Honours Year Project Report

Linear and Hierarchical Text Segmentation Using Product Partition Models

By

Teo Yung Kiat

Department of Computer Science

School of Computing

National University of Singapore

2004/2005

Honours Year Project Report

Linear and Hierarchical Text Segmentation Using Product Partition Models

By

Teo Yung Kiat

Department of Computer Science

School of Computing

National University of Singapore

2004/2005

Project No: H079070
Advisor: Assistant Professor Kan Min-Yen

Deliverables:

Report: 1 Volume
Program: 1 CD

Abstract

This thesis proposes a semi-statistical method to automatically partition a stream of text into coherent segments using product partition models (Barry and Hartigan, 1992). This is done through the use of a modified dynamic programming algorithm proposed by Ath. Kehagias , P. Fragkou and V. Petridis (2002). In addition, we shall discuss an approach to recursively apply the above algorithm to organize the resulting linear segments into a hierarchical structure

Performance of the system was evaluated using standard Information Retrieval evaluation indices and two other evaluations metrics, the P_k metric (Beeferman, Berger, and Lafferty, 1997) and WindowDiff (Pevzner and Hearst, 2002). As no previous evaluation measures have been proposed to test the fitness of a hierarchical structure. I test the concept of my work by concatenating segments from different texts to form a hierarchical structure and then seeing if that structure can be recovered automatically.

Subject Descriptors

H.3.1 Content Analysis and Indexing (Indexing Methods)
I.2.7 Natural Language Processing (Text Analysis)
I.5.4 Application (Text Processing)

Keywords

text segmentation, machine learning, product partition models, lexical chains, dynamic programming

Implementation Software and Hardware

Windows XP, Java 2 SDK, SE v1.4.2, Toshiba Intel Pentium III processor, 628 MHz, 348 MB RAM

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my project supervisor Assistant Professor Kan Min-Yen for advising and patiently guiding me throughout the course of my research.

I would also like to thank my friends, Ang Han Sheng and Voon Kian Loon for helping me proofread this thesis. Thanks also to all my other friends and course mates who have lent me their support and encouragement.

List of Figures

Contents

1 Introduction

1.1 Background

The World Wide Web is a vast repository of information that connects people, providing them access to millions of web resources via the Internet. Although there are many types of representation of this information such as images, videos, etc., free-form text is still by far the most common. There are hundreds of millions of documents on a wide variety of topics available on the net.

As the numbers of these documents grow, so does the need for an efficient and effective method to access the information. Text and discourse processing techniques such as text classification, text segmentation, and text summarization have been the groundwork for organizing such information. Text segmentation, in particular, has been a very useful technique for segmenting unstructured text and building topic and subtopic structure, which can then be used to facilitate indexing and retrieval.

1.2 What is Text Segmentation?

Text segmentation is the effort to identify boundaries where a change of topic occurs in order to segment a continuous stream of unstructured text into coherent stories.

Chafe'76:
Our data ... suggest that as a speaker moves from focus to focus (or from thought to thought) there are certain points at which they may be a more or less radical change in space, time, character configuration, event structure, or even world ... At points where all these change in a maximal way, an episode boundary is strongly present.

Although the task of segmenting text into coherent stories appears to be simple, it has proven a difficult problem to resolve. There are many important and interesting discourse phenomena at the level of the paragraph or multi-paragraph to be taken care of by the text segmentation algorithm in segmenting the texts into multi-paragraph or topical

segments. The structure may be linear or hierarchical and it reveals the relationship between the substructure of the texts, and its content meaning.

Generally, there is no strict definition to what a "topic" is. An intuitive way of representing the problem is to find boundaries that divide the texts into stretches of segments where each segment talks about a different matter. Hence, instead of attempting to describe what a "topic" is, we instead attempt to discover when the "topic" shifts. Such an approach also enables automatic text segmentation to be domain independent. As a result, much of the earlier research on text segmentation (such as Hearst's TextTilling algorithm) has been focused on detecting topic changes rather than identifying the topic itself.

A general-purpose tool for partitioning text or multimedia content into coherent regions will have a number of immediate practical uses. In the field of information retrieval, research in sub-topic segmentation was originally inspired by the following problem: given a large unpartitioned collection of expository text (such as large numbers of newspaper articles strung together) and a user's query, return a collection of coherent segments matching the query. Lacking a tool for detecting topic breaks, an IR application may be able to locate positions in its database which strongly match the user's query, but be unable to determine how much of the surrounding data to provide to the user.

Other practical uses of document segmentation are: improved automated summarization (Salton et al., 1994; Barzilay and Elhadad, 1997), automated genre detection (Karlgren, 1996) and audio or video summarization. In particular, automatic detection of story and topic boundaries in news feeds has recently been the focus of much research. (Allan et al, 1998; Beeferman, Berger, and Lafferty, 1997; Beeferman, Berger, and Lafferty, 1999).

1.3 Previous Works

Text segmentation has been the focus of past research initiatives, as documented in the literature. Most text segmentation techniques fall into two categories (Allan, 1998): content-based technique, which focuses on the story content, and discourse-based technique, which focuses on story structure or discourse.

Content-based approaches resolve the segmentation problem by relying on some measure of the difference in word usage on the two sides of a potential boundary: the larger the difference, the more indicative of a boundary.

The TextTiling algorithm (Hearst, 1993) is a well-known example of a content-based approach. It is a domain-independent technique that assigns a score to each topic boundary candidate based on a cosine similarity measure (Salton, 1983) between chunks of words appearing to the left and right of the candidate. Topic boundaries are placed at the locations of valleys in this measure, and are then adjusted to coincide with known paragraph boundaries. Hearst's success in showing that lexical repetition can be used to locate topic boundaries in a stream of text prompted other researches to investigate similar approaches.

Another content-based approached uses a graphically motivated text segmentation technique called dotplotting (Reynar, 1994). A similarity matrix is obtained from the text and then plotted on a graph. Dense regions on the graph correspond to tight regions of topic similarity and are used to determine how the topic segments are distributed.

The dragon approach (Allan, 1998) uses Hidden Markov Model (HMM) techniques in a way similar to the one used in speech recognition. It treats a story as an instance of some underlying topic. In the model, the hidden states are topics and the observations are words or sentences.

A more recent content-based approach utilizes Product Partition Models (Kehagias et al., 2002) in an attempt to model text segmentation as an optimization problem. The model makes use of prior information obtained from training texts in order to predict the average segment length of the text to be segmented. It has been found that this yields a significant increase in the accuracy of the resulting segmentation when this information is used in conjunction with content information from the text.

Morris and Hirst (1991) first proposed the notion of *lexical chains* to chain semantically related words (i.e. synonyms) together via a thesaurus. The chains are constructed out of selected content terms in the document and represent the lexical cohesive structure of a text. This approach is different but somewhat related to content-based approaches that make use of the difference in word usage at different parts of a text to locate a potential boundary.

Other researchers have subsequently used this method for text segmentation. For example, *Segmenter* (Kan et al., 1998) assigns a score to each paragraph in a text based on their relative positions (beginning, end or at the middle) in the lexical chains. The scores are then used to indicate whether the paragraph marks a topic boundary.

Discourse-based approaches make use of prosodic features such as pause duration as well as lexical features such as the presence of certain cue phrases or cue-words that tend to appear near the segment boundaries. Such methods tend to be domain-specific because of their dependence on the style of text.

The HMM Segmentation method (Allan, 1998) studied by Umass uses a Hidden Markov Model to model "marker words", or words which predict a topic change. It relies on words that predict the beginning or the end of a segment.

The CMU method (Beeferman, 1999; Beeferman and Berger, 1997; Della Pietra, 1997) is an example of a hybrid approach that combines the content and discourse based methods. It uses a statistical framework for feature selection. It can select different features and style of texts and combine content features and discourse features together. The method builds an exponential model incrementally to extract features that are correlated with the presence of boundaries in labeled training text.

1.4 Motivations for this Research

In this research, I will investigate a hybrid approach that combines the usage of lexical chains in addition to information obtained from word repetition in a text to discover its structure. Previous research has used only one or the other but not both. I have found that when I added information obtained from lexical chaining to a previous content-based approach that used only word repetition, I was able to obtain a 20% increase in the segmentation accuracy.

Previous work on text segmentation focused mainly on linear rather than hierarchical text segmentation. Whereas linear segmentation only attempts to find topic boundaries within a text without any concern of their underlying structure, hierarchical segmentation not only locates the subtopics, but also tries to organize them into a meaningful hierarchy.

A hierarchical segmentation may be more useful than a linear one for many real world applications. For example, most textbooks, technical manuals, research papers etc. have an inherent hierarchical structure. Thus, the other motivation of my research is to tackle the problem of *hierarchical* text segmentation. This will be done in two phases. First, I will implement a linear segmentation algorithm to obtain the linear subtopic segments of a text. Next, I will apply a hierarchical clustering algorithm to organize the segments into a hierarchical tree.

1.5 Thesis Structure

The rest of this thesis will be organized as follows. Chapter 2 will discuss the Product Partition Model framework, which will be the basis of this project. In Chapter 3, I will introduce *TextSegmenter*, the system I have created to accomplish the task of text segmentation. Chapter 4 will deal with the *linear* segmentation results of *TextSegmenter* from experiments using Freddy Choi's corpus (Choi, 2000). Chapter 5 will introduce the hierarchical clustering algorithm to cluster linear segments into a hierarchical structure, which is the second part of this project. Chapter 6 will discuss the experiments done to prove the validity of the hierarchical clustering algorithm. Finally, I end with Chapter 7 by giving a conclusion and some basis for future work.

2 The Product Partition Model framework

For this project, I have utilized prior research on using Product Partition Models to formulate text segmentation as an optimization problem, which is then solved using a dynamic programming algorithm. (Kehagias et al. 2002). I have modified the algorithm (as well as add in an additional feature) to give better results for *linear* text segmentation. Furthermore, in the later part of this thesis, I will introduce a way to recursively apply the algorithm so as to achieve *hierarchical* text segmentation. (Note that the original paper was focused solely on the linear aspect).

The objectives of my project are thus as follows:

a. To improve an existing linear segmentation algorithm such that it gives better results than any others so far reported in the literature.
b. To introduce an algorithm to organize linear text segments into a hierarchical structure.

A Product Partition Model (PPM) (Barry & Hartigan, 1992) is a Bayesian inference procedure for segmentation of a sequence of random variables, based on the heterogeneity of the sequence. The model assumes that observations in different components of a random partition of the data (for example, segmented text) are independent. If the probability distribution of random partitions is in a certain product form making the observations, it is also in product form given the observations.

Applying the PPM framework to text segmentation, we can define the posterior joint probability of an observed text and its segmentation by the product of two terms:

a. The prior probability of the segmentation, which is derived through the use of a corpus of training texts that are in the same domain as that of the observed text. This is described by an appropriate *homogeneity* function.
b. The conditional (given the segmentation) probability of the observed text, which is described by a *cohesion* function that is made up of two parts. The first part uses information from lexical repetition in the text. The second part uses information obtained from lexical chaining. (Morris and Hirst, 1991).

Mathematically speaking, suppose we let *o* represent the observed text, and let *t* range over the set *T* of all possible segmentations of the text. Using Bayes' theorem, the most likely segmentation is then:

$$\hat{t} = \underset{t \in T}{\operatorname{argmax}} \quad \overbrace{P(o \mid t)}^{part\ b} \cdot \overbrace{P(t)}^{part\ a}$$

The negative logarithm of the joint probability is the segmentation cost, and this is minimized using a dynamic programming algorithm.

2.1 Problem Representation

Consider a text with T sentences. We label each sentence from 1 to *T* according to their order in the text. 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*}; {t_0, t_1, …t_K} are the segment boundaries that partition the text such that they satisfy:

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

Sentence 0 is a dummy sentence, which we use to represent the start of the text. Hence, any segmentation will contain the two boundaries $t_0 = 0$ and $t_K = T$; the dummy sentence at the beginning and the last sentence at the end of the text. We assume that segment boundaries always appear at the end of sentences.

We denote a possible segmentation using the vector $\hat{\mathbf{t}} = \{t_0, t_1, \ldots t_K\}$ and the set of all possible segmentations of {1, 2…*T*} using Φ_T. Our task, then, is to find the most probable segmentation vector $\hat{\mathbf{t}}$ in Φ_T.

2.2 Text Representation

Suppose that the text has a vocabulary of L distinct words, {1,2…L}. We can represent the text using a T x L matrix $\boldsymbol{c}$, where (for t = 1,2…T and l = 1,2…L):

$$c_{t,l} = \begin{cases} 1 & \text{iff the l–th word appears in the t–th sentence} \\ 0 & \text{otherwise} \end{cases}$$

We use $\mathbf{c}_1$, $\mathbf{c}_{2,}$ … $\mathbf{c}_T$ to denote the *sentence vectors*, which are binary strings of 1's and 0's (either a word of the vocabulary appears in the sentence, or it does not).
From the text, we can extract two classes of information, which we represent using a sentence similarity matrix and a lexical chain matrix, denoted by $\mathbf{d_1}$ and $\mathbf{d_2}$ respectively.

2.2.1 Sentence similarity matrix

The sentence similarity matrix of the text is a T x T matrix $\mathbf{d_1}$ such that:

$$\mathbf{d}_{\mathbf{1}s,\,t} = \begin{cases} \mathbf{cos\_sim}(c_s, c_t) & if\ s \neq t \\ 0 & otherwise \end{cases}$$

where $\mathbf{cos\_sim}(\ )$ is the cosine similarity function. The cosine similarity measure (Salton, 1987) is a metric used to measure the similarity between two documents. It is defined as the cosine of the angle between two vectors v and w such that:

$$\cos\theta = \frac{\sum_{i=1}^{n} v_i w_i}{\sqrt{\sum_{i=1}^{n} (v_i)^2} \cdot \sqrt{\sum_{i=1}^{n} (w_i)^2}}$$

In other words, $\mathbf{d}_{\mathbf{1}s,\,t}$ is the cosine similarity value between sentence s and sentence t. The resulting matrix $\mathbf{d_1}$ has zeros and non-zero values arranged in a characteristic pattern

which corresponds to the structure of the text. In the figure below, a dotplot (Reynar, 1994) of the matrix corresponding to a sample article is given.

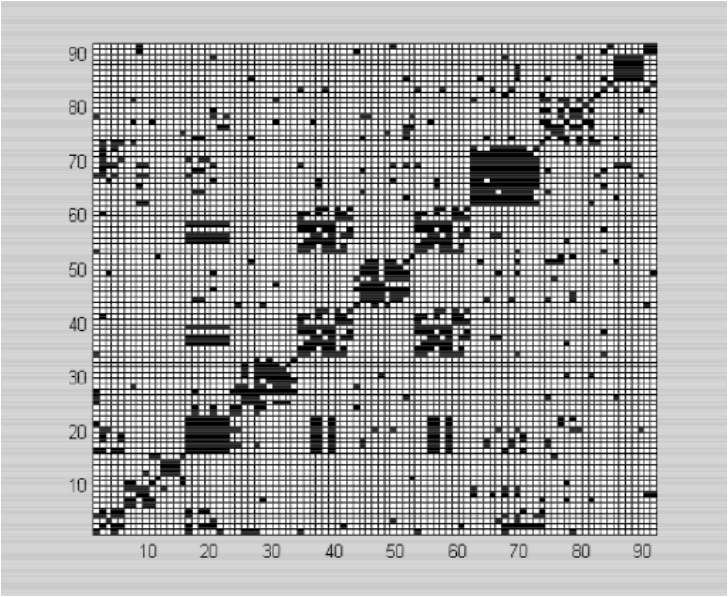

Figure 1: Dotplot of sample article

We will use the notation $\mathbf{d_1}(s, t)$ to denote the square submatrix of $\mathbf{d_1}$ defined by positions $(s + 1, s + 1)$ and (t, t). For every $s = 0, 1, \ldots T\text{-}1$ and $t = 1, 2, \ldots T$, we obtain a submatrix $\mathbf{d_1}(s, t)$ which corresponds to the segment $\{s+1, s+2, \ldots t\}$. Assuming that sentences belonging to the same subtopic (i.e. segment) will have many words in common, then a submatrix which corresponds to an actual segment will have many non-zero values in it. A good segmentation algorithm will thus divide the matrix into submatrices that are densely packed with non-zero values.

2.2.2 Lexical Chain Matrix

Morris and Hirst (1991) first proposed the notion of lexical chains to chain semantically related words together via a thesaurus. For this project, I will use a simpler interpretation of it and chain only words with the same stem. (Kan et al. 1998)

Given a term and the distribution of its occurrences, we can link related occurrences together. The metric for determining relatedness is the proximity of the

terms. If two occurrences of a term occur within *n* sentences, we link them together as a single unit, and repeat until no larger units can be built. This is illustrated in the diagram below.

```
sents   01234567890123456789012345678901234567890123456789012345
cat     1xx1                    1xx1            1x1
ball      1xx1                1xx1                11xx1
```

Figure 2: Lexical chains formed by the terms 'cat' and 'ball'.

Figure 2 shows the lexical chains formed by the terms “cat” and “ball”. The numbers at the top represent the sentence numbers of the text and are grouped in tens. The term “cat” appears a total of six times, “ball”, a total of seven. The occurrences of the terms are grouped together into several term links, as joined by the “x”s.

A “good” segment {s+1, s+2, …t} should have many lexical chains that start at one end of its boundary (s+1) and ends at the other. However, we do not always expect the ends of the chains to coincide *exactly* with the boundaries of the segment. The ends of the chain might just miss touching them and this should still be acceptable. We, thus, allow for a certain distance, b, such that if either ends of a chain miss the boundaries by a distance less than or equal to b, we still consider the chain to be *contained* within the segment. Note that the word “contain” might be slightly misleading. A segment does *not* contain a chain simply because the chain is within its boundaries. The ends of the chain must be within the allowable distance from the boundaries. To further illustrate this, see figure 3 on the next page.

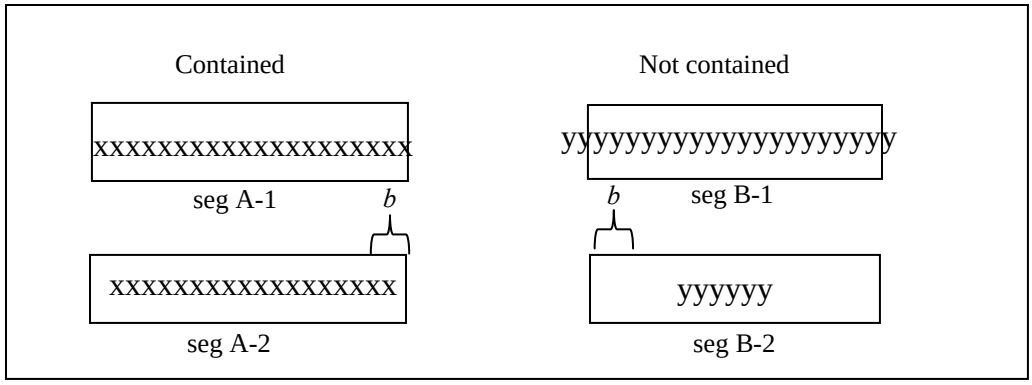

Figure 3: Example of segments containing lexical chains

In Figure 3, both segments A-1 and A-2 are considered to *contain* the chain made up of "x"s. Even though the ends of the chain in A-2 do not exactly touch the boundary, it is still within the allowable distance, *b*. In contrast, the chain in B-2 exceeds this limit, and the one in B-1 extends beyond the boundaries of the segment.

We now proceed to define the lexical chain matrix. Like the sentence similarity matrix, the lexical chain matrix of the text is a T x T matrix $\mathbf{d_2}$ where:

$$\mathbf{d}_{2_{s,t}} = \begin{cases} \text{total number of chains contained in seg(s,t)} & \text{if } s < t \\ 0 & \text{otherwise} \end{cases}$$

(seg(s, t) refers to the segment that starts from sentence $s+1$ and ends at sentence t).

The two matrices, $\mathbf{d_1}$ and $\mathbf{d_2}$ are used in conjunction with each other to give a representation of the observed text. Every text can thus be described by a two-tuple {$\mathbf{d_1}$, $\mathbf{d_2}$}. From here on, we will simply use **d** to represent the two-tuple and **d**(s, t) to denote the square submatrices of both $\mathbf{d_1}$ and $\mathbf{d_2}$. Furthermore, we let Ψ_T denote the set of all possible **d**'s.

2.3 Product Partition Models

We now define two random variables, the *segmentation* variable **T**, which takes values in Φ_T, and the *lexical similarity* variable **D**, which takes values in Ψ_T. The two variables are each specified by its probability function. We will denote a probability function by the letter *f*. For example,

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

A Product Partition Model (PPM), is a pair of random variables (**D**, **T**) which have a particular type of joint probability distribution $f_{\mathbf{D},\mathbf{T}}$ such that

1. The probability of a particular segmentation $\{t_0, t_1, \ldots t_K\}$ has the form:

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

   where $c(s, t)$ is a *homogeneity* function and K can take any value between 1 and *T*.

2. Conditional on $\mathbf{T} = (t_0, t_1, \ldots t_K)$, the probability density of the submatrix

   $\mathbf{D}(t_{K-1}, t_K)$, has the form (for $k = 1,2,\ldots, K$):

$$f_{\mathbf{D}(t_0, t_1, ..., t_K)|\mathbf{T}}(\mathbf{d}(t_{K-1}, t_K) | (t_0, t_1, ..., t_K)) = g(\mathbf{d}(t_{k-1}, t_k))$$

   where $g(\mathbf{d}(t_{k-1}, t_k)$ is the *cohesion* function.

Combining 1 and 2, the joint probability of **D** and **T** has the form

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

Hence, the PPM is made up of a *cohesion* function, which assigns probabilities independent of the observed text, and the *homogeneity* function, which assigns a probability to each segment, *given* a particular segmentation.

In the next sections, we will discuss the homogeneity and cohesion functions used for this project.

2.3.1 Homogeneity function

The homogeneity function is defined as

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

(where $0 \le s < t \le T$)

This function is used to incorporate some information regarding the segment length of the text, without actually seeing the text. It can be used when we know that the text being segmented is of a particular domain and we have training data from that domain (i.e. a corpus of Wall Street Journal articles). We are thus able to use the data to estimate μ, the mean value, and σ, the standard deviation of the segment length.

2.3.2 Cohesion function

The cohesion function is defined as:

$$g(\mathbf{d}(s,t)) = \exp\left[\left(\frac{\sum_{i=s+1}^{t}\sum_{j=s+1}^{t} d_{1i,j}}{(t-s)^r}\right) + d_{2i,j}\right]$$

(*r* is a parameter).

$\sum_{i=s+1}^{t}\sum_{j=s+1}^{t} d_{i,j}$ is the sum of all the cosine similarity values in the submatrix of **d**(s, t) which corresponds to the segment {*s*+1, *s*+2, …, *t*}. When *r* = 2, (*t* - *s*)^r gives us the area of the corresponding square submatrix. The term $\left(\sum_{i=s+1}^{t}\sum_{j=s+1}^{t} d_{1i,j}\right)\Big/(t-s)^r$ is the segment *density*, and $\mathbf{d_{2}}_{i,j}$ gives the total number of lexical chains that the segment contains. A high value of $\left(\sum_{i=s+1}^{t}\sum_{j=s+1}^{t} d_{1i,j}\right)\Big/(t-s)^r + d_{2i,j}$ indicates strong intra-segment similarity (i.e. the segment contains many repeated words).

2.3.3 Segmentation Cost

With the above functions defined, we can now proceed to define the *segmentation cost* as follows:

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

(γ is used to control the relative importance of the cohesion and homogeneity functions). The total segmentation cost is the sum of the cost of the K segments of the text; the cost of the k-th segment is the sum of the two terms.

A good segmentation gives segments that have high density and contain many lexical chains (homogeneity function), while at the same time, does not deviate much

from the average segment length (cohesion function). The corresponding segmentation cost $\boldsymbol{J(t;\mu,\sigma,r,\gamma)}$ will thus have a small value. An *optimal* segmentation will give the global minimum of $J(t;\mu,\sigma,r,\gamma)$

$$\hat{t} = \arg \min_{t=(t_0,t_1,\ldots,t_K)} J(t;\mu,\sigma,r,\gamma)$$

The discussion of the PPM framework for text segmentation is thus concluded. In the next section, I will introduce *TextSegmenter*, a system I have created to accomplish the task of segmenting text files.

3 TextSegmenter

3.1 TextSegmenter System Architecture

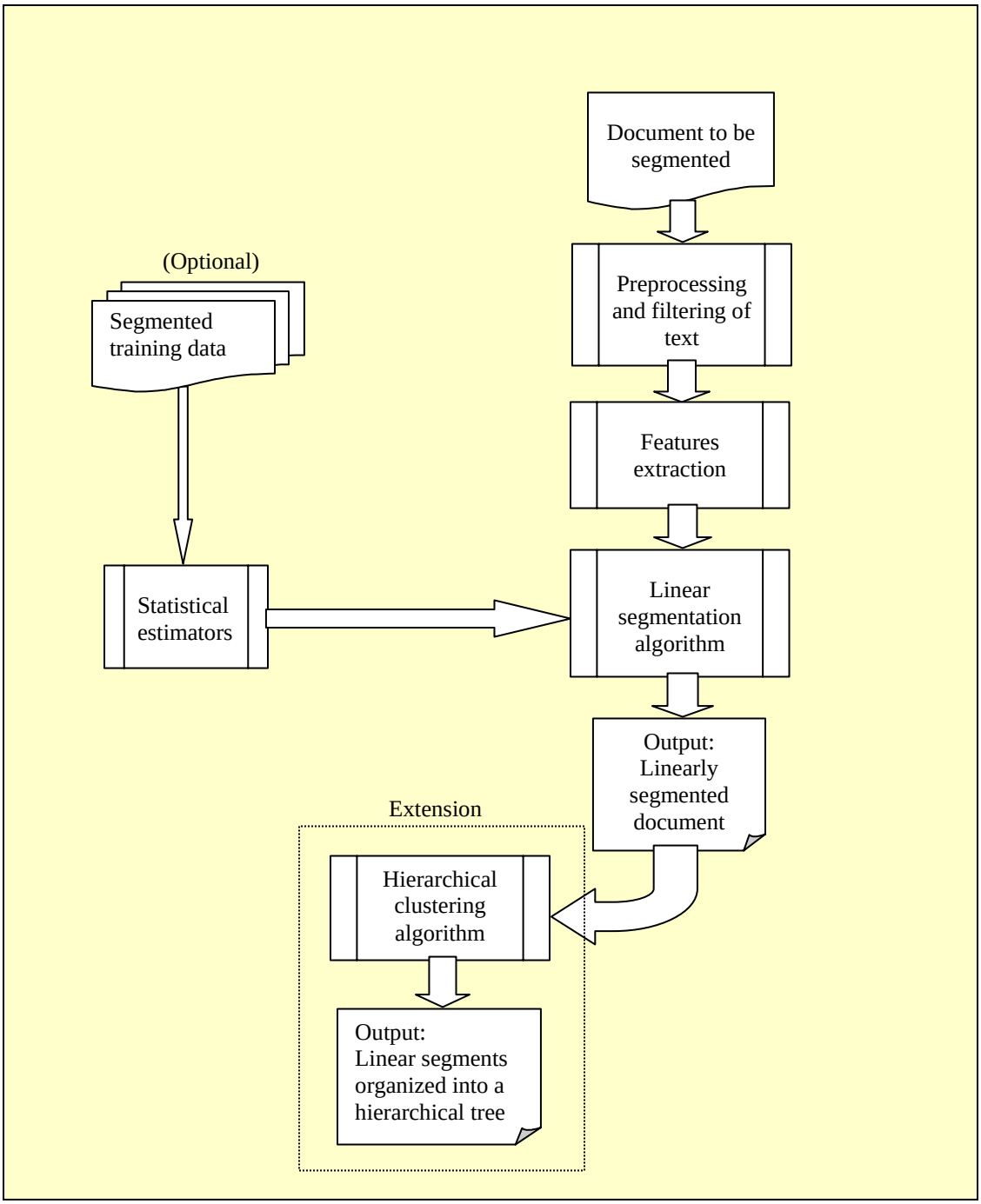

Figure 4: *TextSegmenter* system flow

Figure 4 shows the flow of the TextSegmenter system. The rest of this section will be devoted to explaining each of the various stages.

3.2 Estimating statistical values through the use of training data

If segmented training data is available, the mean value, μ, and standard deviation, σ, of the segment length can be obtained. These values are needed in the homogeneity function of section 2.3.1. If they are not available, then we set γ to zero and omit this part.

3.3 Preprocessing and filtering of text

The input text is first split into its component sentences using MXTerminator (Ratnaparkhi, 1994). Next, punctuation and stop-words (determined by a stoplist) are filtered out. Finally, the remaining words are stemmed using Porter Stemming Algorithm (Porter, 1980).

3.4 Features Extraction

During this stage, the sentence similarity matrix $\mathbf{d_1}$ and the lexical chain matrix $\mathbf{d_2}$ are extracted from the preprocessed text. (See sections 2.2.1 and 2.2.2)

3.5 Linear Segmentation algorithm

Given the sentence similarity matrix $\mathbf{d_1}$, the lexical chain matrix $\mathbf{d_2}$, and the parameters μ, σ, r, γ, we can now proceed with the dynamic programming algorithm to minimize the global segmentation cost.

Segmentation Algorithm

Input: The sentence similarity matrix $\mathbf{d_1}$; the lexical chain matrix $\mathbf{d_2}$; the parameters μ, σ, r, γ .

Minimization:

$C_0 = 0$

$z_0 = 0$

For t = 1,2,…,T

$C_t = \infty$

For s = 0,1,…,t – 1

If $C_s + \gamma \cdot \frac{(t-s-\mu)}{2\sigma^2} - (1-\gamma) \cdot \left[\frac{\sum_{i=s+1}^{t} \sum_{j=t_{k-1}+1}^{t_k} d_{1i,j}}{(t-s)^r} + \mathbf{d_{2i,j}} \right] \le C_t$

$$C_t = C_s + \gamma \cdot \frac{(t-s-\mu)}{2\sigma^2} - (1-\gamma) \cdot \left[\frac{\sum_{i=s+1}^{t} \sum_{j=t_{k-1}+1}^{t_k} d_{1i,j}}{(t-s)^r} + d_{2i,j} \right]$$

$z_t = s$

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 $k = 0,1,\ldots K$

$\hat{t}_k = s_{K-k}$

End

Output: The optimal segmentation $\hat{\mathbf{t}} = (t_0, t_1, ..., t_k)$

The algorithm runs in O(T^2) time.

3.6 Linear Segmentation Output

The output of the linear segmentation part of *TextSegmenter* is a text file containing the original input document arranged line by line (for each sentence). Lines are drawn to indicate where the subtopic boundaries lie.

3.7 Extension – Hierarchical clustering of segments

Depending on the needs of the user, a linearly segmented text may be enough for his purposes. The extension to *TextSegmenter* provides an algorithm to cluster the segments obtained from the earlier stages into a hierarchical structure. This part of the system involves the second objective of my project. I will delay the explanation on how this is done for now, and, in the next chapter, go on to the discussion on the experiments done to evaluate the accuracy of *TextSegmenter* as a linear segmentation program.

4 Experiments and Evaluation

This part of my thesis will focus on evaluating the linear segmentation algorithm, which I introduced in section 3.5.

4.1 Evaluating Text Segmentation algorithms

There are two main problems with evaluating a text segmentation algorithm. Firstly, human readers do not always agree with each other where boundaries should be placed or how fine-grained an analysis should be making it difficult to choose a reference segmentation for comparison. Some evaluations solve this problem by detecting boundaries in sets of concatenated documents, where there can be no dispute on where the boundaries lie (Reynar, 1994; Choi, 2000). Others have human judges make ratings and then take the majority agreement as the standard. In this project, I shall use the former approach.

The second problem is that for different types of applications for text segmentation, different errors become important. For example, for information retrieval, it can be acceptable for boundaries to be off by a few sentences (this is known as a near-miss), whereas for boundary detection of news feeds, accurate placement is crucial.

In the sections below, I will discuss the standard evaluation indices, which have been used for text segmentation algorithms.

4.1.1 Precision and Recall

Precision and recall are both standard evaluation measures used in Information Retrieval applications that are concerned with retrieving documents to match a given query. When used in the context of text segmentation, they are defined as follows:

Precision is the percentage of boundaries identified by an algorithm that are indeed true boundaries; recall is the percentage of true boundaries that are identified by the algorithm. Precision and recall have two major problems. The first problem is that

there is an inherent tradeoff between precision and recall, that is, the accuracy of one can be improved at the cost of the other.

The second problem is that precision and recall are not sensitive to near misses. This means that they penalize equally heavily on boundaries that miss by a wide margin (fifteen or more sentences) as compared to boundaries that miss by say, only a couple of sentences. This shortcoming means that it is unable to differentiate between algorithms that assign boundaries that are close to actual boundaries, and algorithms that assign boundaries that miss the actual ones by a wide margin. Intuitively, we would like to be able to give a higher score to the former type of algorithms.

4.1.2 The P_k Evaluation Metric

The P_k Evaluation Metric (Beeferman, Berger, and Lafferty, 1997) was introduced to resolve the problems with precision and recall, and allows for the assigning of partial credit to near misses.

P_k is defined as *the probability that two sentences drawn randomly from the corpus are correctly identified as belonging to the same document or not belonging to the same document.*

P_k is scaled between 0 and 1. An algorithm that assigns all boundaries correctly receives a score of 0. Boundaries are penalized according to the distance they are from true boundaries - the larger the distance, the greater the penalty.

4.1.3 WindowDiff

Though P_k is fast becoming the standard among researchers in text segmentation, there are still certain drawbacks to it such as false negatives being penalized more heavily than false positives. Pevzner and Hearst (2002) have thus made modifications to the error metric algorithm to remedy its problems. They have called the resulting fix WindowDiff.

4.2 Experiments using Choi's collection

For the experiments on *TextSegmenter*, I have made use of Freddy Choi's collection (Choi, 2000), which consists of texts made up of ten concatenated text segments. Each segment consists of the first *n* sentences of a randomly selected document from the *Brown Corpus* (Francis and Kucera, 1982). The texts are divided into four datasets and are listed in the table below. There are a total of *700* texts.

	Set 0	Set 1	Set 2	Set 3
No. of texts	400	100	100	100
Range of *n*	3-11	3-5	6-8	9-11

Table 1: Statistics of Choi's collection

I have measured segment accuracy using the three different evaluation measures described in the previous sections (4.1.1 – 4.1.3).

My evaluation will take place in two stages. In the first stage, I will compare the results of segmentation using solely word repetition (feature $\mathbf{d_1}$) against using a combination of both lexical chaining (feature $\mathbf{d_2}$) *and* word repetition. The homogeneity function of the algorithm will be omitted in this stage. The purpose of this is to show that adding information from lexical chaining can indeed improve the accuracy of the segmentation algorithm.

The second stage will consist of evaluating the algorithm in its entirety, i.e., using both the homogeneity and cohesion functions, with the cohesion function comprising of both features $\mathbf{d_1}$ and $\mathbf{d_2}$.

4.2.1 Stage 1 – Evaluating effectiveness of lexical chain feature

	Set 0	Set 1	Set 2	Set 3	All Sets
Precision	0.895	0.869	0.847	0.9203	0.883
Recall	0.321	0.438	0.3768	0.389	0.361
P_k	0.45	0.452	0.432	0.374	0.447
WindowDiff	0.37	0.312	0.356	0.2822	0.358

Table 2: Experiment results without lexical chaining

	Set 0	Set 1	Set 2	Set 3	All Sets
Precision	0.797	0.770	0.808	0.887	0.816
Recall	0.734	0.852	0.830	0.905	0.830
P_k	0.148	0.118	0.105	0.0373	0.102
WindowDiff	0.149	0.140	0.119	0.0442	0.114

Table 3: Experiment results with lexical chaining

Tables 2 and 3 above show the evaluation results of the algorithm with and without the use of feature $\mathbf{d_2}$**.** For the experiments above, γ was set to zero (thus omitting the homogeneity function), and r was set to 0.66 (as per the original paper). For simplicity, we let $n = b = 6$. This value was obtained through the use of training data.

	Set 0	Set 1	Set 2	Set 3	All Sets
Without $\mathbf{d_2}$	0.372	0.312	0.356	0.282	0.328
With $\mathbf{d_2}$	0.149	0.140	0.119	0.0442	0.114

Table 4: Comparison of results with and without lexical chaining

Table 4 shows a comparison between the two methods using *Windiff* as the evaluation measure. We can see that there is an significant increase in the segmentation accuracy of about 20% when lexical chaining is used in conjunction with word repetition.

4.2.2 Stage 2 – Evaluating overall effectiveness of algorithm

	Set 0	Set 1	Set 2	Set 3	All Sets
Precision	0.833	0.842	0.911	0.928	0.878
Recall	0.830	0.836	0.907	0.925	0.874
P_k	0.115	0.0932	0.0393	0.020	0.0670
WindowDiff	0.101	0.0996	0.0390	0.020	0.0650

Table 5: Evaluation results using both homogeneity and cohesion functions

The table above shows the performance of *TextSegmenter* when both the homogeneity and cohesion functions are used together. μ and σ values were estimated using another subset of Choi's texts. Though training, we found that the algorithm gives optimal performance when γ was set to 0.5.

	Set 0	Set 1	Set 2	Set 3
Utiyama (2001)	9%	10%	7%	5%
$Choi_{(b)}$ (1999)	12%	12%	9%	9%
Choi (1999)	13%	18%	10%	10%
Reynar (1998)	22%	21%	18%	16%
Kan et al. (1998)	36%	23%	33%	43%
Hearst (1998)	46%	44%	43%	48%
TextSegmenter	11.5%	10%	4%	2%

Table 6: Comparison of *TextSegmenter* against other algorithms

Table 6 gives a comparison of the results of *TextSegmenter* against other algorithms in the literature (obtained from Choi's paper). The metric used for the comparison is the P_k error metric. The results show that *TextSegmenter* is more accurate than the others.

5 Organizing linear segments into a hierarchical structure

In the earlier part of this thesis, I have introduced a method to split a text document into its topical segments. In this section, I will discuss how to organize these linear segments into a well-formed topic hierarchy.

5.1 Why hierarchical?

Organizing linear segments into a hierarchical structure provides a comprehensive form to present the document from which the segments were retrieved. Having such a structure makes it easier for users to view and retrieve needed information.

The task can be defined as such: given contiguous text segments (the segments are arranged in the order in which they were retrieved from the document), organize them into a hierarchical structure. The diagram below illustrates this.

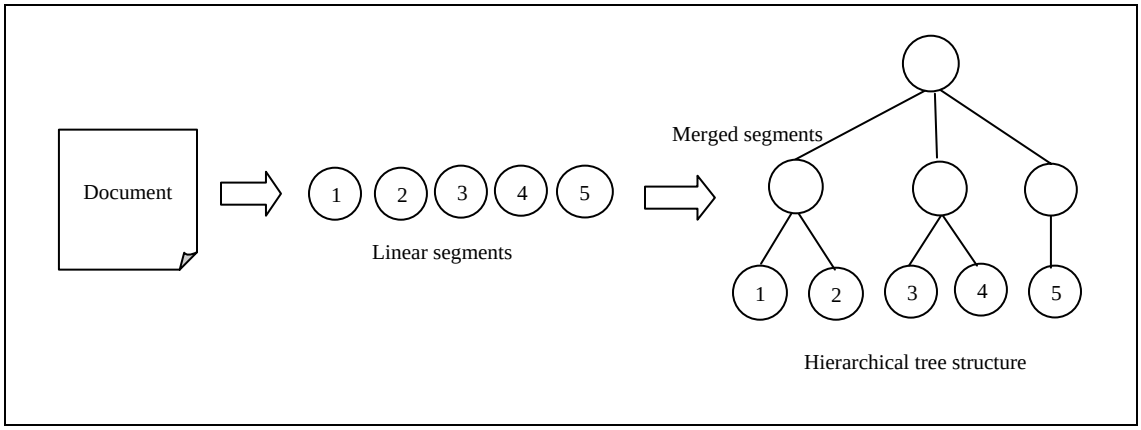

Figure 5: Getting a hierarchical tree from linear segments

5.2 Related Work

To my knowledge, there has been very little research done on this area. Most of the past work on text segmentation has linear rather than hierarchical. A past research paper (Yaari, 1997) had tried to use a variant of the HAC (hierarchical agglomerative clustering) algorithm to cluster paragraphs into a hierarchical tree. However, due to the

nature of the HAC, which always merges two segments at each pass, the resulting tree is a binary one. This is highly unlikely to be the structure of real world documents, where we would expect the tree structure to consist of a variable number of nodes at each level.

5.3 Getting a hierarchical structure through clustering

The approach I am going to use to accomplish this task follows naturally from the solution I have used to achieve linear text segmentation. The key idea of the proposed approach is to apply clustering to create the hierarchical topic structure of text segments. In the linear segmentation algorithm, we have treated sentences as the elementary segments (we assume that boundaries are found only at the end of a sentence). The algorithm then proceeds to partition the text such that the total segmentation cost of the segments is at a minimum. Thus, we effectively cluster sentences together in such a way that the segments that result are coherent and isolated from each other. In other words, the segments have high intra-similarity but low inter-similarity. We can follow up on this idea by treating the linear segments found from the previous step as the elementary segments. We then try to partition the text such that the grouped segments give the minimum global segmentation cost. We merge the segments together, and repeatedly do this until only one segment is left. The general algorithm for this is shown below:

Input: linear segments obtained from earlier segmentation.
While more than one segment left **do**
    Apply segmentation cost minimization algorithm to find the best way to partition segments $\{s_i, s_i+1,\ldots, s_n\}$. ($n$ decreases with each loop.)
    Merge segments within their respective partitions into a single segment.
End While

Recall that the linear segmentation algorithm makes use of features (the sentence similarity and lexical chain matrix) extracted from the text. We will need to reuse these features for the hierarchical clustering algorithm, the sections below discuss how.

5.4 Building the tree from bottom-up

Given the linear segments, we can proceed to organize them into a hierarchical structure. We will bring over the features extracted from the linear segmentation part. For simplicity, we will not take into account the *homogeneity* function but rely solely on the *cohesion* function.

Assuming that the linear segmentation is $\{t_1, t_2, \ldots t_K\}$, the corresponding submatrices would be $\mathbf{d}(t_1, t_2)$, $\mathbf{d}(t_2, t_3) \ldots \mathbf{d}(t_{K-1}, t_K)$. If these segments are truly sub-topics of the text, then we would expect them to be highly cohesive. The next step would be to reapply the minimization algorithm and repartition the segments to obtain the next level of the hierarchical tree. We keep doing this until only one cluster remains (i.e. the original text).

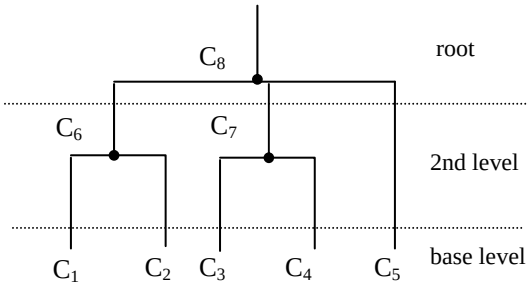

Figure 6: Example of hierarchical tree formed through clustering

In Figure 7, clusters C_1, C_2, C_3, C_4 and C_5 are the original linear segments. Clusters C_6 and C_7 are formed through merging C_1, C_2 and C_3, C_4 respectively. They form the second level of the tree. Finally, C_8 (the entire document itself) is formed through merging C_5, C_6, and C_7.

5.4.1 Recalculating the sentence similarity and lexical chain matrices

We cannot simply reuse the features extracted from the prior level to do the segmentation for the next level, as we would get back the exact same segmentation. The key idea here is that in order to obtain the next level of segments, we ignore their *intra*-similarity and only take into account of their *inter*-similarity. We are no longer interested in the cohesiveness of any single segment once it has been obtained; rather, we want to focus on their similarity to other segments in order to best decide the best way to cluster them.

In order to do this, we have to make changes to the matrices $\mathbf{d_1}$ and $\mathbf{d_2}$ during each pass. For each segment $\mathbf{d}(t_{k-1}, t_k)$, at level *n*, we set all values in the submatrices corresponding to the segments to zero. We then redo the minimization algorithm to obtain the segmentation for level *n*+1.

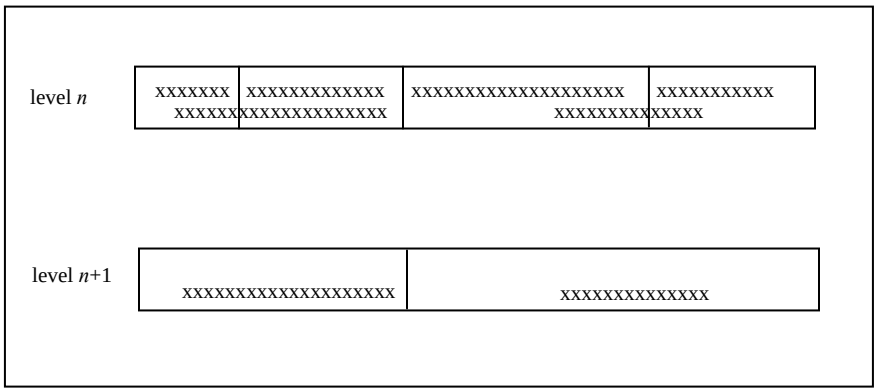

Figure 7: Recalculation of lexical chain matrix

Figure 5 shows the re-segmentation of a text before and after the recalculation of the lexical chain matrix. During level *n*, the optimal segmentation gives 4 segments, as this will give a total of 4 lexical chains. To obtain the next level, we ignore these 4 chains, which are contained within the segments, and only take into account the 2 chains that cross the segment boundaries. The new optimal segmentation for level *n*+1 is shown above.

This is done similarly for the sentence similarity matrix. Hence if we have segmentation $\hat{\mathbf{t}} = (t_0, t_1, ..., t_K)$ at level *n*, we let:

$$\left[\frac{\sum_{i=s+1}^{t} \sum_{j=t_{k-1}+1}^{t_k} d_{1_{i,j}}}{(t-s)^r} + d_{2_{i,j}} \right] = 0$$

(where $i = t_0, t_1, \ldots, t_{K-1}$, $j = t_1, t_2, \ldots, t_K$)

for level n+1.

5.5 Top-down partitioning

Once the bottom-up building of the tree is done, we get a rough shape of what the hierarchical tree structure is like, however, the tree is flawed as during the building process, if there are two or more unrelated segments that are contiguous in the text, they are clustered together mistakenly. Thus boundaries, which should exist, are falsely removed. (See figure below.)

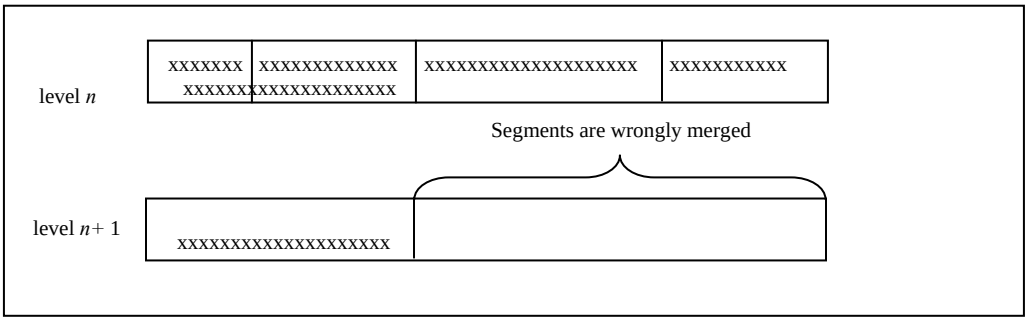

Figure 8: Unrelated contiguous segments are wrongly merged during bottom-up building of tree

In order to remedy this, we traverse the tree from top to bottom and compare the segmentation cost of each cluster (node) at level n of the tree with the segmentation cost of the entire document i.e. the cost of the segmentation if the entire text was to be treated as a single segment. If the segment cost of that node is higher, we split the node back into its constituent clusters. In other words, we restore all boundaries that had been taken away when merging its children nodes. If lower, we leave the segment as it is. Intuitively, this mean that if a single segment has a higher cost than the cost of the entire document, then that segment is *less* cohesive then if the document itself was treated as a

segment. This means that it is likely that the segment contains 2 or more unrelated subtopics that have been merged together and should therefore be split apart again.

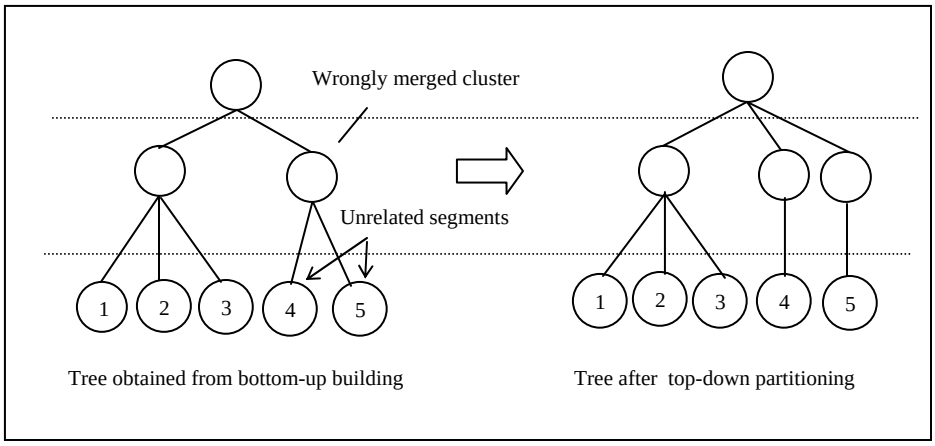

Figure 9: Top-down partitioning of tree to restore boundaries

In the figure above, segments 4 and 5 are wrongly merged together after the bottom-up building of the hierarchical tree. During the top-down partitioning phase, it is found that the segmentation cost of the merged segments is more than the global segmentation cost. The merged segment is then split apart again. The correct tree is thus the one on the right. Note that the segment formed from merging segments 1 and 2 remains untouched.

6 Concept testing of hierarchical clustering algorithm

Currently, there are no evaluation measures that exist in the literature which can be used to evaluate the accuracy of a hierarchical segmentation algorithm. As creating such a measure is out of the scope of this project, I have chosen to test the concept of my work by concatenating segments from different texts to form a hierarchical structure and then seeing if that structure can be recovered automatically. No score is assigned to a test case for partial success - either the algorithm successfully retrieves the structure, or it does not.

6.1 Hierarchical text corpus

In order to facilitate testing, I have created a corpus of documents by concatenating segments taken from scientific articles. The articles are taken from *Wikibooks* (http://en.wikibooks.org/) and are from the genres of Life Sciences, Chemistry and Physical Sciences.

Each of the genres is made up of numerous disciplines, e.g. Life Sciences is made up of the disciplines of biology, animal behavior, ecology etc. And each discipline are further divided into various sub-disciplines. For example, biology can be divided into evolutionary biology, neurobiology, microbiology etc. There is thus, an inherent hierarchy present.

An earthquake is a trembling or shaking movement of the Earth's surface.
Earthquakes typically result from the movement of faults, quasi-planar zones of deformation within its uppermost layers.
The word earthquake is also widely used to indicate the source region itself.
The solid earth is in slow but constant motion (see plate tectonics) and earthquakes occur where the resulting stress exceeds the capacity of Earth materials to support it.
The less frequent events that occur in the interior of the lithospheric plates are called intraplate earthquakes.

Biology is the science of life.
Biology encompasses a broad spectrum of academic fields that are often viewed as independent disciplines.
Biology can be understood at the level of interdependent populations and their habitats through ecology and evolutionary biology.
These include the comparisons of DNA sequences conducted within molecular biology.
The word "biology" in its modern sense seems to have been introduced independently by Gottfried Reinhold Treviranus.

Most non-scientists seem to be quite confused about precise definitions of biological evolution.
Such confusion is due in large part to the inability of evolutionists to communicate effectively to the general public and also to confusion among scientists themselves about how to define such an important term.
When discussing evolution it is important to distinguish between the existence of biological evolution and various theories about the mechanism of evolution.
And when referring to the existence of biological evolution it is important to have a clear definition in mind.
What exactly do biologists mean when they say that they have observed evolution or that humans and chimps have evolved from a common ancestor?

Figure 10: Sample document from corpus

Figure 7 shows a sample document taken from the corpus. Paragraph 1 is taken from an article about earthquakes, paragraph 2 from an article about general biology and 3 from an article about evolutionary biology. It can be seen immediately that 1 is unrelated to 2 and 3 whereas 2 and 3 are both subtopics of a larger topic (i.e. biology).

6.2 Experiments using the corpus

The hierarchical clustering algorithm when fed with the *linear* segments of the document shown in figure 7 should be able to retrieve the following structure (see figure 8 below):

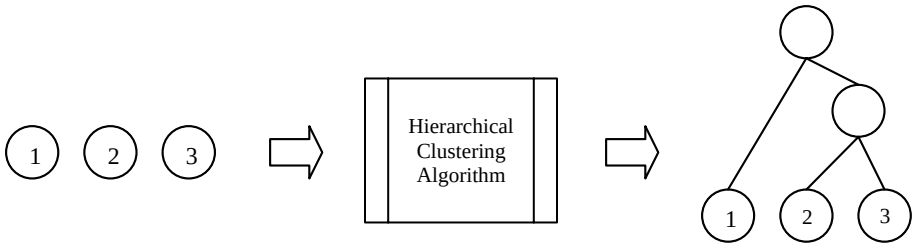

Figure 11: Linear segments to hierarchical tree

Any other tree structure recovered would be deemed to have failed. With this criterion, I have tested the algorithm on 30 documents in the corpus. Out of the 30 documents, 26 documents passed the test, giving a success rate of about 86.7%.

6.3 Limitations

In the above test cases, I have used artificially created documents with enough word repetition such that the hierarchical structures of the documents are immediately obvious. Though my algorithm works for such cases, it remains to be seen whether it can be applied to real world applications, where the documents being segmented might not have such a large number of repeated words. To show that the algorithm can also be applied to real world cases, an evaluation measure must be created that can assign a score for partial success. I leave the creation of such a measure for future work.

7 Conclusions

7.1 Summary

In this project I have accomplished the following objectives:

1. Showed that using lexical chaining in conjunction with word repetition improves linear segmentation accuracy.
2. Modified an existing *linear* segmentation algorithm to use both lexical chaining and word repetition such that it gives results on better than other algorithms in the literature.
3. Created a hierarchical clustering algorithm based on the PPM framework to achieve *hierarchical* text segmentation. To my knowledge, no other such algorithm exists in the literature.

7.2 Future Work

In this thesis, though I have introduced an algorithm to organize linear text segments into a hierarchical tree structure and have shown that the concept works, I have not provided a comprehensive evaluation metric to rate the effectiveness of the algorithm. To my knowledge, as very little research has been done on hierarchical text segmentation, no such metric has yet been created and it is beyond the scope of this project to provide one. Thus, perhaps future work could be devoted to providing just such a metric and to perhaps improve on certain aspects of the algorithm. Furthermore, a corpus containing real-world documents of a hierarchical nature needs to be compiled in order to facilitate testing.

References

[Allan 1998] James Allan, Jaime Carbonell, George Doddington, Jonathan Yamron, and Yiming Yang. *Topic detection and tracking pilot study: Final report.* In Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop, pages 194-218, February 1998.

[Barry, D. and Hartigan, J. 1992] D. Barry, J.Hartigan. *Product partition models for change point problems.* Ann. Statist. 20, 260--279.

[Beeferman 1997] Doug Beeferman, Adam Berger, and John Lafferty. *Text Segmentation Using Exponential Models.* In Proceedings of the Second Conference On Empirical Methods in Natural Language Processing, Providence, RI, pages 35-46, 1997.

[Beeferman 1999] Beeferman D., Berger A., Lafferty J. *Statistical Models for Text Segmentation.* Machine Learning, Vol. 34(1—3), pages 177-210, 1999.
[Beeferman and Berger 1997] D. Beeferman, A. Berger and J. Lafferty, A model of lexical attraction and repulsion, In Proceedings of ACL-EACL'97 Joint Conference, Madrid, Spain, 1997.

[Choi 2000] Freddy Y. Y. Choi. *Advances in Domain Independent Linear Text Segmentation* Proceedings of NAACL-00

[Chafe 1979] Wallace L. Chafe. *The flow of thought and the flow of language*. In Talmy Givón, editor, Syntax and Semantics: Discourse and Syntax, volume 12. Academic Press, New York, pages 159-182, 1979.
[CNN site] http://www.cnn.com/

[Della Pietra 1997] S. Della Pietra, V. Della Pietra, and J. Lafferty. *Inducing features of random fields*. IEEE Trans, on Patter Analysis and Machine Intelligence, Vol.

19(4): pages 380-397, April 1997.

[Engelbart and English 1968] D.C. Engelbart and W.K. English, *A Research Center for Augmenting Human Intellect*, in Proceedings of the Fall Joint Computer Conference, 33, AFIPS Press, Reston, VA, pages 395-410, 1968.

[Halliday and Hasan 1976] M. A. K. Halliday and R. Hasan. *Cohesion in English*. Longman, London.

[Hearst 1994] Marti A. Hearst. *Multi-paragraph Segmentation Of Expository Text*. In Proceedings of 32nd Annual Meeting of the Association for Computational Linguistics (ACL'94), Las Cruces, New Mexico, pages 9-16, 1994.

[Hearst 1997] M. A. Hearst. *TextTiling: Segmenting text into multiparagraph subtopic passages*. Computational Linguistics, vol. 23, pages 33-64, 1997.

[Ingrid Daubechies 1992] Ingrid Daubenchies, Rutgers University and AT&T Bell Laboratories, *Ten Lectures on Wavelets*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, 1992.

[Kan et al. 1998] Min-Yen Kan, Judith L. Klavans, Kathleen R. McKeown. *Linear Segmentation and Segment Significance*. Proceedings of the 6th International Workshop of Very Large Corpora (WVLC-6), Montreal, Quebec, Canada: Aug. 1998. pp. 197-205

[Kehagias et al. 2002] P. Fragkou, V. Petridis, Ath. Kehagias. *A Dynamic Programming Algorithm for Linear Text Segmentation.*

[Kehagias et al. 2002] P. Fragkou, V. Petridis, Ath. Kehagias. *Text Segmentation by Product Partition Models and Dynamic Programming*. Mathematical and Computer Modelling, 2002

[Mallat 1989] S. Mallat. Multiresolution Approximations and Wavelet Orthonormal

Bases of L2(R). Transactions of the American Mathematical Society, Vol. 315(1), pages 69-87, 1989.

[Nelson 1967] T.H. Nelson. Getting It Out of Our System, in Information Retrieval: A Critical View, G. Schecter, editor, Thompson Books, Washington, DC, pages 191-210, 1967.

[Raghavan 1986] V. V. Raghavan and S. K. M. Wong. A critical analysis of vector space model for information retrieval. Journal of the American Society for Information Science, Vol. 37 (5), pages 279-87, 1986.

[Reynar 1994] Jeffrey C. Reynar. An automatic Method of Finding Topic Boundaries. In Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics (ACL'94, student session), Las Cruces, New Mexico, pages 331-333, 1994.

[Salton 1983] G. Salton, McGill and M. J. *Introduction to Modern Retrieval.* McGraw-Hill Book Company, 1983.

[Salton 1986] Gerard Salton. *Automatic Text Processing* (Reprinted), Cornell University, Addison-Wesley Publishing Company, Inc., Reading, MA, 1989.

[Utiyama 2001] 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.