A Dynamic Programming Algorithm for the Segmentation of Greek Texts

Pavlina Fragkou

In this paper we introduce a *dynamic programming* algorithm to perform linear text segmentation by global minimization of a segmentation cost function which consists of: (a) within-segment word similarity and (b) prior information about segment length. The evaluation of the segmentation accuracy of the algorithm on a text collection consisting of Greek texts showed that the algorithm achieves high segmentation accuracy and appears to be very innovating and promissing.
Keywords: Text Segmentation, Document Retrieval, Information Retrieval, Machine Learning.

1. Introduction

Text segmentation is an important problem in information retrieval. Its goal is the division of a text into *homogeneous* ('*lexically coherent*') segments, i.e. segments exhibiting the following properties: (a) each segment deals with a particular subject and (b) contiguous segments deal with different subjects. Those segments can be retrieved from a large database of unformatted (or loosely formatted) text as being relevant to a query.

This paper presents a *dynamic programming* algorithm which performs *linear* segmentation [1] by global minimization of a *segmentation cost*. The *segmentation cost* is defined by a function consisting of two factors: (a) *within-segment word similarity* and (b) *prior information about segment length*. Our algorithm has the advantage that it can be applied to either large texts - to segment them into their constituent parts (e.g. to segment an article into sections) - or to a stream of independent, concatenated texts (e.g. to segment a transcript of news into separate stories).

[1] As opposed to *hierarchical* segmentation (Yaari (1997))

For the calculation of the segment *homogeneity* (or alternatively *heterogeneity*) of a text, several segmentation algorithms using a variety of criteria have been proposed in the literature. Some of those use linguistic criteria such as cue phrases, punctuation marks, prosodic features, reference, syntax and lexical attraction (Beeferman et al. 1997, Hirschberg & Litman 1993, Passoneau & Litman 1993). Others, following Halliday and Hasan's theory (Halliday & Hasan 1976), utilize statistical similarity measures such as word cooccurrence. For example the linear discourse segmentation algorithm proposed by Morris and Hirst (Morris & Hirst 1991) is based on *lexical cohesion relations* determined by use of Roget's thesaurus (Roget 1977). In the same direction Kozima's algorithm (Kozima 1993, Kozima & Furugori 1993) computes the semantic similarity between words using a semantic network constructed from a subset of the Longman Dictionary of Contemporary English. Local minima of the similarity scores correspond to the positions of topic boundaries in the text.

Youmans (Youmans 1991) and later Hearst (Hearst & Plaunt 1993, Hearst 1994) focused on the similarity between *adjacent* parts of a text. They used a sliding window of text and plotted the number of first-used words in the window as a function of the window's position within the text. In this plot, segment boundaries correspond to deep valleys followed by sharp upturns. Kan (Kan et al. 1998) expanded the same idea by combining word-usage with visual layout information.

On the other hand, other researchers focused on the similarity between *all* parts of a text. A graphical representation of this similarity is a *dotplot*. Reynar (Reynar 1998; 1999) and Choi (Choi 2000, Choi et al. 2001) used *dotplots* in conjunction with *divisive clustering* (which can be seen as a form of *approximate* and *local* optimization) to perform *linear* text segmentation. A relevant work has been proposed by Yaari (Yaari 1997) who used *divisive / agglomerative clustering* to perform *hierarchical* segmentation. Another approach to clustering performs *exact* and *global* optimization by *dynamic programming*; this was used by Ponte and Croft (Ponte & Croft 1997, Xu & Croft 1996), Heinonen (Heinonen, O. 1998) and Utiyama and Isahara (Utiyama & Isahara 2001).

Finally, other researchers use probabilistic approaches to text segmentation including the use of *Hidden Markov Models* (Yamron et al. 1999, Blei & Moreno 2001). Also Beeferman (Beeferman et al. 1997) calculated the probability distribution on segment boundaries by utilizing word usage statistics, cue words and several other features.

2. *The algorithm*

2.1. *Representation*

Suppose that a text contains T sentences and its vocabulary contains L distinct words (e.g words that are not included in the stop list, otherwise most sentences would be similar to most others). This text can be represented by a $T \times L$ matrix

F defined as follows: for $t = 1, 2, ..., T$ and $l = 1, 2, ..., L$ we set

$$F_{t,l} = \begin{cases} 1 & \text{iff l-th word is in t-th sentence} \\ 0 & \text{else.} \end{cases}$$

The *sentence similarity matrix* D of the text is a $T \times T$ matrix where for $s, t = 1, 2, ..., T$ we set

$$D_{s,t} = \begin{cases} 1 & \text{if} \quad \sum_{l=1}^{L} F_{s,l} F_{t,l} > 0; \\ 0 & \text{if} \quad \sum_{l=1}^{L} F_{s,l} F_{t,l} = 0. \end{cases}$$

This means that $D_{s,t}$ = 1 if the s-th and t-th sentence have at least one word in common. Every part of the original text corresponds to a submatrix of D. It is expected that submatrices which correspond to actual segments will have many sentences with words in common, thus will contain many 'ones'. In Figure 1 we give a *dotplot* of a matrix corrpesponding to a 91-sentences text. 'Ones' are plotted as black squares and 'zeros' as white squares. Further justification for the use of this similarity matrix and graphical representation can be found in Petridis et al. (2001), Kehagias, A et al. (2002), Reynar (1998; 1999), Choi (2000) and Choi et al. (2001).

We make the assumption that segment boundaries always occur at the end of sentences. A segmentation of a text is a partition of the set $\{1, 2, ..., T\}$ into K subsets (i.e. *segments*, where K is a variable number) of the form $\{1, 2, ..., t_1\}$, $\{t_1 + 1, t_1 + 2, ..., t_2\}$, ..., $\{t_{K-1} + 1, t_{K-1} + 2, ..., T\}$ and can be represented by a variable length vector $\mathbf{t} = (t_0, t_1, ..., t_K)$, where $t_0, t_1, ..., t_K$ are the *segment boundaries* corresponding to the last sentence of each subset.

2.2. Dynamic Programming

Dynamic programming as a method guarantees the optimality of the result with respect to the input and the parameters. Following the approach of Heinonen (Heinonen, O. 1998) we use a *dynamic programming* algorithm which decides the locations of the segment boundaries by calculating the globally optimal splitting $\mathbf{t}$ on the basis of a similarity matrix (or a curve), a preferred fragment length and a cost function defined. Given a similarity matrix D and the parameters μ, σ, r and γ (the role of each of which will be described in the sequel) the *dynamic programming* algorithm tries to minimize a *segmentation cost function* $J(\mathbf{t}; \mu, \sigma, r, \gamma)$ with respect to $\mathbf{t}$ (here $\mathbf{t}$ is the independent variable which is actually a vector specifying the boundary position of each segment and the number of segments K while μ, σ, r, γ are parameters) which is defined in equation (1).

$$J(\mathbf{t}; \mu, \sigma, r, \gamma) = \sum_{k=1}^{K} \gamma \cdot \frac{(t_k - t_{k-1} - \mu)^2}{2 \cdot \sigma^2} - (1 - \gamma) \cdot \frac{\sum_{s=t_{k-1}+1}^{t_k} \sum_{t=t_{k-1}+1}^{t_k} D_{s,t}}{(t_k - t_{k-1})^r} \tag{1}$$

Hence the sum of the costs of the K segments constitutes the total segmentation cost; the cost of each segment is the sum of the following two terms (with their relative importance weighted by the parameter γ):

1. The term $\frac{(t_k - t_{k-1} - \mu)^2}{2 \cdot \sigma^2}$ corresponds to the length information measured as the deviation from the average segment length. In this sense, μ and σ can be considered as the mean and standard deviation of segment length measured either on the basis of words or on the basis of sentences appearing in the text's segments and can be estimated from *training data*.

2. The term $\frac{\sum_{s=t_{k-1}+1}^{t_k} \sum_{t=t_{k-1}+1}^{t_k} D_{s,t}}{(t_k - t_{k-1})^r}$ corresponds to (word) similarity between sentences. The numerator of this term is the total number of 'ones' in the D submatrix corresponding to the k-th segment. In the case where the parameter r is equal to 2, $(t_k - t_{k-1})^r$ corresponds to the area of submatrix and the above fraction corresponds to 'segment density'. A 'generalized density' is obtained when $r \neq 2$ and enables us to control the degree of influence of the surface with regard to the 'information' (i.e. the number of 'ones') included in it. Strong intra-segment similarity (as measured by the number of words which are common between sentences belonging to the segment) is indicated by large values of $\frac{\sum_{s=t_{k-1}+1}^{t_k} \sum_{t=t_{k-1}+1}^{t_k} D_{s,t}}{(t_k - t_{k-1})^r}$, irrespective of the exact value of r.

Segments with high density and small deviation from average segment length (i.e. a small value of the corresponding $J(\mathbf{t}; \mu, \sigma, r, \gamma)$ [2]) provide a 'good' segmentation vector $\mathbf{t}$. The *global* minimum of $J(\mathbf{t}; \mu, \sigma, r, \gamma)$ provides the *optimal* segmentation $\widehat{\mathbf{t}}$. It is worth mentioning that the optimal $\widehat{\mathbf{t}}$ specifies both the optimal number of segments K and the optimal positions of the segment boundaries $t_0, t_1, ..., t_K$. In the sequel, our algorithm is presented in a form of pseudocode.

Dynamic Programming Algorithm

Input: The $T \times T$ similarity matrix D; the parameters μ, σ, r, γ.
Initilization
For $t = 1, 2, ..T$
    $Sum = 0$
    For $s = 1, 2, ..., t-1$
        $Sum = Sum + D_{s,t}$
        $S_{s+1,t} = \frac{Sum}{(t-s)^r}$
    End
End
Minimization

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

$C_0 = 0, Z_0 = 0$
For $t = 1, 2, , T$
    $C_t = \infty$
    For $s = 1, 2, ..., t-1$
        If $C_s + S_{s,t} + \frac{(t-s-\mu)^2}{2\sigma^2} \leq C_t$
            $C_t = C_s - (1-\gamma) \cdot S_{s+1,t} + \gamma \cdot \frac{(t-s-\mu)^2}{2\sigma^2}$
            $Z_t = s$
        EndIf
    End
End
BackTracking
$K = 0$, $s_K = T$
While $Z_{s_K} > 0$
        $K = K + 1$
        $s_K = Z_{s_{K-1}}$
End
$K = K + 1, s_K = 0, \widehat{t}_0 = 0$
For $k = 1, 2, ..., K$
    $\widehat{t}_k = s_{K-k}$
End
Output: The optimal segmentation vector $\widehat{\mathbf{t}} = (\widehat{t}_0, \widehat{t}_1, ..., \widehat{t}_K)$.

---

3. *Evaluation*

3.1. *Measures of Segmentation Accuracy*

The performance of our algorithm was evaluated by three indices: *Precision*, *Recall* and Beeferman's P_k *metric*. *Precision* and *Recall* measure segmentation *accuracy*. For the segmentation task, *Precision* is defined as 'the number of the estimated segment boundaries which are actual segment boundaries' divided by 'the number of the estimated segment boundaries'. On the other hand, *Recall* is defined as 'the number of the estimated segment boundaries which are actual segment boundaries' divided by 'the number of the true segment boundaries'. High segmentation accuracy is indicated by high values of *both Precision* and *Recall*. However, these two indices have some shortcomings. First, high *Precision* can be obtained at the expense of low *Recall* and conversely. Additionally, those two indices penalize equally every inaccurately estimated segment boundary whether it is near or far from a true segment boundary.

An alternative measure P_k which overcomes the shortcomings of *Precision* and *Recall* and measures segmentation *in*accuracy was introduced recently by Beeferman et al. (Beeferman et al. 1997). Intuitively, P_k measures the proportion of 'sentences which are wrongly predicted to belong to the same segment (while

actually they belong in different segments)' or 'sentences which are wrongly predicted to belong to different segments (while actually they belong to the same segment)'. P_k is a measure of how well the true and hypothetical segmentations agree (with a low value of P_k indicating high accuracy (Beeferman et al. 1997)). P_k penalizes near-boundary errors less than far-boundary errors. Hence P_k evaluates segmentation accuracy more accurately than *Precision* and *Recall*.

3.2. Experiments

For the experiments, we use a text collection compiled from a corpus comprising of text downloaded from the website `http://tovima.dolnet.gr` of the newspaper entitled 'To Vima'. This newspaper contains articles belonging to one of the following categories: 1) Editorial, diaries, reportage, politics, international affairs, sport reviews 2) cultural supplement 3) Review magazine 4) Business, finance 5) Personal Finance 6) Issue of the week 7) Book review supplement 8) Art review supplement 9) Travel supplement. Stamatatos et al. (2001) constructed a corpus collecting texts from supplement no. 2) which includes essays on science, culture, history etc. They selected 10 authors from the above set without taking any special criteria into account. Then 30 texts of each author were downloaded from the website of the newspaper as shown in the table below:

Author	**Thematic Area**
Alachiotis	Biology
Babiniotis	Linguistics
Dertilis	History,Society
Kiosse	Archeology
Liakos	History,Society
Maronitis	Culture,Society
Ploritis	Culture,History
Tassios	Technology,Society
Tsukalas	International Affairs
Vokos	Philosophy

Table 1:
List of Authors and Thematic Areas dealt by each of those.

No manual text preprocessing nor text sampling was performed aside from removing unnecessary heading irrelevant to the text itself. All the downloaded texts were taken from the issues published from 1997 till early 1999 in order to minimize the potential change of the personal style of an author over time. Further details can be found in Stamatatos et al. (2001).

The preprocessing of the above texts was made using the morphosyntactic tagger (better known as part-of-speech tagger) developed by G. Orphanos (Orphanos & Christodoulakis 1999, Orphanos & Tsalidis 1999). The aforementioned tagger

is a POS tagger for modern Greek (a high inflectional language) which is based on a Lexicon capable of assigning full morphosyntactic attributes (i.e. POS, Number, Gender, Tense, Voice, Mood and Lemma) to 876.000 Greek word forms. This Lexicon was used to build a tagged corpus capable of identifying the behavior of all POS ambiguity schemes present in Modern Greek (e.g. Pronoun-Clitic-Article, Pronoun-Clitic, Adjective-Adverb, Verb-Noun, etc) as well as the characteristics of unknown words. This corpus was used for the induction of decision trees, which along with the Lexicon are integrated into a robust POS tagger for Modern Greek texts.

The tagger architecture consists of three parts: the Tokenizer, the Lexicon and finally the Disambiguator and Guesser. Raw text passes through the Tokenizer, where it is converted to a stream of tokens. Non-word tokens (e.g. punctuation marks, numbers, dates etc.) are resolved by the Tokenizer and receive a tag corresponding to their category. Word tokens are looked up in the Lexicon and those found receive one or more tags. Words with more than one tags and those not found in the Lexicon pass through the Disambiguator/Guesser, where the contextually appropriate tag is decided.

The Disambiguator/Guesser is a 'forest' of decision trees, one tree for each ambiguity scheme present in Modern Greek and one tree for unknown guessing. When a word with two or more tags appears, its ambiguity scheme is identified. Then, the corresponding decision tree, is selected, which is traversed according to the values of the morphosyntactic features extracted from contextual tags. This traversal returns the contextually appropriate POS along with its corresponding lemma. The ambiguity is resolved by eliminating the tag(s) with different POS than the one returned by the decision tree. The POS of an unknown word is guessed by traversing the decision tree for unknown words, which examines contextual features along with the word ending and capitalization and returns an open class POS and the corresponding lemma.

3.2.1. Preprocessing

For the experiments we use the texts taken from the collection compiled from the corpus of the newspaper 'To Vima'. Each of the 300 texts of the collection of articles compiled from this newspaper is preprocessed using the POS tagger developed by G. Orphanos. More specifically, every word in the text was substituted by its lemma, determined by the tagger. Punctuation marks, numbers and all words were removed except from words that are either nouns, verbs, adjectives or adverbs. For those words that their lemma was not determined by the tagger, due to the fact that those words were not contained in the Lexicon used for the creation of the tagger, no substitution was made and the words were used as they were. The only information that was kept was the end of each sentence appearing in each text. We next present two suites of experiments. The difference between those suites lies in the length of the created segments and the number of authors

used for the creation of the texts to segment, where each text being a concatenation of ten text segments.

3.2.2. First suite of experiments

In the first suite of experiments, our collection consists of 6 datasets: Set0, ... , Set5. The difference between those datasets lies in the number of authors used for the generation of the texts to segment and consequently the number of texts used from the collection. The table below contains the aforementioned information.

Dataset	Authors	No. of texts per dataset
0	Kiosse, Alachiotis	60
1	Kiosse, Maronitis	60
2	Kiosse, Alachiotis, Maronitis	90
3	Kiosse, Alachiotis, Maronitis, Ploritis	120
4	Kiosse, Alachiotis, Maronitis, Ploritis, Vokos	150
5	All Authors	300

Table 2:
List of the datasets compilied in the 1st suite of experiments and the number of the author's texts used for each of those.

For each of the above datasets, we constructed four subsets. The difference between those subsets lies in the range of the sentences appearing in each segment for every text. If a and b correspond to the lower and higher values of sentences consisting each segment, we have used four different pairs: (3,11), (3,5), (6,8) and (9,11). In every dataset, before generating any of the texts to segment, each of which contains 10 segments, we selected the authors, whose texts will be used for this generation. If X is the number of authors contributing to the generation of the dataset, for all datasets, each text is generating according to the following procedure (which guarantees that each text contains ten segments):

For the j-th out of 10 segments (e.g. $j = 1, 2, ..., 10$) of the generated text:

(1) *Select randomly a number $i \in \{1, ..., X\}$ corresponding to an author among those contributing to the generation of the dataset.*

(2) *Select randomly a number $k \in \{1, 2, ..., 30\}$ corresponding to the texts belonging to the i-th author.*

(3) *Select randomly a number $l \in \{a, ..., b\}$ corresponding to the number of consecutive sentences extracted from the $k - th$ text (starting from the first sentence). Those sentences constitutes the generated segment.*

For every subset, using the procedure described above, we generated 50 texts. As it was mentioned before, our algorithm uses four parameters μ, σ, γ and r,

where μ and σ can be interpreted as the average and standard deviation of segment length; it is not immediately obvious how to calculate the optimal values for each of the parameters. A procedure for determining appropriate values of μ, σ, γ and r was introduced using *training data* and a *parameter validation* procedure. Then our algorithm is evaluated on (previously unseen) *test data*. More specifically, for each of the datasets Set0,..., Set5 and each of their subsets we perform the procedure described in the sequel:

1. Half of the texts in the dataset are chosen randomly to be used as training texts; the rest of the samples are set aside to be used as test texts.
2. Appropriate μ and σ values are determined using all the training texts and the standard statistical estimators.
3. Parameter γ is set to take the values 0.00, 0.01, 0.02, ... , 0.09, 0.1, 0.2, 0.3, ... , 1.0 and r to take the values 0.33, 0.5, 0.66,1. This yields 20$\times$4=80 possible combinations of γ and r values. Appropriate γ and r values are determined by running the segmentation algorithm on all the training texts with the 80 possible combinations of γ and r values; the one that yields the minimum P_k value is considered to be the optimal (γ, r) combination.
4. The algorithm is applied to the test texts using previously estimated γ, r, μ and σ values.

An idea of the influence of γ and r on P_k on the first suite of experiments can be observed in Figures 2-5 (corresponding to subsets '3-11', '3-5', '6-8' and '9-11' of Set5). In those figures *Exp 1* refers to the first suite of experiments.

The above procedure is repeated five times for each of the six datasets and the resulting values of *Precision*, *Recall* and P_k are averaged. The performance of our algorithm (as obtained by the validated parameter values) is presented in Table 3.

3.2.3. Second Suite of experiments

In the second suite of experiments we used the same collection of texts compiled from the corpus of the newspaper 'To Vima'. The difference between those two suites lies in the way of generating the texts used for training and for testing. In this suite of experiments, we used all the available (300) texts of the collection of the Greek corpus, which means all the available authors. We constructed a single dataset containing 200 texts. Half of them were used for training while the rest of them were used for testing. Each of the aforementioned texts was generated according to the following procedure (which guarantees that each text contains 10 segments):

For the j-th out of 10 segments (e.g. $j = 1, 2, ..., 10$) of the generated text:

(I) *Select randomly a number* $i \in \{1, ..., 10\}$ *corresponding to an author among the 10 contributing to the generation of the dataset.*

(II) *Select randomly a number* $k \in \{1, 2, ..., 30\}$ *corresponding to the texts belonging to the* i*-th author. The selected text is read and scanned in order to determine the number of paragraphs that consists it. If* Z *is the number of paragraphs that consists it then:*

(III) *Select randomly a number* $l \in \{1, ..., Z\}$ *corresponding to the number of paragraphs appearing in the* $k - th$ *text.*

(IV) *Select randomly a number* $m \in \{1, ..., Z - l\}$ *corresponding to the "starting paragraph". Thus the segment contains all the paragraphs of the* $k - th$ *text starting from paragraph* m *and ending at the paragraph* $m + l$.

1st suite of Experiments	(3,11)	(3,5)	(6,8)	(9,11)
Set0 Precision	70.65%	86.82%	96.44%	93.33%
Set0 Recall	71.11%	87.11%	96.44%	93.33%
Set0 Beeferman's P_k	14.04%	6.20%	0.82%	0.84%
Set1 Precision	63.86%	82.98%	91.11%	94.67%
Set1 Recall	67.11%	83.56%	91.11%	94.67%
Set1 Beeferman's P_k	15.82%	8.47%	2.81%	0.98%
Set2 Precision	71.14%	90%	91.11%	92.44%
Set2 Recall	60.89%	89.78%	91.11%	92.44%
Set2 Beeferman's P_k	14.42%	3.45%	2.15%	1.247%
Set3 Precision	59.99%	84.44%	86.22%	91.11%
Set3 Recall	58.67%	83.56%	86.22%	91.11%
Set3 Beeferman's P_k	17.93%	7.36%	3.28%	1.45%
Set4 Precision	57.99%	85%	88.89%	91.11%
Set4 Recall	51.11%	84.89%	88.89%	91.11%
Set4 Beeferman's P_k	17.38%	6.76%	2.65%	1.39%
Set5 Precision	65.74%	81.56%	89.33%	88.89%
Set5 Recall	61.78%	81.78%	89.33%	88.89%
Set5 Beeferman's P_k	14.54%	6.49%	3.57%	1.86%

Table 3

Exp.Suite 1: The Precision, Recall and Beeferman's P_k metric values for the datasets Set0, Set1, Set2, Set3, Set4 and Set5, using sentences as a unit of segment, obtained by a validation procedure.

From the aforementioned method of generating texts, it is obvious that, the 200 generated texts for segmentation are longer than those generated during the first suite of experiments. Thus the segmentation of such texts consists a more difficult

problem. We used the same validation procedure as before with the same values for the parameters r and γ. The obtained validated results are listed in the table below:

2nd suite of Experiments	
Precision	60.60%
Recall	57.00%
Beeferman's P_k	11.07%

Table 4

Exp.Suite 2: The Precision, Recall and Beeferman's P_k metric values for the unique dataset using paragraphs as a unit of segment obtained by a validation procedure.

4. Discussion

Our algorithm was previously tested on Choi's data collection (Kehagias, A et al. 2003), which contains English texts, achieving significantly better results than the ones previously reported in Choi (2000), Choi et al. (2001) and Utiyama & Isahara (2001). Since the collection used here has not been previously used in the literature for the purpose of text segmentation, we cannot provide a direct comparative assessment. However, the performance obtained is comparable and in most cases better than the corresponding on the Choi's collection, even though, for several cases the problem dealt by our algorithm is more difficult. The difficulty lies in the fact that, the thematic area dealt by several authors is very similar (see Table 1). One of the reasons for the high segmentation accuracy is the robustness of the POS tagger used. We have observed that, in general, the tagger fails to find the tag and lemma of very technical words. The use of them as they appear in the original text, does not have a negative impact on the segmentation accuracy. The robustness of our algorithm is also indicated by the performance obtained at the second suite of experiments where the segment length is bigger and the deviation from the average length is high. Even in that case our algorithm achieved very high results. This is the result of the combination of the following facts: First, the use of the segment length term in the cost function seems to improve segmentation accuracy significantly. Second, the use of 'generalized density' ($r \neq 2$) appears to significantly improve performance. Even though the use of 'true density' ($r = 2$) appears more natural, the best segmentation performance (minimum value of P_k) is achieved for significantly smaller values of r. This performance in most cases is improved when using appropriate values of μ, σ, γ and r derived from training data and parameter validation.

Finally, it is worth mentioning that our approach is 'global' in two respects. First, sentence similarity is computed globally through the use of the D matrix and *dotplot*. Second, this global similarity information is also optimized globally by the use of the *dynamic programming algorithm*. This is in contrast with the local optimization of global information (used by Choi) and global optimization

of local information (used by Heinonen).

It is worth mentioning that, the computational complexity of our algorithm is comparable to that of the other methods (namely $O(T^2)$ where T is the number of sentences). Finally, our algorithm has the advantage of automatically determining the optimal number of segments.

5. *Conclusion*

We have presented a *dynamic programming* algorithm which performs text segmentation by global minimization of a segmentation cost consisting of two terms: within-segment word similarity and prior information about segment length. The performance of our algorithm is quite satisfactory considering that it yields a high segmentation accuracy in a text collection containing Greek texts. In the future we intent to use other measures of sentence similarity. We also plan to apply our algorithm to a wide spectrum of text segmentation tasks. We are interested in the segmentation of non artificial real life texts, texts having a diverse distribution of segment length, long texts, and change-of-topic detection in newsfeeds.

Acknowledgements

The author would like to thank G. Orphanos for kindly letting her use the POS Tagger for the preprocessing of the Greek texts, professor E. Stamatatos for providing the corpus of Greek articles, and professors A. Kehagias and V. Petridis for their undivided guidance and support.

References

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

Blei, D.M. & Moreno, P.J. (2001). Topic segmentation with an aspect hidden Markov model. Tech.Rep. CRL 2001-07, COMPAQ Cambridge Research Lab.

Choi, F.Y.Y. (2000). Advances in domain independent linear text segmentation. *In Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics*, pp. 26-33.

Choi, F.Y.Y., Wiemer-Hastings, P.& Moore, J. (2001). Latent semantic analysis for text segmentation. *In Proceedings of the 6th Conference on Empirical Methods in Natural Language Processing*, pp. 109-117.

Halliday, M. & Hasan, R. (1976). *Cohesion in English*. Longman.

Hearst, M. A. & Plaunt,C. (1993). Subtopic structuring for full-length document access. *In Proceedings of the 16th Annual International of Association of Computer Machinery - Special Interest Group on Information Retrieval (ACM / SIGIR) Conference on Research and Development in Information Retrieval*, pp. 59-68.

Hearst, M. A. (1994). Multi-paragraph segmentation of expository texts. *In Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistic*, pp. 9-16.

Heinonen, O. (1998). Optimal Multi-Paragraph Text Segmentation by Dynamic Programming. *In Proceedings of 17th International Conference on Computational Linguistics*, pp. 1484-1486.

Hirschberg, J. & Litman, D. (1993). Empirical studies on the disambiguation and cue phrases. *Computational Linguistics*,vol.19, pp. 501-530.

Kan, M., Klavans, J.L. & McKeown, K. R. (1998). Linear segmentation and segment significance. *In Proceedings of the 6th International Workshop of Very Large Corpora*, pp. 197-205.

Kehagias, A., Petridis, V., Kaburlasos, V.G. & Fragkou, P. (2002). *A Comparison of Word- and Sense-based Text Categorization Using Several Classification Algorithms*. Journal of Intelligent Information Systems, Kluwer Academic Publishers vol. 21(3), pp. 227-247.

Kehagias, A , Fragkou, P. & Petridis, V. (2003). Linear Text Segmentation using a Dynamic Programming Algorithm. *In Proceedings of the European Association of Computational Linguistics*, Budapest, Hungary, pp. 171-178.

Kozima, H. (1993). Text Segmentation based on similarity between words. *In Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, pp. 286-288.

Kozima, H & Furugori, T. (1993). Similarity between words computed by spreading activation on an English dictionary. *In Proceedings of 6th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 232-239.

Morris, J. & Hirst, G. (1991). *Lexical cohesion computed by thesaural relations as an indicator of the structure of text*. Computational Linguistics, vol.17, pp. 21-42.

Orphanos, G. & Christodoulakis, D. (1999). Part-of-speech Disambiguation and Unknown Word Guessing with Decision Trees, *In Proceedings of EACL'99*, Bergen, Norway.

Orphanos, G. & Tsalidis, C. (1999). Combining handcrafted and corpus-acquired Lexical Knowledge into a Morphosyntactic Tagger. *In Proceedings of the 2nd Research Colloquium for Computational Linguistics in United Kingdom (CLUK)*, Essex, UK.

Passoneau, R. & Litman, D.J. (1993). Intention - based segmentation: Human reliability and correlation with linguistic cues. *In Proceedings of the 31st Meeting of the Association for Computational Liguistics*, pp. 148-155.

Petridis, V., Kaburlazos, V., Fragkou, P. & Kehagias, A. (2001). Text Classification using the σ-FLNMAP Neural Network. *In Proceedings of the IJCNN'01 on Neural Networks*.

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

Reynar, J.C. (1998). Topic Segmentation: Algorithms and Applications. Ph.D. Thesis, Dept. of Computer Science, Univ. of Pennsylvania.

Reynar, J.C. (1999). Statistical models for topic segmentation. *In Proceedings of the 37th Annual Meeting of the Association for Computational Liguistics*, pp. 357-364.

Roget, P.M. (1977). *Roget's International Thesaurus*. Harper and Row, 4th edition.

Stamatatos, E., Fakotakis, N., & Kokkinakis, G. (2001).*Computer-Based Authorship Attribution Without Lexical Measures*. Computer and the Humanities vol35, Kluwer Academic Publishers, pp. 193-214.

Utiyama, M. & Isahara, H. (2001). 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.

Xu, J. & Croft, W.B. (1996). Query expansion using local and global document analysis. *In Proceedings of the 19th Annual International of Association of Computer Machinery - Special Interest Group on Information Retrieval (ACM / SIGIR) Conference on Research and Development in Information Retrieval*, pp. 4-11.

Yaari, Y. (1997). Segmentation of expository texts by hierarchical agglomerative clustering. *In Proceedings of the Conference on Recent Advances in Natural Language Processing*, pp. 59-65.

Yamron, J. I., Carp, I.,Gillick,L. Lowe, S. & van Mulbregt P. (1999). Topic tracking in a news stream. *In Proceedings of DARPA Broadcast News Workshop*, pp. 133-136.

Youmans, G. (1991). *A new tool for discourse analysis: The vocabulary management profile*. Language, vol. 67, pp. 763-789.

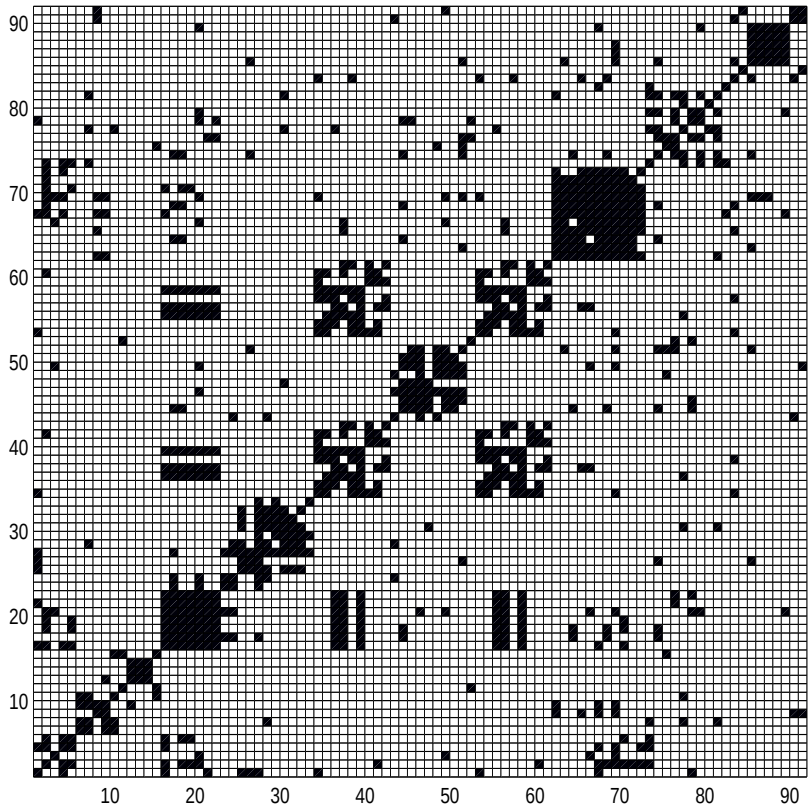

Figure 1: The similarity matrix D corresponding to a text from the dataset '9-11' of Set5. This text contains 91 sentences, hence D is a 91 x 91 matrix. A black dot at position (m,n) indicates that the m-th and n-th sentences have at least one word in common.

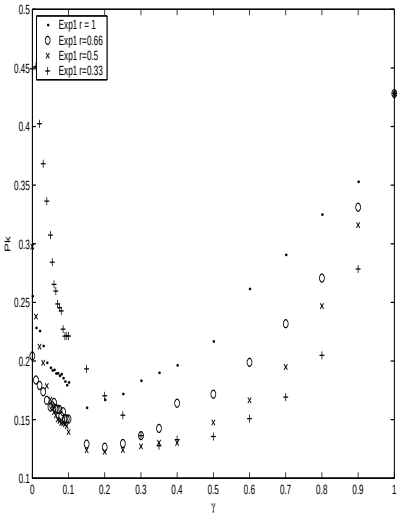

Figure 2: P_k for '3-11' of Set5

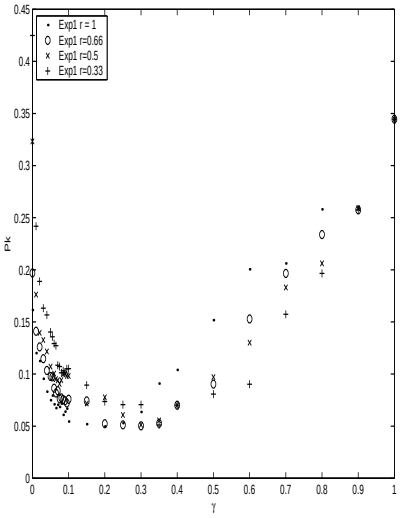

Figure 3: P_k for '3-5' of Set5

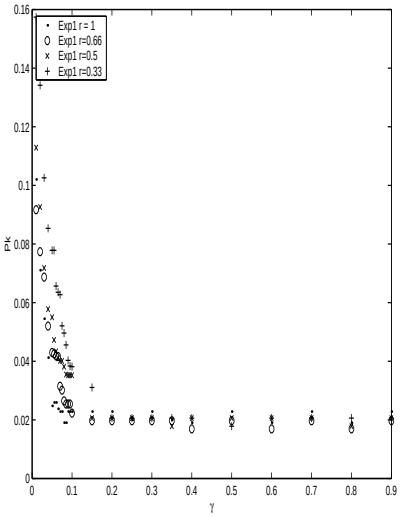

Figure 4: P_k for '6-8' of Set5

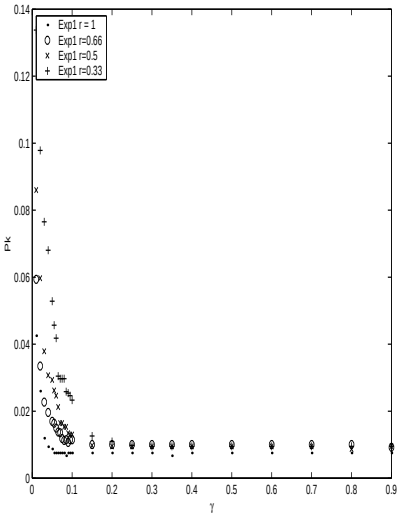

Figure 5: P_k for '9-11' of Set5