sentences are processed [Zamir and Etzioni 1998]. Nevertheless, since our purpose is to assess the robustness of the segmentation w.r.t. the errors of clustering, we may assume that the worse this clustering is, the more conclusive the results would be. In our experiments, an average of 12 classes per document of 100 sentences and of 7 classes per document of 50 sentences have been produced.

3.2 Segmentation algorithm

ClassStruggle is based on the distribution of the occurrences of the members of each class. Classes have been created only with regards to the similarities between sentences. Our purpose is now to define a re-clustering process in order to take into account a notion of sequential organisation of the text. During the process, sentences move from a class to another, w.r.t. the classes of their neighborhood, in the aim to create groups that contain only sentences related to a same topic development. Boundaries are finally created between sentences belonging to different classes.

Let $\mathcal{U}$ be the sequence of the nu sentences $U_1 \dots U_{nu}$ of the initial text. After the initial clustering process [Section 3.1], we have a set C of nc classes, each containing distinct subsets of $\mathcal{U}$. Then, the algorithm parses the text in order to attribute a score of potential membership of each sentence to each class. These scores are computed using [Eq. 2]:

$$S(C_k, U_i) = Sim(C_k, C_{c_i}) + \beta \times S(C_k, U_{i-1}) + \beta \times S(C_k, U_{i+1}) \qquad (2)$$

where $Sim(C_k, C_{c_i})$ represents the similarity between the class C_k and the class C_{c_i} that contains the sentence U_i [Eq.4]. This formula [Eq. 2] takes into account the similarity of the current class with the class that contains the considered sentence as well as the scores it obtained on the preceding $(S(C_k, U_{i-1}))$ and on the following $(S(C_k, U_{i+1}))$ sentences. These two last scores enable us to introduce the concept of proximity in term of distance in the text. They are both weighted by a propagation coefficient β (similar to the notion of hiatus mentioned in [Section 2] that has to be determined empirically. It allows us to adjust the influence of the context. The formula [Eq. 2] can be more formally rewritten:

$$S(C_k, U_i) = \sum_{j=1}^{nu} \beta^{|i-j|} \times Sim(C_k, C_{c_j}) \qquad (3)$$

The general algorithm can be sketched as follows:

1. **Scores computation:** Computation of the membership values of each sentence to each class [Eq. 3];
2. **Classes assignment updating:** Sentences having the same best dominant class according to their scores are grouped. These groups form the new classes to be considered. If the new classes are different from the old ones, the similarities between classes are recomputed and the algorithm restarts at stage 1;
3. **Boundaries setting:** Setting boundaries between all adjacent sentences belonging not to the same class.

The similarity function between classes $Sim(C_k, C_{k'})$ uses the cosine measure [Eq. 1] extended to classes. Therefore, we could have reduced $Sim(C_k, C_{k'})$ to $\cos(\mathbf{C_k}, \mathbf{C_{k'}})$ but, in that case, a problem occurs when considering the score of potential membership of a sentence to the class already containing it, since $C_k = C_{c_i}$ and thus $\cos(\mathbf{C_k}, \mathbf{C_{c_i}}) = 1$. This gives too much importance to the class containing the sentence compared to the others, value 1 being generally quite higher than the values of similarity between classes. In this case, the context does not influence the computation enough and classes would never change. We thus decided to replace this value by the maximal existing similarity between classes, weighted by a coefficient α, empirically determined. The similarity function $Sim(C_k, C_{k'})$ is defined as:

$$Sim(C_k, C_{k'}) = \begin{cases} \cos(\mathbf{C_k}, \mathbf{C_{k'}}) \quad \text{if } k \neq k', \\ \\ \alpha \times \max_{\{(x,y) \in [1,nc]^2, x \neq y\}}(\cos(\mathbf{C_x}, \mathbf{C_y})) \quad \text{else.} \end{cases} \tag{4}$$

where $\max(\cos(\mathbf{C_x}, \mathbf{C_y}))$ is the maximal existing similarity, C_x and C_y being two distinct classes of our set of nc classes. Using this formula, the score of the class containing the considered sentence would be more homogeneous with regards to the scores obtained by the other classes.

The relaunching of the process at the stage 2 of the algorithm permits to give more weight to the dominant class of a text area and thus to better handle the irregularities of the document. When a sentence has been associated to a given class, it is indeed advisable to reflect the changes in the computation of the frontiers. The purpose is then to converge into a stable segmentation. During the experiments conducted in [Section 5], the numbers of iterations needed to reach the stability of segmentation have been recorded for each execution of ClassStruggle in order to assess the convergence ability of the algorithm. According to the results obtained, ClassStruggle always converges relatively quickly to a stable state (the maximal number of iterations it needed to reach this goal is

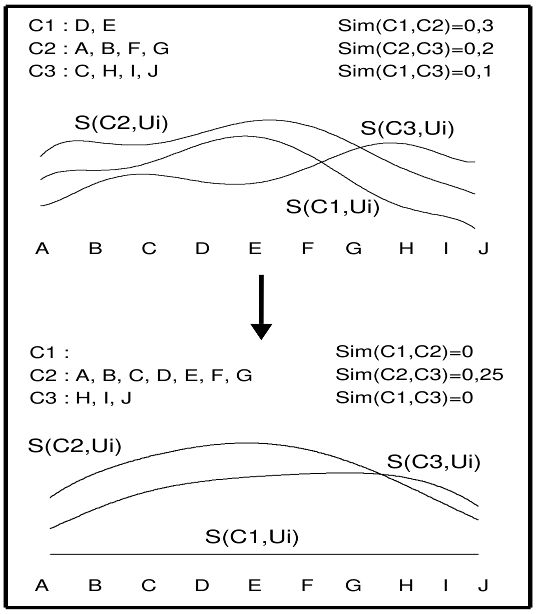

Figure 2: Example of segmentation by ClassStruggle

15). The clustering method, the weight of the context and the size of the texts seem to slightly influence the convergence but the differences are not significant, the convergence appears thus to be relatively robust.

3.3 Example

[Fig. 2] is an example of the ClassStruggle's segmentation process. Let us consider a script size text composed of ten sentences (from A to J), extracted from two different newspaper's articles: sentences from A to G come from the first article and sentences from H to J come from the second one. Let us also assume that three clusters C1, C2 and C3 have been generated by the preliminary clustering process mentioned in [Section 3.1]. The value of the membership score of the sentences w.r.t. the three classes [Eq. 2] is represented by means of curves whose vertical height is proportional to these scores. Since sentence C contains some terms in common with sentences H, I and J, it has been grouped with them. Sentences D and E, which are very similar, have been separated from other sentences. Note that, using the initial assumption mentioned in [Section 2], we would have produced five segments: AB, C, DE, FG and HIJ. Here, the algorithm reaches a stable state after the second iteration and finally creates a boundary between G and H.

4 Parameters tuning

4.1 Experimental process

The experiments were carried out on articles from the "AP" corpus of texts of the international conference TREC-1 [Harman 1993], which contains full articles from the Associated Press published between 1988 and 1989. Those articles have been gathered to form different sets of 350 documents. Separations between articles constitute the segmentation of reference. These tests based on concatenated articles may be less conclusive than tests performed on more homogeneous texts but this evaluation seems to be the most commonly used in the literature (see for example [Choi 2000]) and, according to the existing methods results, this kind of boundaries appears to be difficult enough to be recognized. According to this principle, four corpuses have been created in order to test the behavior of the algorithm w.r.t. different kinds of text. We note $AP(ns, nb)$ a corpus composed by texts of ns sentences and an average of nb boundaries. We use the corpuses $AP(50, 2), AP(50, 4), AP(100, 4), AP(100, 8)$. The segmentation algorithms are evaluated w.r.t. three criteria: the precision (number of exact boundaries found / number of boundaries found), the recall (number of exact boundaries found / number of boundaries of the reference) and a criterion, called WindowDiff [Pevzner and Hearst 2002], which takes into account the number of boundaries between two sentences separated from a distance k.

$$WindowDiff(hyp, ref) = \frac{1}{N-k}\sum_{i=0}^{N-k}(|b(ref_i, ref_{i+k}) - b(hyp_i, hyp_{i+k})|) \quad (5)$$

where $b\,(x_i, x_j)$ is the number of boundaries between i and j in a segmented text x containing N sentences, ref corresponds to the segmentation of reference and hyp the one found with the method to evaluate. The size k of the window has to be set to half of the average true segment size. Among the interesting properties of this measure is its ability to cope with near-misses.

4.2 Experimental tuning

The experiments conducted here concern the research of optimal values for the two parameters of the algorithm, α and β. The coefficient of profit α must be a little greater than 1 since the cluster containing the sentence in question has to gain at least as much as the other clusters in order to converge to a stable segmentation [see Section 3.2]. The coefficient of propagation β must be higher than 0, otherwise the context has no influence, and lower than 1 since, in that case, a cluster is likely to dominate all the others in a permanent way, preventing the achievement of any segmentation. In order to determine the

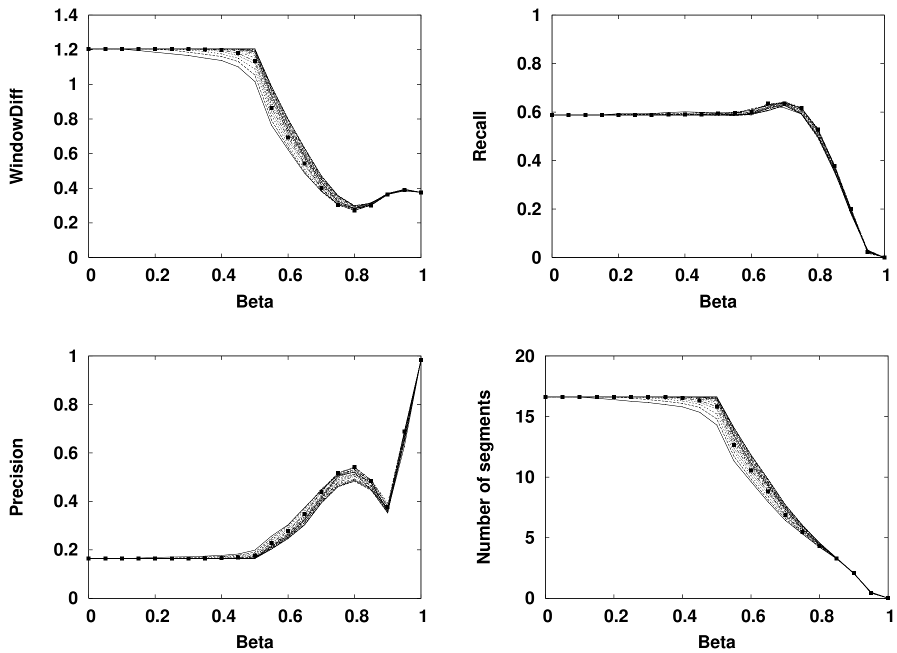

Figure 3: Results of the experiments

optimal coefficients α and β, we have test all combinations (α, β) between $(1, 0)$ and $(2, 1)$ while varying α and β by step of 0.05. The figure [Fig. 3] gives the results of these experiments in term of WindowDiff, recall, precision and number of segments obtained, each curve representing a set of experiments (each point corresponding to a test over the 350 benchmark documents) for a given value of α, β varying from 0 to 1. The curve whose points are highlighted corresponds to $\alpha = 1.25$.

4.2.1 α tuning

The search for an optimal value α according to β corresponds to a balance between the score assigned to the class containing the current sentence and the scores coming from the environment. Therefore, when β is low (i.e., the context is of few significance) the lowest values of α are the best and when β increases, the method works better with a slightly higher α. We remark that when α exceeds the value of 1.25, the score assigned to the cluster containing the concerned sentence becomes too high and involves then a loss of global efficiency. Given an optimal value for β, the different values of α induce roughly the same behavior. α is only of a little influence on the results and does not really perturb the efficiency of the segmentation method.

4.2.2 β tuning

The different curves stagnate until $\beta = 0.5$. From this value, the scores history induces a significant weight in the computation of the dominant cluster. According to WindowDiff and precision, $\beta = 0.8$ corresponds to the best results but a lower value provides a better recall. This is due to the fact that the higher the value of propagation of the signal is, the higher chance a dominant cluster has to influence its neighboring units and to reduce the number of obtained segments. In spite of the decrease of the number of segments, the precision fall for $0.8 \leq \beta \leq 0.9$ corresponds to an obvious loss of quality in the segmentation (the precision should increase with the fall of the number of segments). The dominant clusters have too much impact on their surroundings and they overlap the "territory" of the other clusters, involving a shifted (or canceled) segmentation. Therefore, the optimal value of β seems to be located between 0.7 and 0.8, according to the required frequency of segmentation. The fact that the optimal value for β is really greater than 0 shows that our approach has a significant interest compared to the initial hypothesis [Section 2]. Over the optimal β values interval, $\alpha = 1.25$ provides the best results.

5 Evaluation

5.1 Experimental process

In order to evaluate the methods, we use benchmarks created similarly to those described in [Section 4]. Of course, in order not to skew the results, the selected articles are different from those used for tuning purposes. Two additional corpuses has been created in order to assess the robustness of the methods and their ability to segment more specific texts. In this way, articles from the "ZIFF" corpus of texts of TREC-1 [Harman 1993], which contains full articles taken from computer science magazines, have been gathered to form two sets of 350 documents, noted $ZF(100, 4)$ and $ZF(100, 8)$. These corpuses, being more specific than others and thus containing semantically closer articles, simulates better the segmentation of real-world texts.

ClassStruggle has been compared to the following methods to evaluate its efficiency:

- **Rand**: Given a corpus $T(ns, nb)$, this procedure creates randomly nb boundaries and represents a reference point for the experiments.

- **TT**: The TextTiling Method [Hearst 1997], which is certainly the most popular existing method, is an extension of [Morris and Hirst 1991]. It is based on the distribution of the terms in the text by giving scores of cohesion between textual units according to several criteria (number of common words, number of new words, number of active lexical chains). A window slides on the text and compares, at each position, two adjacent blocks of words. The main parameters of TextTiling are the size of the blocks to compare and the sliding step of the window. These parameters have been tuned on the training corpuses used in the [Section 4]. The best results on these corpuses have been obtained with blocks of 120 words and a step of 20 words. The experiments realized in this section use then these values.

- **C99**: The C99 method [Choi 2000] introduces a "local ranking" of the similarities between textual units by determining for each pair the number of surrounding pairs having a lower similarity in the k closest pairs. Then, the whole textual units are laid out on a 2D plan (in a similar way to the Dot-Plotting method [Reynar 2000]) according to their position in the text. The algorithm aims at finding the densest areas according to the rank of the units in the ranking carried out. The areas which maximize this density form the subtopic segments of the text. The main parameter of C99, the number k of pairs of sentences to consider, has been tuned on the training corpuses used in the [Section 4]. The best value on these corpuses appeared to be $k = 11$, this is thus the value used for the evaluation process.

Five versions of ClassStruggle are used for this evaluation:

- $\mathbf{C}_R$: A version of ClassStruggle that generates clusters randomly (12 per documents of 100 sentences and 7 per documents of 50) instead of using the clustering process described in [Section 3.1]. β is equal to $0,75$.
- $\mathbf{C}_{SP(0.7)}$, $\mathbf{C}_{SP(0.75)}$ and $\mathbf{C}_{SP(0.8)}$: Three versions of ClassStruggle using the clustering method "Single-Pass" described in [Section 3.1]. β is respectively equal to 0.7, 0.75 and 0.8.
- $\mathbf{C}_{DC(0.7)}$, $\mathbf{C}_{DC(0.75)}$ and $\mathbf{C}_{DC(0.8)}$: Three versions of ClassStruggle using a better partitioning of sentences than other versions. The clustering process used is the same but the partitioning is refined by a "Dynamic Clouds" algorithm [Frakes and Baeza-Yates 1992], which reaffects each sentence to its most similar cluster. The process converges to a stable state in an average of ten iterations. It enables to overcome the problem of the "Single-Pass" algorithm which is too much dependent on the order of the sentences. This version allows us to assess the influence of the clustering quality on the segmentation accuracy. β is respectively equal to 0.7, 0.75 and 0.8.

[Tab. 1] gives results of methods on each AP corpus and [Tab. 2] on the additional corpuses $ZF(100, 4)$ and $ZF(100, 8)$, according to WindowDiff (W), precision (P), recall (R) and number of boundaries created (N).

AP(50,2)	W		P		R		N	
	μ	σ	μ	σ	μ	σ	μ	σ
$Rand$	0.46	0.21	0.05	0.19	0.05	0.19	2.02	0.98
TT	1.02	0.71	0.26	0.22	0.54	0.38	4.21	1.39
$C99$	0.89	0.75	0.32	0.22	0.63	0.38	4.22	1.42
C_R	0.93	0.90	0.09	0.19	0.16	0.29	3.88	2.76
$C_{SP(0.7)}$	0.43	0.42	0.49	0.30	0.68	0.35	3.39	1.66
$C_{SP(0.75)}$	0.36	0.42	0.54	0.36	0.62	0.38	2.62	1.40
$C_{SP(0.8)}$	0.31	0.35	0.55	0.39	0.52	0.39	2.10	1.20
$C_{DC(0.7)}$	0.40	0.41	0.50	0.30	0.72	0.38	3.12	1.43
$C_{DC(0.75)}$	0.33	0.34	0.54	0.34	0.68	0.37	2.79	1.38
$C_{DC(0.8)}$	0.26	0.37	0.58	0.40	0.60	0.36	2.11	1.10

AP(50,4)	W		P		R		N	
	μ	σ	μ	σ	μ	σ	μ	σ
$Rand$	0.53	0.14	0.06	0.12	0.06	0.12	4.03	1.45
TT	0.40	0.26	0.49	0.26	0.49	0.26	4.31	1.40
$C99$	0.38	0.31	0.48	0.23	0.59	0.27	4.94	1.31
C_R	0.50	0.39	0.18	0.22	0.18	0.20	4.24	3.16
$C_{SP(0.7)}$	0.29	0.21	0.57	0.27	0.65	0.29	4.67	1.78
$C_{SP(0.75)}$	0.27	0.20	0.60	0.29	0.58	0.30	3.91	1.54
$C_{SP(0.8)}$	0.28	0.17	0.61	0.38	0.49	0.29	3.03	1.37
$C_{DC(0.7)}$	0.27	0.21	0.58	0.28	0.67	0.29	4.47	1.62
$C_{DC(0.75)}$	0.23	0.17	0.60	0.27	0.60	0.29	3.96	1.60
$C_{DC(0.8)}$	0.25	0.16	0.63	0.38	0.52	0.31	3.42	1.51

AP(100,4)	W		P		R		N	
	μ	σ	μ	σ	μ	σ	μ	σ
$Rand$	0.55	0.15	0.04	0.09	0.04	0.09	4.20	1.57
TT	0.97	0.66	0.27	0.14	0.59	0.25	9.52	2.33
$C99$	0.54	0.45	0.40	0.19	0.64	0.26	6.81	2.09
C_R	0.79	0.42	0.07	0.19	0.12	0.17	7.07	2.96
$C_{SP(0.7)}$	0.40	0.41	0.48	0.23	0.65	0.27	7.21	2.94
$C_{SP(0.75)}$	0.33	0.29	0.52	0.26	0.60	0.27	5.23	2.29
$C_{SP(0.8)}$	0.29	0.23	0.53	0.28	0.52	0.28	4.37	1.86
$C_{DC(0.7)}$	0.37	0.39	0.50	0.23	0.68	0.28	7.12	2.83
$C_{DC(0.75)}$	0.30	0.31	0.53	0.26	0.66	0.27	5.30	2.23
$C_{DC(0.8)}$	0.25	0.25	0.55	0.28	0.53	0.29	4.30	1.82

AP(100,8)	W		P		R		N	
	μ	σ	μ	σ	μ	σ	μ	σ
$Rand$	0.57	0.10	0.08	0.09	0.08	0.09	8.15	2.30
TT	0.37	0.16	0.47	0.17	0.56	0.18	9.82	2.03
$C99$	0.33	0.16	0.53	0.17	0.59	0.20	9.15	2.23
C_R	0.55	0.14	0.17	0.16	0.14	0.13	7.49	3.03
$C_{SP(0.7)}$	0.30	0.16	0.54	0.18	0.63	0.21	9.61	2.80
$C_{SP(0.75)}$	0.28	0.13	0.57	0.21	0.56	0.22	7.86	2.45
$C_{SP(0.8)}$	0.29	0.12	0.59	0.28	0.48	0.20	6.11	2.26
$C_{DC(0.7)}$	0.29	0.17	0.55	0.19	0.64	0.21	9.44	2.46
$C_{DC(0.75)}$	0.26	0.13	0.60	0.19	0.60	0.20	8.10	2.35
$C_{DC(0.8)}$	0.28	0.14	0.61	0.19	0.51	0.20	6.49	2.33

Table 1: Results of the methods on the AP corpuses

5.2 Results

According to the results obtained on the evaluation corpuses [Tab. 1 and 2], ClassStruggle appears globally more efficient than C99 and TextTiling. First, C99 and overall TextTiling seem to have some difficulties to adapt themselves w.r.t. the number of boundaries to retrieve, the length of the texts has a great impact on their frequency of segmentation. ClassStruggle appears to better follow these variations.

As mentioned in [Section 4.1], WindowDiff is a better measure of accuracy than recall and precision since it enables to consider near-misses. However, given two methods, WindowDiff makes sense only if they create a similar amount of boundaries. In fact, a method, which creates more boundaries, has a greater chance to obtain a high score. Therefore, the WindowDiff scores obtained by

ZF(100,4)	W		P		R		N	
	μ	σ	μ	σ	μ	σ	μ	σ
$Rand$	0.55	0.12	0.04	0.07	0.04	0.07	3.82	1.02
TT	0.85	0.62	0.25	0.12	0.58	0.23	8.78	2.08
$C99$	0.46	0.38	0.39	0.18	0.63	0.24	6.26	2.07
C_R	0.81	0.39	0.06	0.09	0.10	0.11	6.92	2.83
$C_{SP(0.7)}$	0.43	0.45	0.45	0.25	0.61	0.28	6.31	2.96
$C_{SP(0.75)}$	0.33	0.32	0.47	0.26	0.59	0.28	4.80	1.99
$C_{SP(0.8)}$	0.33	0.28	0.48	0.27	0.51	0.24	4.12	1.81
$C_{DC(0.7)}$	0.42	0.42	0.45	0.24	0.62	0.24	6.35	2.31
$C_{DC(0.75)}$	0.33	0.31	0.47	0.26	0.60	0.27	5.01	2.17
$C_{DC(0.8)}$	0.32	0.28	0.48	0.27	0.53	0.23	4.35	1.75

ZF(100,8)	W		P		R		N	
	μ	σ	μ	σ	μ	σ	μ	σ
$Rand$	0.57	0.09	0.07	0.07	0.07	0.07	8.04	1.93
TT	0.37	0.16	0.45	0.16	0.53	0.16	9.18	1.95
$C99$	0.34	0.16	0.50	0.16	0.54	0.17	8.68	1.87
C_R	0.56	0.15	0.16	0.16	0.12	0.10	7.33	3.05
$C_{SP(0.7)}$	0.32	0.15	0.50	0.16	0.56	0.20	8.76	2.66
$C_{SP(0.75)}$	0.30	0.18	0.52	0.18	0.54	0.23	6.80	2.30
$C_{SP(0.8)}$	0.29	0.15	0.53	0.19	0.47	0.17	5.93	1.42
$C_{DC(0.7)}$	0.31	0.15	0.50	0.17	0.57	0.15	8.82	2.83
$C_{DC(0.75)}$	0.30	0.17	0.53	0.20	0.55	0.19	7.02	2.46
$C_{DC(0.8)}$	0.29	0.19	0.54	0.20	0.50	0.13	6.13	1.64

Table 2: Results of the methods on the ZIFF corpuses

C99 and $C_{0.8}$ can not be compared. However, $C_{0.7}$ creates roughly the same number of boundaries than C99 (and even a little more) and we remark the better efficiency of ClassStruggle. Moreover, the recall and precision scores of $C_{0.7}$ are clearly better on every corpuses than those obtained by C99 and TextTiling. An additional Student t-test[1] has shown that these differences are statistically significant with a 99% confidence rate.

Moreover, the poor results obtained by the version using a random clustering C_R show the importance of the initial clustering step. The clustering process described in [Section 3.1] is relatively basic. Therefore, a more powerful technique may provide a great improvement in term of efficiency. The tests performed with C_{DC} support this assumption since their results are clearly the best ones.

On the specific corpuses, $ZF(100,4)$ and $ZF(100,8)$, ClassStruggle seems to loose a little of its advantage compared with the existing methods. Sentences of texts being more similar, the clustering process is more difficult. The refinement of the partition by dynamic clouds does not solve the problem. The use of more sophisticated clustering methods should improve the results of ClassStruggle, especially by using semantic resources.

6 Conclusion

The cutting out of a text into thematic segments depends on the whole set of elements composing it. The determination of a boundary cannot thus be effectively achieved in a local way. This observation led us to follow the way

[1] The 99% Student t-test is based on the average, the standard deviation and the cardinality of a set of runs. It uses a p-value equal to 2.57 to insure with a rate of confidence of 99% that the difference in means of two sets is significant.

of a global segmentation. ClassStruggle, by using a preliminary clustering of the sentences, deals with the set of the topics of the whole text. It does not use similarities between sentences individually but in a group, limiting then the impact of false clues (for example, repetitions of terms that have not anything in common because of their use in different contexts). Moreover, the assignment of a sentence to a given segment is influenced by every others [Eq. 2]. The relaunching of the process while the process has not reach a stable state leads to reconsider the whole segmentation when a boundary has been detected.

The suggested segmentation was evaluated by comparison with efficient existing methods. The results show that our approach provides a more accurate segmentation of the texts, considering the text in its whole to segment it appears then to be relevant. As future works, we have to test how improvements of the clustering of the units will be reflected on the segmentation. Indeed, three kinds of clustering techniques have been tested and results show that the accuracy of the segmentation realized by ClassStruggle increases with the quality of the partitioning. We then may expect better results by introducing more powerful clustering techniques.

Acknowledgments

This work was supported by "Angers Loire Metropole".

References

[Amini et al. 2000] Amini, M. Zaragoza, H. Gallinari, P.: "Learning for Sequence Extraction Tasks"; Proc. of 6th Conference on Content-Based Multimedia Information Access (2000), Paris, France, 476-490.

[Baeza-Yates and Ribeiro-Neto 1999] Baeza-Yates, R. Ribeiro-Neto, B.: "Modern Information Retrieval"; ACM Press/Addison-Wesley.

[Bellot 2000] Bellot, P.: "Méthodes de classification et de segmentation locales non supervisées pour la recherche documentaire"; PhD thesis, Université d'Avignon, France.

[Caillet et al. 2004] Caillet, M. Pessiot, J. Amini, M. Gallinari, P.: "Unsupervised learning with term clustering for thematic segmentation of texts"; Proc. of Rercherche d'Information Assiste par Ordinateur (2004), Avignon, France, 1-11.

[Choi 2000] Choi, F.: "Advances in domain independent linear text segmentation"; Proc. of the first conference on North American chapter of the Association for Computational Linguistics (2000), San Francisco, CA, USA, 26-33.

[Frakes and Baeza-Yates 1992] Frakes, W. B. Baeza-Yates, R. A.: "Information Retrieval: Data Structures & Algorithms"; Prentice-Hall.

[Galley et al. 2003] Galley, M. McKeown, K. Fosler-Lussier, E. Jing, H.: "Discourse segmentation of multi-party conversation"; Proc. of the 41st Annual Meeting on Association for Computational Linguistics (2003), ACL, Morristown, NJ, USA, 562-569.

[Harman 1993] Harman, D.: "Overview of the first trec conference"; Proc. of the 16th annual international ACM SIGIR conference on Research and development in information retrieval (1993), ACM, New York, NY, USA, 36-47.

[Hearst 1997] Hearst, M.: "Texttiling: segmenting text into multi-paragraph subtopic passages"; Computational Linguistics, 23(1), 33-64.

[Kan et al. 1998] Kan, M. Klavans, J. McKeown, K.: "Linear segmentation and segment significance"; Proc. 6th Workshop on Very Large Corpora (1998), ACL SIGDAT, 197-205.

[Kozima 1993] Kozima, H.: "Text segmentation based on similarity between words"; Proc. Meeting of the Association for Computational Linguistics (1993), 286-288.

[McDonald and Chen 2002] McDonald, D. Chen, H.: "Using sentence-selection heuristics to rank text segments in txtractor"; Proc. of the 2nd ACM/IEEE-CS joint conference on Digital libraries (2002), ACM, New York, NY, USA, 28-35.

[Morris and Hirst 1991] Morris, J. Hirst, G.: "Lexical cohesion computed by thesaural relations as an indicator of the structure of text"; Computational Linguistics, 17(1), 21-48.

[Pevzner and Hearst 2002] Pevzner, L. Hearst, M.: "A critique and improvement of an evaluation metric for text segmentation"; Computational Linguistics, 28(1), 19-36.

[Reynar 2000] Reynar, J.: "Topic Segmentation : Algorithms and applications". PhD thesis, University of Pennsylvania, Seattle, WA.

[Salton et al. 1996] Salton, G. Singhal, A. Buckley, C. Mitra, M.: "Automatic text decomposition using text segments and text themes"; Proc. Hypertext '96, The Seventh ACM Conference on Hypertext (1996), Washington DC, pages 53-65. ACM.

[Utiyama and Isahara 2001] Utiyama, M. Isahara, H.: "A statistical model for domain-independent text segmentation"; Proc. Meeting of the Association for Computational Linguistics (2001), 491-498.

[Zamir and Etzioni 1998] Zamir, O. Etzioni, O.: "Web document clustering: A feasibility demonstration"; Proc. Research and Development in Information Retrieval (1998), 46-54.