Text Segmentation

by Product Partition Models and Dynamic Programming

Ath. Kehagias[*], A. Nicolaou[†], V. Petridis[‡] and P. Fragkou[‡]

October 30, 2002

Abstract

In this paper we use Barry and Hartigan's *Product Partition Models* to formulate text segmentation as an optimization problem, which we solve by a fast dynamic programming algorithm. We test the algorithm on Choi's segmentation benchmark and achieve the best segmentation results so far reported in the literature.

Keywords: Text Segmentation, Dynamic Programming, Product Partition Models.

1 Introduction

Text segmentation is a problem of great practical significance. The goal is to divide a text into *homogeneous segments*, so that each segment deals with a particular subject while contiguous segments deal with different subjects. In this manner documents relevant to a query can be retrieved from a large database of unformatted (or loosely formatted) text. For an overview of the problem and various methods for its solution see [3, 4, 7, 8, 12, 13, 15, 18, 19, 20].

[*]Department of Math., Phys. and Comp. Sciences, Electrical and Computer Engineering, Faculty of Engineering, Aristotle University of Thessaloniki, Greece.

[†]Department of Business Administration, University of Macedonia, Thessaloniki, Greece.

[‡]Department of Electrical and Computer Engineering, Faculty of Engineering, Aristotle University of Thessaloniki, Greece.

A *Product Partition Model (PPM)* is a Bayesian inference procedure for segmentation of a sequence of random variables, based on the *heterogeneity* of the sequence. PPM's were introduced by Barry and Hartigan [1, 2] (see also [9, 14]) to identify multiple *change points* in the mean and variance of a sequence of normally distributed random variables. The model assumes that the random segmentation produced by the change points has a probability distribution proportional to a product of *prior cohesions*, one for each segment. Given the observations, a new product partition model holds, with *posterior* cohesions for the segments.

In this paper we use the PPM framework to identify text segments. To this end we obtain the posterior joint probability of an observed text and its segmentation as a product of two terms: (a) the probability of the segmentation, described by appropriate prior cohesions, and (b) the conditional (given the segmentation) probability of the *sentence similarity matrix,* described by an appropriate homogeneity function. Note that we use PPM's to assign probabilities to *two-dimensional* structures (the sentence similarity matrices) rather than to one-dimensional sequences. The negative logarithm of the joint probability is the *segmentation cost*, which is minimized by a fast dynamic programming algorithm (compare to the use of computationally demanding Markov Chain Monte Carlo algorithms in [1, 2]). The homogeneity function we use depends on some parameters which are estimated from training data; as far as we know, this approach has not been previously used in conjunction with PPM's.

In Section 2 we describe the PPM's we use to tackle the text segmentation problem; in Section 3 we present a dynamic programming algorithm to solve the problem; in Section 4 we present some experiments to evaluate our algorithm; finally, in Section 5 we discuss our results, review some work related to our own and present future research directions.

2 Problem Formulation

2.1 Representation

Consider a text with T sentences. A *segmentation* of the text is a partition of $\{1, 2, ..., T\}$ into K contiguous *segments*: $\{1, 2, ..., t_1\}$, $\{t_1 + 1, t_1 + 2, ..., t_2\}$, ..., $\{t_{K-1} + 1, t_{K-1} + 2, ..., T\}$; and $t_0, t_1, ..., t_K$ are the *segment boundaries*[1] which satisfy:

$$0 = t_0 < t_1 < ... < t_{K-1} < t_K = T.$$

A concise representation of the segmentation is given by the vector $\mathbf{t} = (t_0, t_1, ..., t_K)$; note that vector length K (i.e. the number of segments) is variable but it satisfies $K \leq T$. We will denote the set of all possible segmentations of $\{1, 2, ..., T\}$ by Φ_T.

Assume that the text has a vocabulary of L distinct words (common *uninformative* words such as "and", "the" etc. are not included). The text can be represented by a $T \times L$ matrix $\mathbf{c}$ where (for $t = 1, 2, ..., T$ and $l = 1, 2, ..., L$):

$$c_{t,l} = \begin{cases} 1 & \text{iff the } l\text{-th word appears in the } t\text{-th sentence;} \\ 0 & \text{else.} \end{cases}$$

The *sentence similarity matrix* of the text is a $T \times T$ matrix $\mathbf{d}$ where:

$$d_{t,t} = 0 \text{ for } 1 \leq t \leq T \text{ and } d_{s,t} = \begin{cases} 1 & \text{if} \quad \sum_{l=1}^{L} c_{s,l} c_{t,l} > 0; \\ 0 & \text{if} \quad \sum_{l=1}^{L} c_{s,l} c_{t,l} = 0. \end{cases} \qquad \text{for } 1 \leq s \neq t \leq T.$$

In other words $d_{s,t} = 1$ when the s-th and t-th sentence have at least one word in common. We will denote the set of all possible sentence similarity matrices (of dimension $T \times T$) by Ψ_T. The resulting

[1]We assume that segment boundaries always appear at the ends of sentences.

matrix $\mathbf{d}$ has 0's and 1's arranged in a characteristic pattern which corresponds to the structure of the text. In Figure 1 we give a *dotplot* [7, 8, 18] of a matrix $\mathbf{d}$ corresponding to a 101-sentence text (the diagonal elements have been set to 0). Ones are plotted as black squares and zeros as white squares.

Figure 1 to appear here

In the following we will use the notation $\mathbf{d}(s,t)$ to denote the square submatrix of $\mathbf{d}$ defined by positions $(s+1,s+1)$ and (t,t). For every $s=0,1,...,T-1$ and $t=1,2,...,T$ we obtain a submatrix $\mathbf{d}(s,t)$, which corresponds to the segment $\{s+1,s+2,...,t\}$. If it is assumed that sentences belonging to the same segment will have many words in common, then submatrices which correspond to actual segments must contain many 1's. Indeed, in Figure 1 we see several "high-density" regions which we expect to correspond to actual segments. Hence a "good" segmentation should maximize the *density* of 1's in the submatrices of $\mathbf{d}$ which correspond to actual segments.

2.2 Product Partition Models

In the rest of the discussion we will consider T, the number of sentences, to be fixed. Let us define two random variables. The *segmentation variable* $\mathbf{T}$ takes values in Φ_T and the *sentence similarity variable* $\mathbf{D}$ takes values in Ψ_T. Each of these variables is specified by its (discrete) probability function. In what follows, probability functions will be denoted by the letter f (with appropriate subscripts). For example

$$f_{\mathbf{T}}(t_0,t_1,...,t_K) = \Pr(\mathbf{T} = (t_0,t_1,...,t_K)),$$

$$f_{\mathbf{D}|\mathbf{T}}(\mathbf{d}|t_0,t_1,...,t_K) = \Pr(\mathbf{D}=\mathbf{d}|\mathbf{T} = (t_0,t_1,...,t_K)).$$

In [1, 2] PPM's are introduced to describe probabilistically the generation of inhomogeneous time series. The same formalism can be applied (with a few modifications) to text segmentation. Modifying the definition of [1, 2] we define a PPM to be a pair of random variables $(\mathbf{D}, \mathbf{T})$ which have a particular type of joint probability distribution $f_{\mathbf{D},\mathbf{T}}$. In particular, a pair $(\mathbf{D}, \mathbf{T})$ is a PPM if the following conditions are satisfied.

1. The probability of a particular segmentation $(t_0, t_1, ..., t_K)$ has the form:

$$f_{\mathbf{T}}(t_0, t_1, ..., t_K) = G_1 \cdot c(t_0, t_1) \cdot c(t_1, t_2) \cdot ... \cdot c(t_{K-1}, t_K) \tag{1}$$

   where the *cohesion* function $c(s,t)$ (associated with the segment $\{s+1, s+2, ..., T\}$) is defined for all integers $s, t \in \{1, 2, ..., T\}$. G_1 is a normalizing constant and K in (1) can take any value between 1 and T.[2]

2. *Conditional* on $\mathbf{T} = (t_0, t_1, ..., t_K)$, the probability density of the submatrix $\mathbf{D}(t_{k-1}, t_k)$, has the form (for $k = 1, 2, ..., K$):

$$f_{\mathbf{D}(t_{k-1},t_k)|\mathbf{T}}(\mathbf{d}(t_{k-1}, t_k)|t_0, t_1, ..., t_K) = G_2(t_{k-1}, t_k) \cdot g\left(\mathbf{d}(t_{k-1}, t_k)\right) \tag{2}$$

   where $g(\cdot)$ is a *homogeneity* function and $G_2(t_{k-1}, t_k)$ is a normalizing constant.

Then the joint probability of $\mathbf{D}$ and $\mathbf{T}$ has the form

$$f_{\mathbf{D},\mathbf{T}}(\mathbf{d}, t_0, t_1, ..., t_K) = G_1 \cdot \prod_{k=1}^{K} \left[c(t_{k-1}, t_k) \cdot G_2(t_{k-1}, t_k) \cdot g(\mathbf{d}(t_{k-1}, t_k))\right]. \tag{3}$$

It can be seen that a PPM is characterized by the *cohesion function* (which assigns probabilities to segmentations $(t_0, t_1, ..., t_K)$, independently of sentence similarities $\mathbf{d}$) and the *homogeneity function* (which assigns a probability to each segment, *given* a segmentation).

[2]Note that (1) does *not* imply that the segment lengths $t_2 - t_1$, $t_3 - t_2$, ..., $t_K - t_{K-1}$ are independent.

We now present the specific forms of cohesion and homogeneity which we use in this paper. For cohesion we use (with $0 \leq s < t \leq T$)

$$c(s,t) = \exp\left[-\gamma \cdot \left(\frac{t-s-\mu}{\sqrt{2}\sigma}\right)^2\right] \tag{4}$$

(where μ, σ, γ are tuning parameters). For homogeneity we use

$$g(\mathbf{d}(s,t)) = \exp\left[(1-\gamma) \cdot \left(\frac{\sum_{i=s+1}^{t}\sum_{j=s+1}^{t} d_{i,j}}{(t-s)^r}\right)\right] \tag{5}$$

(where r, γ are parameters). The significance of the above functions (and the associated parameters) is as follows.

1. The cohesion of (4) is used to incorporate some information regarding segment *length* (for example, if segmented *training data* are available, they may be used to estimate the mean value μ and standard deviation σ of segment length).

2. The homogeneity of (5) assigns high probability to segments with large $\left(\sum_{i=s+1}^{t}\sum_{j=s+1}^{t} d_{i,j}\right)/(t-s)^r$ values. The term $\sum_{i=s+1}^{t}\sum_{j=s+1}^{t} d_{i,j}$ is the total number of ones in the submatrix of $\mathbf{d}(s,t)$ which corresponds to the segment $\{s+1, s+2, .., t\}$. When the parameter r is equal to 2 then $(t-s)^r$ is the "area" of the corresponding submatrix. Hence, when $r = 2$, the term $\left(\sum_{i=s+1}^{t}\sum_{j=s+1}^{t} d_{i,j}\right)/(t-s)^r$ corresponds to "segment density" (of 1's). When $r \neq 2$ we obtain a "generalized density". Irrespective of the exact value of r, large values of $\left(\sum_{i=s+1}^{t}\sum_{j=s+1}^{t} d_{i,j}\right)/(t-s)^r$ indicate strong intra-segment similarity (as measured by the number of words which are common between sentences belonging to the segment).

3. Finally, the parameter γ is used to control the relative importance of cohesion and homogeneity.

Many other choices of cohesion and homogeneity functions are possible but they will not be discussed in this paper. With the choices indicated above, we define the *segmentation cost* to be the negative logarithm of the joint probability:

$$J(\mathbf{t};\mu,\sigma,r,\gamma)=\sum_{k=1}^{K}\left(\gamma\cdot\frac{(t_k-t_{k-1}-\mu)^2}{2\cdot\sigma^2}-\log G_2(t_{k-1},t_k)-(1-\gamma)\cdot\frac{\sum_{t=t_{k-1}+1}^{t_k}\sum_{t=t_{k-1}+1}^{t_k}d_{s,t}}{(t_k-t_{k-1})^r}\right). \tag{6}$$

Hence the total segmentation cost is the sum of the costs of the K segments; the cost of the k-th segment is the sum of two terms[3], both of which depend *only on data from the k-th segment.* Note that K in (6) is variable; but it does not appear as an argument of J because it is determined by $\mathbf{t}$.

A "good" segmentation vector $\mathbf{t}$ gives segments with high density and small deviation from average segment length and hence the corresponding $J(\mathbf{t};\mu,\sigma,r,\gamma)$ takes a small value[4]. The *optimal* segmentation $\widehat{\mathbf{t}}$ gives the *global* minimum of $J(\mathbf{t};\mu,\sigma,r,\gamma)$

$$\widehat{\mathbf{t}}=\arg\min_{\mathbf{t}=(t_0,t_1,\dots,t_K)\in\Phi_T}J(\mathbf{t};\mu,\sigma,r,\gamma). \tag{7}$$

Note that $\widehat{\mathbf{t}}$ specifies both the optimal number of segments K and the optimal positions of the segment boundaries $\widehat{t}_0,\widehat{t}_1,...,\widehat{t}_K$.

Before concluding this section, let us emphasize that, despite the formal similarity of (4) with the normal distribution, segment length is *not* a Gaussian random variable. The key point is that (4) defines a probability distribution on *segmentations,* not on *segment lengths.* [5]

[3]The term $\log G_2(t_{k-1},t_k)$ can be subsumed in the cohesion part of the cost.

[4]Small in the *algebraic* sense; note that $J(\mathbf{t};\mu,\sigma,r,\gamma)$ can take both positive and negative values.

[5]We *can* obtain the probability distribution of segment length from (4) by appropriate summations over segmentations; the resulting probability will be *conditioned* on the total segment length being equal to the (given number) T. If one performs this (tedious) calculation it will be seen that segment length does *not* have a normal distribution.

3 The Segmentation Algorithm

Hence the problem of text segmentation has been reduced to the minimization of segmentation cost $J(\mathbf{t}; \mu, \sigma, r, \gamma)$ as given by (6). We now present a dynamic programming algorithm which, given the sentence similarity matrix $\mathbf{d}$, and the parameters μ, σ, r, γ computes the optimal segmentation $\widehat{\mathbf{t}}$ (the choice of values for μ, σ, r, γ will be discussed in Section 4).

Text Segmentation Algorithm

```
**Input:** The sentence similarity matrix **d;** the parameters $\mu, \sigma, r, \gamma$ .
**Minimization**
$C_0 = 0$
$z_0 = 0$
For $t = 1, 2, ..., T$
    $C_t = \infty$
    For $s = 0, 1, ..., t-1$
        If $C_s + \gamma \cdot \frac{(t-s-\mu)^2}{2\sigma^2} - \log G_2(s,t) - (1-\gamma) \cdot \frac{\sum_{i=s+1}^{t} \sum_{j=t_{k-1}+1}^{t_k} d_{i,j}}{(t-s)^r} \leq C_t$
            $C_t = C_s + \gamma \cdot \frac{(t-s-\mu)^2}{2\sigma^2} - \log G_2(s,t) - (1-\gamma) \cdot \frac{\sum_{i=s+1}^{t} \sum_{j=t_{k-1}+1}^{t_k} d_{i,j}}{(t-s)^r}$
            $z_t = s$
        End
    End
End
**Backtracking**
$K = 0$
$s_K = T$
```

```
While $z_{s_K} > 0$
    $K = K + 1$
    $s_K = z_{s_{K-1}}$
End
$K = K + 1$
$s_K = 0$
For $k = 0, 1, ..., K$
    $\widehat{t}_k = s_{K-k}$
End
```

Output: The optimal segmentation $\widehat{\mathbf{t}} = (\widehat{t}_0, \widehat{t}_1, ..., \widehat{t}_K)$..

It can be shown easily (by standard dynamic programming arguments [6]) that the minimization phase computes the *global* minimimum of (6). The actual $\widehat{\mathbf{t}}$ which minimizes segmentation cost is obtained in the backtracking phase. The algorithm runs in O(T^2) time[6].

4 Experiments

In this section we evaluate our algorithm using a text collection which has been first presented by Choi in [7]. Several researchers have used this collection as a benchmark for text segmentation algorithms [7, 8, 19]. Choi's collection consists of 700 texts, each text being a concatenation of ten text segments. Each segment consists of "the first n sentences of a randomly selected document from the *Brown Corpus* [11]. (News articles ca**.pos and the informative text cj**.pos)"[7]. The 700 texts can be divided into

[6]A form of *pruning* can be employed by choosing some integer S_{low} and restricting the inner loop to run for $s = T - S_{low} + 1$, $T - S_{low} + 2$, ..., T; in this case the algorithm runs in O(T) time.

[7]It follows that segment boundaries will always appear at the end of sentences.

four datasets: Set0 has 400 texts with n in the range 3-11, Set1 has 100 texts with n in the range 3-5, Set2 has 100 texts with n in the range 6-8 and Set3 has 100 texts with n in the range 9-11. We preprocess the texts by removing punctuation marks and stop-words (determined by a *stoplist*) and stemming the remaining words by Porter's algorithm [16].

In the experiments reported below we measure segmentation accuracy using Beeferman's P_k metric [3, 4] which is used widely for text segmentation tasks. P_k takes values between 0% and 100%, with high values indicating *low* segmentation accuracy. More details on P_k can be found in [3, 4] .

We use some of Choi's text as *training data* to determine appropriate μ, σ, γ and r values by a *parameter validation* procedure. Then we evaluate our algorithm on (previously unused) *test data*. More specifically, we perform the following procedure for each of the datasets Set0, Set1, Set2, Set3.

1. We choose randomly half of the texts in the dataset to be used as training texts; the rest of the samples are set aside to be used as test texts.

2. We determine μ and σ values using the training texts and standard statistical estimators.

3. We determine appropriate γ and r values by running the segmentation algorithm on all the training texts with the 80 possible combinations of $\gamma \in$\{0.00, 0.01, 0.02, ..., 0.09, 0.1, 0.2, 0.3, ..., 1.0\} and $r \in$\{0.33, 0.50, 0.66, 1.00\} values; the optimal (γ, r) combination is the one which yields the minimum P_k value[8].

4. We apply the algorithm to the test texts using previously estimated μ, σ, γ and r values.

The above procedure is repeated five times for each of the four datasets and the resulting values of P_k are averaged. The results are considerably better than any previously reported on Choi's dataset. In Table 1 we list the P_k values achieved by our algorithm to the ones obtained by various other

[8]In this and the next step we omit computation of the term $G_2(s,t)$ from our algorithm. This simplifies the algorithm and, we have found, does not degrade performance.

segmentation algorithms operating on Choi's dataset [7, 8, 19]. The first row lists the algorithm used, the second row lists the publication in which the result appears; the next four rows list P_k for Set0, Set1, Set2, Set3; the final row lists the P_k value averaged over all samples. The results of our segmentation algorithm appear in the last column. It can be seen that our algorithm performs considerably better than all the remaining ones. Let us note that the best performance has been achieved for γ in the range [0.08, 0.4] and for r equal to either 0.5 or 0.66.

Method	CWM1	CWM2	CWM3	C99b	C99	C99b,-r	U00b	U00	Our Algo.
Paper	[8]	[8]	[8]	[7]	[7]	[7]	[19]	[19]	
Set0	9.00%	14.00%	12.00%	12.00%	13.00%	23.00%	10.00%	11.00%	7.00%
Set1	10.00%	10.00%	10.00%	11.00%	18.00%	19.00%	9.00%	13.00%	5.45%
Set2	7.00%	11.00%	9.00%	10.00%	10.00%	21.00%	7.00%	6.00%	3.00%
Set3	5.00%	12.00%	8.00%	9.00%	10.00%	20.00%	5.00%	6.00%	1.33%
All Sets	8.00%	13.00%	11.00%	11.00%	13.00%	22.00%	9.00%	10.00%	5.39%

Table 1

In Table 2 we give execution times (for segmenting a single text) of our algorithm and some of the algorithms of [7, 8, 19]. Our algorithm was executed on a Pentium III 600Mhz computer with 256Mbyte RAM[9].

Algorithm	U00b	U00	C99b	C99	Our Algo
Avg Exec. Time in sec	1.37	1.36	1.45	1.49	0.91

Table 2

[9]It is possible that the remaining algorithms of Table 1 were executed on slower machines

5 Discussion

Using a variation of Product Partition Models we have formulated text segmentation as an optimization problem which we have solved by a fast dynamic programming algorithm. On Choi's segmentation benchmark we achieve the best results so far reported in the literature.

Our approach has several features which have not been previously combined. Several authors [13, 15, 18, 19] have used dynamic programming for text segmentation; most notably our algorithm is very similar to the one used by Heinonen [13]. However, Heinonen uses a different homogeneity function which takes into account only the similarity of *adjacent sentences*; our function takes into account the homogeneity of *all sentence pairs* within a segment. Also note that, as far as we know, PPM's have not been previously applied to two-dimensional structures (such as the sentence similarity matrix); the use of training data to estimate the parameters of the homogeneity function also appears to be new.

It is interesting to compare PPM's to Hidden Markov Models (HMM's), which have been widely applied to time series segmentation tasks [5, 17]. While our segmentation algorithm is related to the Viterbi algorithm (used for HMM-based segmentation), there is an important difference between HMM's and PPM's. Namely, in the HMM framework each *observation* within a segment depends only on the current *state*; in the PPM framework one specifies a probability distribution for the entire set of observations within a segment. In this sense, the PPM appears as a more natural model of text generation.

In this paper we have assumed that segmented training data are available to train the algorithm parameters. If this is not the case, a useful approach for parameter training can be adopted from the HMM literature. Namely, one can use an EM-style algorithm [10], alternating segmentation and parameter estimation steps. We are currently investigating the use of this approach within the PPM framework; our results will be reported in a future publication.

In this paper we have examined an *offline* segmentation problem. In other words, we assume that the entire text is available to be processed by the segmentation algorithm. An *online* problem would require the segmentation of a continuous incoming text-stream. The algorithm we have presented here cannot solve this type of problem. However, it is not too difficult to adapt a dynamic programming algorithm for online operation; hence in the future we plan to investigate online versions of our segmentation algorithm.

References

[1] D. Barry and J. A. Hartigan. "Product partition models for change point problems". *Ann. of Stat.*, vol.20, pp.260-279, 1992.

[2] D. Barry and J. A. Hartigan. "A Bayesian analysis for change point problems". *JASA*, vol.88, pp.309-319, 1993.

[3] D. Beeferman, A. Berger and J. Lafferty. "Text segmentation using exponential models". In *Proceedings of the 2nd Conference on Empirical Methods in Natural Language Processing*, pp. 35-46, 1997.

[4] D. Beeferman, A. Berger and J. Lafferty. "Statistical models for text segmentation". *Machine Learning*, vol. 34, pp.177-210, 1999.

[5] Y. Bengio. "Markovian models for sequential data". *Neural Comp. Surveys*,vol.2,pp.129-162, 1998.

[6] D. Bertsekas. *Dynamic Programming: Deterministic and Stochastic Models.* Prentice Hall, 1987.

[7] F.Y.Y. Choi. "Advances in domain independent linear text segmentation". *Proc. of the North American Chapter of the Ass. for Comp. Linguistics*, pp. 26-33, 2000.

[8] F.Y.Y. Choi, P. Wiemer-Hastings and J. Moore. “Latent semantic analysis for text segmentation”. In *Proceedings of the 6th Conference on Empirical Methods in Natural Language Processing*, pp.109–117, 2001.

[9] E.M. Crowley, “Product partition models for normal means”. *JASA*, vol.92, pp.192-198, 1997.

[10] A.P. Dempster, N.M. Laird and D.B. Rubin. “Maximum likelihood from incomplete data via the EM algorithm”. *J. Roy. Stat. Soc. B.*, vol. 39, pp.1-38, 1977.

[11] W.N. Francis and H. Kucera. *Frequency Analysis of English Usage: Lexicon and Grammar.* Houghton Mifflin, 1982.

[12] M. Hearst. “Multi-paragraph segmentation of expository text”. In *Proc. of the 32nd Annual Meeting of the Association for Computational Linguistics*, Las Cruces, NM, 1994.

[13] O. Heinonnen. “Optimal multi-paragraph text segmentation by dynamic programming”. In *Proceedings of COLING-ACL’98*, 1998.

[14] R.H. Loschi, F. R. B Cruz. “An analysis of the influence of some prior specifications in the identification of change points via product partition model”. *Comput. Statist. Data Anal.*, vol.39, pp.477–501, 2002.

[15] J.M. Ponte and W. B. Croft. “Text segmentation by topic”. In *Proceedings of the 1st European Conference on Research and Advanced Technology for Digital Libraries*, pp.120-129, 1997.

[16] M.F. Porter. “An algorithm for suffix stripping”. *Program*, vol.14, pp.130-137, 1980.

[17] L. R. Rabiner. “A tutorial on hidden Markov models and selected applications in speech recognition”. *Proc. of the IEEE*, vol. 77, pp. 257–285, 1989.

[18] J. C. Reynar. “Statistical models for text segmentation.” In *Proc. of the 37th Annual Meeting of the Association for Computational Linguistics*, Baltimore, MD, 1999.

[19] M. Utiyama and H. Isahara. “A statistical model for domain - independent text segmentation”. In *Proceedings of the 9th Conference of the European Chapter of the Association for Computational Linguistics*, pp.491-498, 2001.

[20] Y. Yaari. “Segmentation of expository texts by hierarchical agglomerative clustering,” in *Proc. of the RANLP'97*, 1997.