SegGen: a Genetic Algorithm for Linear Text Segmentation

S.Lamprier, T.Amghar, B.Levrat and F.Saubion
LERIA, Université d'Angers
2, Bd Lavoisier 49045 Angers (France)
{lamprier,amghar,levrat,saubion}@info.univ-angers.fr

Abstract

This paper describes SegGen, a new algorithm for linear text segmentation on general corpuses. It aims to segment texts into thematic homogeneous parts. Several existing methods have been used for this purpose, based on a sequential creation of boundaries. Here, we propose to consider boundaries simultaneously thanks to a genetic algorithm. SegGen uses two criteria: maximization of the internal cohesion of the formed segments and minimization of the similarity of the adjacent segments. First experimental results are promising and SegGen appears to be very competitive compared with existing methods.

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 units, disconnected from other adjacent parts [Salton *et al.*, 1996]. More precisely, segmentation partitions a text by determining boundaries between contiguous segments related to different topics, defining so semantically coherent parts of text that are sufficiently large to expose some aspect of a given subject. Thematic segmentation of texts can also be seen as a grouping process of basic units (words, sentences, paragraphs...) in order to highlight local semantical coherences [Kozima, 1993]. The granularity level of the segmentation depends on the size of the units.

The increasing interest in text segmentation is mainly explained by the number of its applications such as text alignment, document summarization, information extraction, or information retrieval [Baeza-Yates and Ribeiro-Neto, 1999]. Text segmentation can be indeed very useful for these tasks by providing the structure of a document in terms of the different topics it covers [McDonald and Chen, 2002].

Many segmentation methods have been proposed and we focus here on the most general and significant methods that rely on statistical approaches such as Text Tiling [Hearst, 1997], C99 [Choi, 2000], DotPlotting [Reynar, 2000] or Segmenter [Kan *et al.*, 1998]. These methods perform an analysis of the distribution of the words in the text, in order to determine the thematic changes by means of lexical inventory variations in fixed size windows and thus create boundaries in the text where the local cohesion is the lowest.

In this paper, we introduce SegGen a genetic algorithm to achieve a statistical linear segmentation of texts. Section 2 presents the main motivations of our work. Our segmentation algorithm is described in section 3. Then, section 4 describes an experimental study of the algorithm in order to tune its parameters. Finally, section 5 evaluates SegGen by comparing it with other segmentation systems.

2 Motivations and Preliminary Works

In most of existing segmentation approaches, the relationships between sentences are usually very local. For example, the methods addressing segmentation by means of lexical chains[1] mainly use the repetitions of the terms in order to define thematic boundaries. More precisely, these methods cut the texts where the number of lexical chains is minimal. Nevertheless, the context of these multiple occurrences is not addressed neither the significance of simultaneous lexical chains at a given position in the text.

According to the preceding remark, we attempted to propose an alternative approach to the segmentation of texts by taking into account a more complete view of the texts. In [Bellot, 2000], P. Bellot has shown that there exists a strong relationship between clustering and text segmentation. His hypothesis states that it is possible to set a boundary between two adjacent sentences belonging to two different semantic classes. This assumption seems too strong since text segmentation not only depends on the similarity of the sentences, but must also consider their layout in the text. Indeed, discourse structures of documents may be very diverse and a part of a text related to a particular topic may contain sentences deviating somewhat from it. These sentences are likely to 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.

[1] Lexical chains are formed between two occurrences of a same term [Galley *et al.*, 2003] [Utiyama and Isahara, 2001] and occasionally between synonyms and terms having statistical associations such as generalization/specialization or part-whole/whole-part relationships [Morris and Hirst, 1991]

However, we were convinced that a preliminary clustering could provide a more complete view of the sentences relations. Therefore, we have defined a first segmentation method, called ClassStruggle, that uses an initial clustering of the sentences of the text based on their similarity, in order to have a global view on their semantic relations. In this approach, the resulting clusters evolve by taking into account their proximity in the text. Considering the clusters as topics of the text, ClassStruggle performs a linear traversal of the document in order to determine the most appropriate class assignment for each sentence, depending on their context of occurrence. This process goes on as long as modifications occur in the clusters. Finally, boundaries are created between sentences belonging to different classes. Due to a lack of space, ClassStruggle cannot be fully described but is nevertheless used for the evaluation of SegGen (section 5).

According to the definition of text segmentation by Salton [Salton *et al.*, 1996], the internal cohesion of the segments and their semantical proximity with their neighborhood constitute important factors of an efficient segmentation method. But, in the case of a sequential creation of boundaries, these characteristics cannot be computed since, when a boundary is created, the concerned segment is not yet entirely delimited. Therefore, sequential methods need to introduce a concept of window in which cohesion measures can be achieved. Nevertheless, it is difficult to determine the size of the window: a too short window may lead the algorithm not to consider some cohesion clues and a too long one may take into account some repetitions of terms that should indeed not be associated, since used in different contexts.

From this second remark, we decided to consider an approach where the boundaries are considered globally. Based on an evaluation of the possible segmentation schemes, such a method would have a complete view of the text and would be able to compute coherence without using windows. Considering the segmentation problem as a combinatorial problem, SegGen is an evolutionary algorithm that evaluates the segmentations of the whole text rather than setting boundaries incrementally. The lack of knowledge on the structure of the text or on the number of segments to create, induces a huge search space and leads us to consider a genetic algorithm to cope with complexity.

3 SegGen: a Genetic Algorithm for Text Segmentation

Inspired by the mechanisms of natural selection and evolution, genetic algorithms [Holland, 1975; Goldberg, 1989] have been successfully applied to the resolution of numerous combinatorial optimization problems. The general principle of genetic algorithms consists in managing a population of individuals, which represents potential configurations of the problem to be solved. Genetic algorithms have been used for several text mining purposes (such as document clustering [Raghavan and Agarwal, 1987]).

As mentioned above, text segmentation can be viewed as finding parts of text with strong intrinsic relationships, disconnected from other adjacent parts. Therefore, we consider text segmentation as a multi-objective problem with two criteria to optimize: the internal cohesion of the segments and the dissimilarity between adjacent ones. As usual for multi-objective problems, these two criteria are negatively correlated. Indeed, the internal cohesion tends to increase with the number of boundaries whereas the dissimilarity between adjacent segments tends to decrease with it. Following the multi-objective optimization principles, we consider the objective functions separately in order to allow our algorithm to preserve enough diversity in the search process and to extract good segmentation proposals.

Our SegGen algorithm is a variant of the ”Strength Pareto Evolutionary Algorithm” (SPEA) [Zitzler, 1999], an elitist algorithm for multi-objective problems. The method uses an external archive $\bar{P}$ to memorize the non-dominated[2] individuals w.r.t. both criteria and a current population P_t. Individuals are selected from these two populations in order to generate new individuals thanks to genetic operators. These new individuals constitute the next P_t and are used to update $\bar{P}$. $\bar{P}$ constitutes, at the end of the process, a set of potential segmentations of the text. The algorithm has then to extract an unique solution from this set (see section 3.5). All documents do not need the same number of generations to reach a satisfactory segmentation. Therefore, our stop criterion corresponds to a stagnation of the population evaluation.

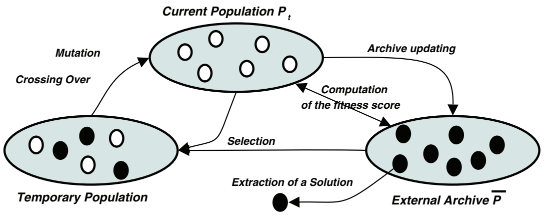

Figure 1: General functioning of SegGen

Algorithm 1 Pseudo-code of the Seggen algorithm

```
Similarities computation and populations initialization;
while $Stop\_Criterion$ not yet encountered do
    - Fitness evaluation of each individual of $P_t \cup \bar{P}$;
    - Selection, Crossover, Mutation;
    - Replacement of $P_t$ by the set of new individuals;
    - Update of $\bar{P}$ w.r.t. the non-dominated individuals;
end while
Selection of the best individual in $\bar{P}$.
```

3.1 Problem Representation

Given a text of ns sentences, the individuals are binary vectors $\vec{x}$ of $(ns - 1)$ elements x_i; $x_i = 1$ indicates that there is a boundary between sentence i and $i + 1$. For an individual $\vec{x}$, we consider two evaluation criteria: $C(\vec{x}) \in [0, 1]$, which corresponds to the internal cohesion of its segments,

[2] An individual is dominated if there exists an other individual in the populations having a better score on one criterion and at least the same score on the others.

and $D(\vec{x}) \in [0,1]$, which evaluates the dissimilarity between its adjacent segments. Our optimization problem consists in approximating the set of optimizers $\mathcal{O}$:

$$\begin{aligned}\mathcal{O} = \;& \{\vec{x} \in \{0,1\}^{ns-1} \mid \nexists \vec{x'} \in \{0,1\}^{ns-1}, \\ & \big(C(\vec{x}) < C(\vec{x'})\big) \wedge \big(D(\vec{x}) < D(\vec{x'})\big)\}\end{aligned} \tag{1}$$

According to this definition, the optimum is not a unique solution but a set of compromises, called the Pareto Front.

3.2 Initial Population Generation

Initial solutions are randomly and incrementally created but, in order to start with an heterogeneous population, individuals are created w.r.t. the boundaries of others. The size of the population is determined empirically (see section 4).

Genetic algorithms may converge easier if the individuals of the initial population are closer to the optimum [Goldberg, 1989]. Therefore, we tried to insert in the population an individual created by an external segmentation method. Experiments w.r.t. this strategy will be described in section 5.

3.3 Fitness Function

Two criteria are used to evaluate the individuals: internal cohesion and external dissimilarity. Both use similarities between sentences computed beforehand. According to the vectorial representation of the sentences, inspired by the vectorial model [Baeza-Yates and Ribeiro-Neto, 1999] (i.e., a vector of weights w.r.t. the set of meaningful terms), the similarity $sim(s_1, s_2)$ between two sentences s_1 and s_2 is computed with a cosine measure:

$$Sim(s_1, s_2) = \frac{\sum_{i=1}^{t} w_{i,s_1} \times w_{i,s_2}}{\sqrt{\sum_{i=1}^{t} w_{i,s_1}^2 \times \sum_{i=1}^{t} w_{i,s_2}^2}} \tag{2}$$

where t is the number of meaningful terms, w_{i,s_j} the weight of the term i in the sentence s_j.

Internal Cohesion

The internal cohesion of segments can be computed in two different ways by using the sentence similarities. First, it can be seen as the average internal cohesion of segments:

$$C(\vec{x}) = \frac{1}{nbseg} \times \sum_{i=1}^{nbseg} \frac{SumSim(seg_i)}{NbCouples(seg_i)} \tag{3}$$

with $nbseg$ being the number of segments of the individual, seg_i the segment i of the individual, $SumSim(seg_i)$ the sum of the sentence similarities of seg_i and $NbCouples(seg_i)$ the number of possible couples of sentences in seg_i.

Since the distribution of similarities in the text is not likely to be homogeneous, one may probably find very cohesive small areas. In this case, individuals representing small segments would have a really greater chance to obtain a good score of cohesion. A normalization of the score could be performed according to the size of the segments but may induce some bias. Therefore, we propose to realize the sum of each cohesion divided by the number of couples:

$$C(\vec{x}) = \frac{\sum_{i=1}^{nbseg} SumSim(seg_i)}{\sum_{i=1}^{nbseg} NbCouples(seg_i)} \tag{4}$$

In presence of long segments, the number of couples is greater than if all the segments have the same size (for example, $\binom{n+1}{2} + \binom{n-1}{2} > \binom{n}{2} + \binom{n}{2}$). Therefore, individuals having a lower variance in their segment size will be preferred. However, this bias is limited by the second objective function.

Dissimilarity Between Adjacent Segments

The similarity of a segment w.r.t. its successor is computed as follows:

$$SimSeg(seg_1, seg_2) = \frac{\sum_{s_j \in S(seg_1)} \sum_{s_k \in S(seg_2)} Sim(s_j, s_k)}{|S(seg_1)| \times |S(seg_2)|} \tag{5}$$

with seg_i being a segment, s_j a sentence, $S(seg_i)$ the set of the sentences of the segment seg_i and $|S(seg_i)|$ the cardinality of this set. This formula allows us to define the computation of the general dissimilarity between adjacent segments in an individual:

$$D(\vec{x}) = 1 - \left(\frac{\sum_{i=1}^{nbseg-1} SimSeg(seg_i, seg_{i+1})}{nbseg - 1} \right) \tag{6}$$

with $nbseg$ being the number of segments of $\vec{x}$ and seg_i a segment of this individual.

Fitness of the Individuals

Following [Zitzler, 1999], the computation of the fitness value of each individual is realized in two steps. A hardness value H is given to each individual of the external archive $\bar{P}$. This value is computed according to the number of individuals of P_t (the current population) that an element $\vec{x} \in \bar{P}$ slightly dominates on both objective functions:

$$H(\vec{x}) = \frac{|\{\vec{y}/\vec{y} \in P_t \wedge C(\vec{x}) \geq C(\vec{y}) \wedge D(\vec{x}) \geq D(\vec{y})\}|}{|P_t| + 1} \tag{7}$$

The fitness value of an individual $\vec{x} \in \bar{P}$ is equal to the inverse of its hardness value: $F(\vec{x}) = \frac{1}{H(\vec{x})}$. The fitness value of an individual $\vec{y} \in P_t$ is computed by summing the hardness scores of individuals $\vec{x}$ of $\bar{P}$ dominating $\vec{y}$:

$$F(\vec{y}) = \frac{1}{1 + \sum_{\vec{x} \in \bar{P} \wedge C(\vec{x}) \geq C(\vec{y}) \wedge D(\vec{x}) \geq D(\vec{y})} H(\vec{x})} \tag{8}$$

This fitness function, allowing us to select individuals for mutation and crossover, aims to diversify the search (from $\bar{P}$ point of view).

3.4 Exploration Operators

Three operators are used in the evolution process: selection, crossover and mutation (see fig.1). At each generation, the algorithm select $NbIndiv$ individuals and produces $NbIndiv$ new individuals to maintain a fixed size of population.

The individuals are selected from $\bar{P}$ and P_t according to their fitness value by "roulette wheel" [Holland, 1975]. This method first sums all the fitness scores of the individuals. Each individual gets a probability of selection equal to their percentage of contribution in this sum.

Among the selected individuals, while the number of $NbIndiv$ new individuals has not been reached yet, two parents are randomly chosen in order to produce two new individuals (the offsprings) by crossover. Here, we use a classical single point crossover: a position is randomly chosen in the parents and the two corresponding parts of the parents are exchanged to form two children.

Two different mutation operators are applied to these children: a mutation that replaces a parent by a randomly produced individual with a probability Pms and a mutation used with a probability Pmc that shifts a boundary of the individual to the next or the previous sentence. Both mutation operators appear complementary since the first enables to explore new areas while the second refines the existing solutions.

3.5 Extraction of a Solution

When the algorithm meets the stop criterion, the archive $\bar{P}$ contains a set of potential segmentations. We have then to extract the best individual from this set. This choice can be guided by an aggregation of both objective functions:

$$Agg(\vec{x}) = C(\vec{x}) + \alpha \times D(\vec{x}) \tag{9}$$

The extracted solution is the one having the best score w.r.t. this aggregation function. The coefficient α weights the second objective compared to the first. It acts upon the features of the final segmentation of the text and is thus tuned experimentally in section 4.

4 Experiments

4.1 Experimental Process

The experiments were carried out on articles from the corpus of texts of the international conference TREC-1 [Harman, 1993], which includes 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 this kind of boundaries appears to be difficult enough to recognize. According to this principle, four corpuses have thus been created in order to test the behavior of the algorithm with regards to different kinds of text. We note $T(ns, nb)$ a corpus composed by texts of ns sentences and an average of nb boundaries.We use the corpuses $T(50, 2), T(50, 4), T(100, 4), T(100, 8)$. 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.

$$\begin{aligned} WinDiff(hyp, ref) = \tfrac{1}{N-k} \quad \textstyle\sum_{i=0}^{N-k}(|b(ref_i, ref_{i+k}) \\ -b(hyp_i, hyp_{i+k})|) \end{aligned} \tag{10}$$

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. This criterion enables to consider the near-misses.

4.2 Parameters Tuning

The quality of the population is evaluated with a function $Eval(\bar{P})$ that returns the score of the best individual in $\bar{P}$ w.r.t the aggregation function (section 3.5):

$$Eval(\bar{P}) = Max_{\vec{x} \in \bar{P}} Agg(\vec{x}) \tag{11}$$

A study of different $\bar{P}$, issued from executions of SegGen on multiples texts after various numbers of generations, has shown that a weight $\alpha = 6$ enables the selection of a good individual in the majority of cases. The following tests will use this value.

In order to adjust the probabilities of mutation Pms and Pmc, SegGen has been ran for each couple of (Pms, Pmc) between $(0, 0)$ and $(1, 1)$, using a step of 0.2 for each. These experiments have been carried out on the set of 350 documents of the corpus $T(50, 2)$, 10 times each in order to limit random variations. The number of individuals in the population is set to 10, value which appears to provide good results. Results of figure 2 are averages of the population evaluation function $Eval(\bar{P})$ on these multiple executions.

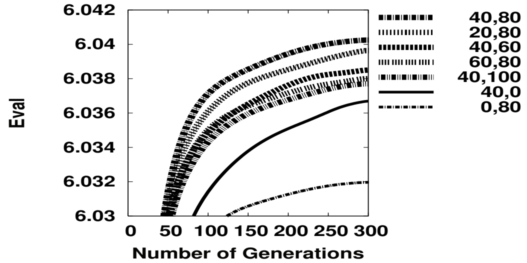

Figure 2: Evolution of $Eval$ w.r.t. Pms, Pmc

The figure 2 shows the evolution of $Eval(\bar{P})$ according to some different combinations of mutation probabilities[3]. We may first notice the influence of both mutation operators since the curves that corresponds to a single mutation operator are

[3] The legends correspond to the values Pms, Pmc (times 100) and are given in the same order than the obtained score at the end of the process (the greatest is the best).

the lowest. The couple $Pms = 0.4$ and $Pmc = 0.8$ seems to be the best compromise.

A study on the number of individuals $NbIndiv$ has shown that values greater than 10 allow the algorithm to converge more quickly in term of generations but induce a greater computation cost in term of evaluation. For instance, on the corpus $T(50, 2)$, with $NbIndiv = 20$, the algorithm needs only 150 generations to reach the score of evaluation of 6.04 against 250 with 10 individuals. However, the number of generated individuals is greater ($250 \times 10 < 150 \times 20$). On the other hand, a lower number of individuals leads to the opposite problem, the number of generations is too high, which implies a loss of time.

Concerning the stop criterion, long texts may need more time to be segmented but the quality of segmentation can be the same than for shorter ones. A value of 100 iterations without any improvement seems to be sufficient enough to reach a good solution.

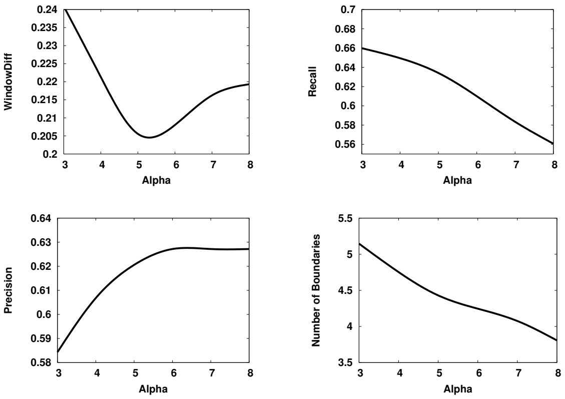

Figure 3: Results of experiments on α

Concerning the value of the α parameter used in the aggregation function in order to extract a good individual from $\bar{P}$, we have to remark that the first objective function C favorizes individuals that generate a lot of boundaries, small segments having more chances to be cohesive. At the contrary, the second objective function D is in favor of individuals having few boundaries, since long segments have more chances to own sentences faraway the following. The parameter α has thus an influence on the number of detected boundaries (figure 3). The best results according to WindowDiff are obtained around $\alpha = 5$. Nevertheless, an higher value enables to obtain a better precision and a lower a better recall. This value may be set between 4 and 6 w.r.t. the frequency of the required segmentation.

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. SegGen has been compared to the following methods to evaluate its efficiency:

- **TT**: TextTiling[4] [Hearst, 1997] is based on the distribution of the terms in the text, according to several criteria (number of common words, new words, active chains).
- **C99**: C99[5] [Choi, 2000], which seems to be the most efficient existing approach, uses a ”local ranking” by determining, for each pair of textual units, the number of surrounding pairs having a lower similarity.
- **CS**: ClassStruggle, mentioned in section 2, is based on a clustering process to segment the texts.

As mentioned in section 3.2, we tried to improve the initial population by inserting a good solution. Therefore, two versions of the algorithm, $\mathbf{S}_5$ and $\mathbf{SC}_5$, have been evaluated. In $\mathbf{SC}_5$, a segmentation scheme, computed by $C99$, has been inserted in the initial population. Both use a coefficient $\alpha = 5$ (see section 4.2).

Since SegGen uses stochastic parameters in mutation and crossover, the range of error has been evaluated by computing the standard deviations, on ten executions, of the average of scores obtained on each corpus. Standard deviation of WindowDiff is located around 0.015, of recall around 0.01 and of precision around 0.012. These values are really low w.r.t. the ranges of scores given in table 1.

5.2 Results

According to results given in table 1, ClassStruggle, having a more complete view of the relations of the sentences thanks to the use of a clustering process, is globally more efficient than C99 and TextTiling. However, its incremental process reduces its ability to take into account the segment cohesion and the similarity between adjacent segments. SegGen overrides this problem and obtains really better results for all accuracy criteria, especially on corpuses that do not have a lot of boundaries to retrieve (i.e. $T(50, 2)$ and $T(100, 4)$). TextTiling and C99 seem to have difficulties to adapt themselves w.r.t the number of boundaries to retrieve, the length of the text has a great impact on their number of detected boundaries. SegGen seems to better follow these variations.

However, on corpora that have a lot of boundaries to retrieve, i.e. $T(50, 4)$ and $T(100, 8)$, SegGen seems to have difficulties to product individuals sufficiently segmented. This is due to the fact that good individuals with a lot of boundaries are more difficult to find than others. The insertion of a good individual in the initial population (SC$_5$), obtained by C99, seems to solve this problem. C99, providing segmentations with a lot of boundaries, helps SegGen to find solutions when the search space is huge.

Additionally, a Student t-test[6] has shown that these differences are really significant with a 99% confidence rate, all of the values being higher than the p-value 2.57.

[4] Available on: www.sims.berkeley.edu/˜hearst/.

[5] Available on: www.freddychoi.me.uk.

[6] 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.

WinDiff	T(50,2)		T(50,4)		T(100,4)		T(100,8)	
	μ	σ	μ	σ	μ	σ	μ	σ
TT	**1.02**	**0.71**	**0.40**	**0.26**	**0.97**	**0.66**	**0.37**	**0.16**
C99	**0.89**	**0.75**	**0.38**	**0.31**	**0.54**	**0.45**	**0.33**	**0.16**
CS	**0.36**	**0.42**	**0.27**	**0.20**	**0.33**	**0.29**	**0.28**	**0.13**
S_5	**0.22**	**0.36**	**0.19**	**0.17**	**0.20**	**0.22**	**0.22**	**0.12**
SC_5	**0.22**	**0.37**	**0.18**	**0.18**	**0.20**	**0.23**	**0.18**	**0.11**

Recall	T(50,2)		T(50,4)		T(100,4)		T(100,8)	
	μ	σ	μ	σ	μ	σ	μ	σ
TT	**0.54**	**0.38**	**0.49**	**0.26**	**0.59**	**0.25**	**0.56**	**0.18**
C99	**0.63**	**0.38**	**0.59**	**0.27**	**0.64**	**0.26**	**0.59**	**0.20**
CS	**0.62**	**0.38**	**0.58**	**0.30**	**0.60**	**0.27**	**0.56**	**0.22**
S_5	**0.69**	**0.37**	**0.60**	**0.27**	**0.61**	**0.27**	**0.56**	**0.20**
SC_5	**0.69**	**0.37**	**0.64**	**0.27**	**0.64**	**0.27**	**0.65**	**0.21**

Prec.	T(50,2)		T(50,4)		T(100,4)		T(100,8)	
	μ	σ	μ	σ	μ	σ	μ	σ
TT	**0.26**	**0.22**	**0.49**	**0.26**	**0.27**	**0.14**	**0.47**	**0.17**
C99	**0.32**	**0.22**	**0.48**	**0.23**	**0.40**	**0.19**	**0.53**	**0.17**
CS	**0.54**	**0.36**	**0.60**	**0.29**	**0.52**	**0.26**	**0.57**	**0.21**
S_5	**0.61**	**0.39**	**0.68**	**0.28**	**0.62**	**0.29**	**0.67**	**0.21**
SC_5	**0.61**	**0.39**	**0.68**	**0.28**	**0.63**	**0.29**	**0.69**	**0.19**

Nb.	T(50,2)		T(50,4)		T(100,4)		T(100,8)	
	μ	σ	μ	σ	μ	σ	μ	σ
TT	**4.21**	**1.39**	**4.31**	**1.40**	**9.52**	**2.33**	**9.82**	**2.03**
C99	**4.22**	**1.42**	**4.94**	**1.31**	**6.81**	**2.09**	**9.15**	**2.23**
CS	**2.62**	**1.40**	**3.91**	**1.54**	**5.23**	**2.29**	**7.86**	**2.45**
S_5	**2.44**	**1.32**	**3.49**	**1.40**	**4.19**	**1.87**	**6.50**	**1.80**
SC_5	**2.47**	**1.36**	**3.74**	**1.51**	**4.48**	**2.10**	**7.75**	**2.11**

Table 1: Averages (μ) and Standard deviations between documents (σ) of WindowDiff (WinDiff), recall, precision (Prec.) and number of detected boundaries (Nb.) for each tested method on each corpus.

6 Conclusion

The experiments have shown that our approach appears to realize a really more accurate segmentation of the texts than existing incremental methods. Moreover, SegGen seems to adapt itself much better than other methods according to the number of boundaries to retrieve.

Nevertheless, the fitness function can be improved, in particular by a more sophisticated calculus of similarity, overpassing the sole repetitions of words. Finally, SegGen will be at term integrated as a preliminary step in an ongoing project of information retrieval that aims to furnish composite documents as responses to user's queries.

Acknowledgments

This work was supported by ”Angers Loire Metropole”.

References

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

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

[Choi, 2000] F.Y.Y. Choi. Advances in domain independent linear text segmentation. In *Proc. of the ACL*, pages 26–33, 2000. Morgan Kaufmann Publishers Inc.

[Galley *et al.*, 2003] M. Galley, K. McKeown, E. Fosler-Lussier, and H. Jing. Discourse segmentation of multi-party conversation. In *ACL '03:*, pages 562–569, 2003.

[Goldberg, 1989] D.E. Goldberg. *Genetic Algorithms in Search Optimization and Machine Learning*. Addison-Wesley, Reading, MA, USA, 1989.

[Harman, 1993] D. Harman. Overview of the first trec conference. In *SIGIR '93*, pages 36–47, 1993. ACM Press.

[Hearst, 1997] M.A. Hearst. Texttiling: segmenting text into multi-paragraph subtopic passages. *Comput. Linguist.*, 23(1):33–64, 1997.

[Holland, 1975] J.H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, USA, 1975.

[Kan *et al.*, 1998] M. Kan, J. Klavans, and K. McKeown. Linear segmentation and segment significance. In *6th Workshop on Very Large Corpora (WVLC-98)*, pages 197–205. ACL SIGDAT, 1998.

[Kozima, 1993] H. Kozima. Text segmentation based on similarity between words. In *ACL*, pages 286–288, 1993.

[McDonald and Chen, 2002] D. McDonald and H. Chen. Using sentence-selection heuristics to rank text segments in txtractor. In *JCDL '02*, pages 28–35, 2002. ACM Press.

[Morris and Hirst, 1991] J. Morris and G. Hirst. Lexical cohesion computed by thesaural relations as an indicator of the structure of text. *Comp. Ling.*, 17(1):21–48, 1991.

[Pevzner and Hearst, 2002] L. Pevzner and M.A. Hearst. A critique and improvement of an evaluation metric for text segmentation. *Comp. Ling.*, 28(1):19–36, 2002.

[Raghavan and Agarwal, 1987] V. Raghavan and B. Agarwal. Optimal determination of user-oriented clusters: an application for the reproductive plan. In *Proc. of GA*, pages 241–246, New York, NY, USA, 1987. ACM Press.

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

[Salton *et al.*, 1996] G. Salton, A. Singhal, C. Buckley, and M. Mitra. Automatic text decomposition using text segments and text themes. In *Hypertext '96*, pages 53–65. ACM, 1996.

[Utiyama and Isahara, 2001] M. Utiyama and H. Isahara. A statistical model for domain-independent text segmentation. In *ACL*, pages 491–498, 2001.

[Zitzler, 1999] E. Zitzler. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. Phd thesis, Swiss Federal Institute of Technology (ETH) Zurich, December 1999.