LOCAL CONTEXT BASED LINEAR TEXT SEGMENTATION

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTER ENGINEERING

AND THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Hayrettin Erdem

February, 2014

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. Fazlı Can (Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. Özgür Ulusoy

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assist. Prof. Dr. Gönenç Ercan

Approved for the Graduate School of Engineering and Science:

Prof. Dr. Levent Onural
Director of the Graduate School

ABSTRACT

LOCAL CONTEXT BASED LINEAR TEXT SEGMENTATION

Hayrettin Erdem
M.S. in Computer Engineering
Supervisor: Prof. Dr. Fazlı Can
February, 2014

Understanding the topical structure of text documents is important for effective retrieval and browsing, automatic summarization, and tasks related to identifying, clustering and tracking documents about their topics. Despite documents often display structural organization and contain explicit section markers, some lack of such properties thereby revealing the need for topical text segmentation systems. Examples of such documents are speech transcripts and inherently unstructured texts like newspaper columns and blog entries discussing several subjects in a discourse. A novel local-context based approach depending on lexical cohesion is presented for linear text segmentation, which is the task of dividing text into a linear sequence of coherent segments. As the lexical cohesion indicator, the proposed technique exploits relationships among terms induced from semantic space called HAL (Hyperspace Analogue to Language), which is built upon by examining co-occurrence of terms through passing a fixed-sized window over text. The proposed algorithm (BTS) iteratively discovers topical shifts by examining the most relevant sentence pairs in a block of sentences considered at each iteration. The technique is evaluated on both error-free speech transcripts of news broadcasts and documents formed by concatenating different topical regions of text. A new corpus for Turkish is automatically built where each document is formed by concatenating different news articles. For performance comparison, two state-of-the-art methods, TextTiling and C99, are leveraged, and the results show that the proposed approach has comparable performance with these two techniques. The results are also statistically validated by applying the ANOVA and Tukey post–hoc test.

Keywords: Text Segmentation, Topic Segmentation, Natural Language Processing, Lexical Cohesion, Semantic Relatedness.

ÖZET

YEREL İÇERİK TABANLI KONUSAL METİN BÖLÜMLENDİRME

Hayrettin Erdem
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Yöneticisi: Prof. Dr. Fazlı Can
Şubat, 2014

Metin dokümanlarında konusal yapının anlaşılması, etkili erişim ve arama, otomatik özetleme ve dokümanları konuları hakkında tanımlamak, biraraya getirmek ve takip etmek gibi görevler için önemlidir. Dokümanlar genellikle içerdiği bölümleri birbirinden ayıran başlıklar ve yapısal ayıraçlar içeriyor olsalar da, bazı dokümanlar bu özelliklere sahip değildir ve bu durum konu bakımından metin bölümleme sistemlerine olan ihtiyacı ortaya çıkarmaktadır. Konuşma verisinden elde edilen transkript metinler ve gazete, blog yazıları gibi konusal bakımdan yapısı belirsiz olan metinler, bu tür dokümanlara örnek olarak gösterilebilir. Metinlerde konu bölümlendirme için, yani metni kendi içerisinde tutarlı konusal bölümlere ayırmada, yerel içerik tabanlı ve kelimeler arasındaki ilişkilerden yararlanan yeni bir yöntem sunulmaktadır. Kelimeler arasındaki anlam bütünlüğünü ifade etmede, önerilen yöntem HAL anlamsal uzayından yararlanmaktadır. Bu uzay, metin içerisinde birlikte gözüken kelimelerin incelenip sabit uzunluktaki bir pencerenin metin boyunca kaydırılmasıyla oluşturulur. Önerilen algoritma olan BTS, konusal değişiklikleri döngüsel olarak tespit etmektedir. Her döngüde, cümlelerden oluşan bir blok ele alınarak, birbiriyle en ilişkili cümle ikilileri bulunur ve bu çiftlerin incelenmesiyle yeni bir bölüm oluşturulur. Önerilen yöntem, hata içermeyen haber bülteni transkriptlerinde ve yapay olarak farklı bölümlerin biraraya getirildiği dokümanlar üzerinde değerlendirilmektedir. Türkçe dili için, otomatik olarak haber metinlerinin kullanılmasıyla yapay bir veri seti oluşturulmuştur. Performans karşılaştırması için, TextTiling ve C99 yöntemleri kullanılmaktadır ve sonuçlar, önerilen yöntemin bu yöntemlerle karşılaştırılabilir olduğunu göstermektedir. Sonuçlar ayrıca, ANOVA ve Tukey testleri ile istatistiksel olarak doğrulanmaktadır.

Anahtar sözcükler: Metin Bölümlendirme, Konu Bölümlendirme, Doğal Dil İşleme, Kelime Bütünlüğü, Anlamsal İlişki.

Acknowledgement

I would like to thank my supervisor, Prof. Dr. Fazlı Can for his endless support and guidance during my research.

I am grateful to the jury members, Prof. Dr. Özgür Ulusoy and Assist. Prof. Dr. Gönenç Ercan for reading and reviewing this thesis.

I would like to thank all my friends but especially Çağrı Toraman, Onur Yorulmaz, Ahmet Çağrı Şimşek and Osman Kayataş for their encouragement, support and friendship.

Finally, I owe the most gratitude to my family for their endless love and support. I dedicate this thesis to them.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

In the last decades, with the huge advances in science and technology, any information can be stored in various content forms (text, audio, image, video etc.) as digitally, and consequently, there has been a vast quantity of information stored on various digital sources. Thus, searching for information among these massive amounts of data becomes a crucial task. The most common search is web search although search is applied to many other domains like corporations and government. Loosely speaking, search and retrieval is done as follows: People search for information by submitting queries to the desired system, and the system returns the list of relevant results in ranked order in response to corresponding queries. In this study, our focus is on text and text documents, and for effective retrieval, we need to understand the underlying structure of the documents and their contents. A document may contain a single topic or multiple topics, and it has a structure associated with its content or it can be relatively unstructured. Considering all these factors affects the ranking of that document as opposed to a query. Ranking becomes challenging when dealing with documents containing multiple topics. A specific query may only be related to some portion or section of text instead of its whole content. An ineffective system places such a document in a low rank

position by considering its whole content instead of its related portion. Ranking remains problematic when different terms of a query match with unrelated portions of such a document thus accounting for spurious matches. To circumvent these problems, documents containing multiple topics, whether they are structured associated with their content or relatively unstructured, are required to be decomposed into topically coherent sections and retrieval task is done accordingly. Since users are often interested in particular parts of a document, only the related sections can be showed to the users as opposed to a certain query instead of its whole content.

Segmentation is also appropriate as dealing with speech transcripts. Such documents are relatively long and contain uninterrupted streams of text. They can be obtained from monologues like news broadcasts, lectures and testimonies as well as from dialogues (between two or many people) like TV shows and business meetings. Topic boundary detection can be achieved by merely analyzing the original content of a certain non-text form (audio or video); however, boundaries may not always be truly detected. By incorporating lexical attributes with or without audio features (e.g. prosody, video scene changes, silence), more reliable boundaries can be discovered. Moreover, given a segmentation of a certain transcribed text, it is possible to classify the contents of each segment into topics. It enables later tracking of some predefined topic or event.

Transcripts are produced either by automatic speech recognition (ASR) systems or manually by human annotators. Despite notable advances in understanding and technology of automatic speech recognition systems today, transcripts are not yet guaranteed to be perfectly error-free thus accounting for word errors. Moreover, transcripts often do not have any structural information (i.e. no paragraph and section information, and even no clear sentence boundaries through the lack of any sentence segmentation system). Therefore, segmenting transcripts produced by ASR systems can be much harder than segmenting traditional text documents since they are explicitly structured more-or-less.

Segmentation of transcripts received attention by Topic Detection and Tracking (TDT) project [4]. In this project, datasets were produced which includes

both text news stories from newswire and audio recordings of radio and TV news broadcasts over 600 hours in English and Chinese. The project did not only address the problem of segmentation but also included tasks related to identify, cluster, track and link stories about their topics. It is remarkable that segmentation was the first necessary step for the start of the remaining tasks for spoken data.

Segmentation also aids to produce a good quality of summary which requires condensing the original content of a document while preserving its essential insights [5, 6]. Summarization is applied for documents typically containing multiple topics. To produce a quality summary, one needs to select the most relevant sentences for each section, and generate a summary which includes all topics that are mentioned in the document. Therefore, segmentation is required to identify individual sections of unstructured documents. Consequently, segmentation assists the task of summarization by discovering the underlying structure of documents.

1.2 Task Description

Text segmentation is about developing computational techniques to discover the topical structure of texts, and it is one of the main concerns in natural language processing. The task is to automatically divide documents with unknown topic structures into a linear sequence of topically-coherent adjacent sections thereby revealing the underlying organization of them. In other words, the objective is to detect boundaries (topic shifts) between individual segments of the document. Understanding the notions of topic and coherence is important in terms of the segmentation task.

The notion of topic (or theme) is defined as the aboutness of a unit of text (a discourse, a portion of text or a sentence) [7]. A discourse may contain portions of text about a single topic as well as multiple topics. For instance, if we are interested in segmenting news broadcast transcripts then each transcript contains

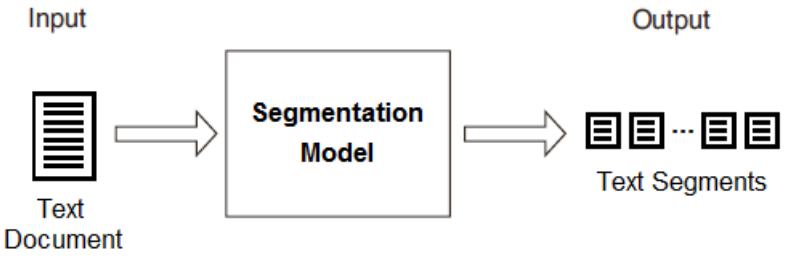

Figure 1.1: Text segmentation as the task of dividing a text into linear sequence of topical segments.

multiple topics each corresponds to an individual news story. On the other hand, if the text to be segmented is about a single main topic then we need to identify the subtopics related to it by inducing subtopic shifts. Units of text appeared in a certain segment determines the topicality of that segment. Consequently, units of text being about the same thing induce the notion of topic.

A well-written text has some underlying sense which is defined as coherence. In other words, coherence represents the semantic relationship among sentences or clauses appeared in a certain portion of text [8]. Coherence is typically provided by relations such as elaboration, support, cause or exemplification. However, identifying such relations is not straightforward, and it is computationally expensive. Therefore, coherence is typically found by searching cohesion in the text although they do not necessarily exist together. In other words, coherence may occur without cohesion. Still, cohesion is assumed to be a strong indicator of coherence. The term "cohesion" means that the text all sticks together [9]. Relations among text elements of any size without grammatical limitations exhibit cohesion in the text [10], and it is easier to detect such relations rather than coherence relations. Reference, conjunction and lexical cohesion are examples of cohesion relations [11]. Particularly, lexical cohesion is employed to reveal the individual segments for the task of segmentation, and it is simply represented as the semantic relationships among words appeared in the text. As a result, lexical cohesion can benefit the task of segmentation as segments typically display lexically homogeneous characteristic.

1.3 Segmentation Types

Documents typically display a hierarchical structure about their contents [12], and subtopics may occur within individual topic segments. As users might be interested in different levels of detail in text, setting the level of granularity as opposed to information needs of users would be a grateful aspect for any segmentation system. The task of hierarchical text segmentation addresses the problem of inferring a fine-grained (well-formed) structural organization for a given text whereas linear text segmentation deals with finding a coarse-grained segmentation of a given document. In other words, linear segmentation displays a simple picture about the structural organization of a document while hierarchical segmentation produces a more detailed depiction of it. Even people were often in conflict as determining the subtopic shifts [13, 14], automatically inducing multiple levels of topic hierarchy is an extremely difficult task. It is partly because the transitions between low-level subtopic regions tend to be more smooth than the shifts between high-level topics but it is also because exploiting word distribution information is not usually adequate by itself to determine the actual subtopic regions. Consequently, hierarchical topic segmentation is a more challenging task rather than linear text segmentation.

To the best of my knowledge, there has not been much research on hierarchical segmentation. One approach is to develop an unsupervised technique which employs agglomerative clustering over paragraphs to obtain resulting hierarchical segmentation [15]. Another methodology finds a topical hierarchy by training separate classifiers for topics and subtopics [16]. A recent approach incorporates multi-scale lexical cohesion information into a bayesian framework to induce a hierarchical topic structure [17]. Although a fine-grained segmentation is more desirable, most of the existing work has been focused on linear, coarse-grained segmentation. In this thesis, our focus is also on the task of linear text segmentation.

1.4 Contributions

The contributions of this thesis are the followings:

- Introduction of novel concepts regarding to term correlation and sentence relatedness
- Introduction and implementation of a novel local–context based segmentation method for the task of linear text segmentation
- Compilation of a new Turkish dataset which contains documents by randomly concatenating different news articles from BilCol[18] for the task of topical text segmentation
- Performance comparison of the proposed algorithm with the well-known methods, *C99* and *TextTiling* resulting as comparable performance with them

Chapter 2

Related Work

A particular topic is generally discussed by either using the same set of words and phrases or by using semantically related words and referents [8]. Therefore, the vocabulary is typically homogeneous within a particular topic. As moving to discuss another topic, a new set of words and phrases is encountered [19]. To benefit these arguments, one needs to typically identify lexical cohesion relationships in the text by either considering lexical distributional information or deeply analyzing semantic relationships among textual units in the discourse. Lexical repetitions and chains are examples about lexical distributional information. Semantic knowledge in the text can be extracted by solely analyzing the text to be segmented as well as through dictionaries and thesauruses. As a result, lexical cohesion strongly benefits the task of segmentation. Approaches leveraging lexical cohesion basically falls into two categories, namely, discriminative and generative approaches.

Discriminative approaches, as the name implies, differentiate text regions belonging to different topical segments by measuring the difference among the portions of the text. To this end, one requires a suitable similarity metric to computationally discriminate different topical regions. Another point is that such approaches either consider a small portion of text or the whole text as hypothesizing segments. For instance, one attempts to identify segment boundaries by measuring the difference between two adjacent windows as they are moving across

the text [2, 20]. On the other hand, another attempts to discover boundaries by finding the most likely segmentation due to its objective function as considering the whole content [21, 22]. Memory and CPU time requirements may be a decisive factor since the former needs less memory and CPU time rather than the latter. Another remarkable point is that local methods are more suitable than global ones as the streams of text received in an online way and the segmentation needs to be online accordingly [20, 23].

Generative approaches typically characterize lexical cohesion by estimating language models for topics, and inducing the most likely topic sequence based on the derived models given observed words thereby finding segment boundaries. One advantage of generative models is that they are capable of handling the problem of topic segmentation and identification as together. Specifically, they generate topic segments as well as topic models in terms of either language models or latent concepts [24, 25, 23, 26]. Then, these models can ease the task of classifying or clustering topics.

In addition to lexical cohesion, another auxiliary idea is to use boundary features as detecting individual topic segments. As moving to another topic or ending the discussion within a particular topic, there may exist features indicating those events. Such a well-known feature is cue words and phrases which include information about the semantics or structure of text [12, 27]. Cue phrases tend to vary across different domains; therefore, a suitable set of cue phrases can be obtained by examining a particular domain of interest. Another issue is that cue phrases can be extracted by either defining them manually or deriving them automatically. As cue phrases are typically domain-specific, automatic extraction of them [3, 28] is an important aspect. As a result, employing cue words and phrases in the text ease the segmentation task.

Given a set of boundary features like cue phrases and prosodic speech features (e.g. silence, speech overlap, speaker change), another tendency is to combine these features with lexical cohesion to obtain more reliable segmentations. Integrating such features can be achieved by using classifiers [13, 29, 30] or generative models [28, 31, 32].

Rest of the chapter is dedicated to explain some noteworthy studies based on lexical cohesion in more detail.

2.1 Discriminative Approaches

2.1.1 Dotplotting

Reynar adapts the dotplotting technique [33] to segment text into multiple coherent units [1]. A dotplot is a two dimensional matrix where both axes comprises the terms observed sequentially in the text. A cell of this matrix takes a value of one if both axes correspond to the same term, otherwise zero. It is obvious that the diagonal of the matrix is composed of ones.

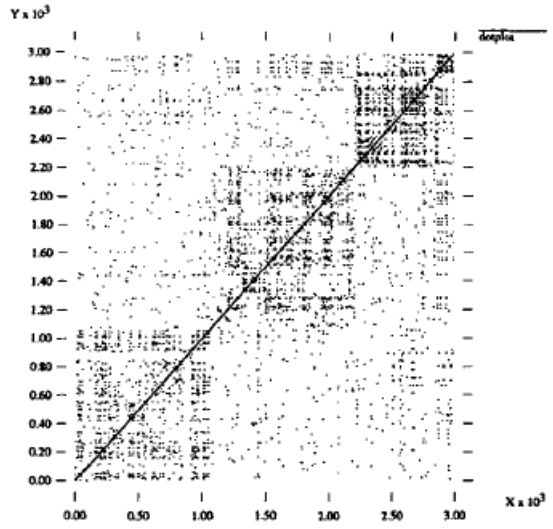

Figure 2.1: The dotplot of four concatenated *Wall Street Journal* articles [1].

At the beginning, the text is processed to eliminate stop words and the remaining terms are lemmatized. Dotplot is then created on these remaining terms. Finally, the dotplot is algorithmically analyzed to identify boundaries. In the dotplot, dark regions (high density regions) along the diagonal correspond to candidate segments. The density of a region is measured by the proportion of the

number of points located on it to the total area of it. Reynar proposes two related algorithms to discover boundaries by measuring the density of regions. The first one detects boundaries which maximize the density within regions along the diagonal where the second one identifies boundaries which minimize the density of regions outside of the diagonal. In more detail, the density of regions for all putative boundaries (sentence or paragraph boundaries) are calculated and sorted in either ascending or descending order depending on the selection of algorithm. Although both algorithms are similar, it is concluded that they do not necessarily yield the same results. Another issue is that the number of boundaries is unknown by the system thus given as input. The proposed technique is experimented on concatenated Wall Street Journal articles and evaluated in terms of precision and recall.

2.1.2 TextTiling

TextTiling [2] is one of the initial algorithms for this task and still considered to be one of the baseline algorithms for comparison of recent studies. Hearst proposes *TextTiling* to identify subtopic segments within single-topic expository texts. He states that expository texts are generally composed of lengthy paragraphs, each of which does not necessarily indicate the beginning of a new subtopic thus giving rise to the need of a system for segmentation. There exists three proposed algorithms within the same overall approach, one of which is the core algorithm (block comparison algorithm) and the other ones (vocabulary introduction and lexical chain methods) are alternatives that use different similarity metrics in their calculations.

The core algorithm consists of three main steps which are tokenization, similarity determination and boundary identification, respectively. In the tokenization step, the text is tokenized, excluded from stop words and stemmed. Pseudo sentences (token-sequences) are preferred over real ones to prevent normalization problems. The size for pseudo sentences (w) is set to be 20 tokens per token-sequence. Analogously, blocks of token-sequences are preferred over real paragraphs for the reason of irregular paragraph lengths leading to unbalanced

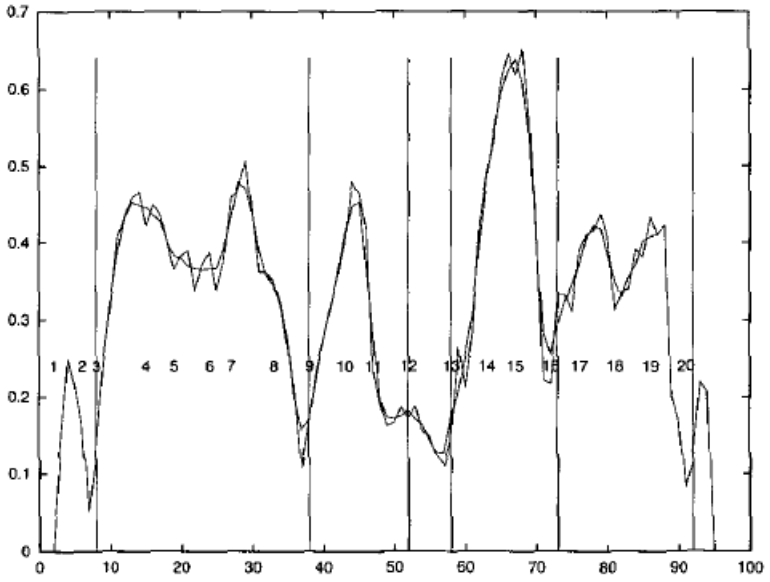

Figure 2.2: Token-sequence gap number vs. similarity scores for block-comparison algorithm illustrating both smoothed and unsmoothed versions. Vertical lines indicate hypothesized boundaries [2].

comparisons. The size of each block (k) is determined as the average paragraph length in terms of token-sequences (k = 6). Then, for each block, a lexical frequency vector is constructed, whose features are the unique tokens appeared in that block and values are simply their frequencies in it. In similarity determination step, a score for each adjacent pair of blocks is calculated as moving across the text (shifting blocks by one sentence at a time). This score is simply calculated by using the cosine similarity between the lexical vectors of the blocks. An earlier version of this algorithm uses tf.idf weighting [34] as weighting terms which comprise the values of lexical vectors, however, the results does not seem as good as those in which terms are weighted according to their within block frequency alone. Lastly, in boundary identification step, all computed scores are plotted against their gap number and for each gap, a depth score is calculated based on its nearest peaks on either side. However, the algorithm uses smoothing to eliminate small perturbations and flat valleys in the plot before computing depth scores to alleviate the problem of misleading boundary detection. Assuming that depth scores are normally distributed, the algorithm determines the number of segment boundaries according to some cutoff function using the average and

standard deviation of those scores. Then, depth scores are sorted and used to determine segment boundaries. Finally, gap points with the highest depth scores are chosen as the hypothesized boundaries.

For vocabulary introduction and lexical chains methods, all but the similarity calculation step in the core algorithm are identical. In the vocabulary introduction algorithm, the positions of newly introduced terms are stored. As in block comparison algorithm, a moving window is also used here to assign a score between consecutive blocks of tokens. The score is simply calculated by adding the number of newly introduced terms in both blocks. In lexical chains method, chains are created for each unique term which occurs more than once in the text. A score between each pair of adjacent blocks is computed as the number of active chains which spans the gap between them. A lexical chain for a term is active if that term appears within some distance threshold. After score computation for each gap, the identical boundary identification step is applied.

2.1.3 LCA based Segmentation

Ponte and Croft [35] tackle the same problem but with a non-ordinary test collection where segments are relatively small and within segment sentences often do not share any common words. In such cases, word repetition based approaches fail to segment the data correctly. To deal with such data, detecting semantically related words and phrases within topic sentences plays a crucial role. Thus, they make use of Local Context Analysis (LCA) [36] as an association thesaurus to measure the similarity between sentences. In more detail, they query each sentence to the LCA and it returns top m related concepts regarding to each corresponding sentence (m = 100). Then, pairwise sentence similarities are calculated by simply looking at the number of common concepts between every pair of sentences.

After similarity calculation, for each possible segment size, a score is assigned for each position in the text. For a given segment, this score has three components. The first component is internal similarity which is the sum of pairwise

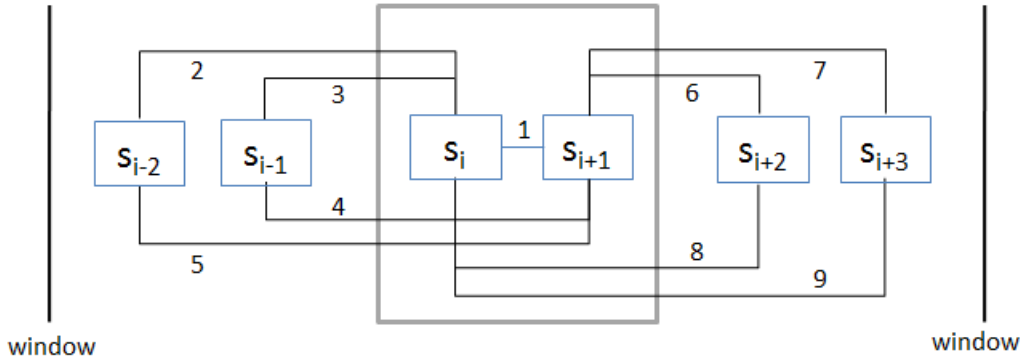

Figure 2.3: Score calculation for a candidate segment with numbered pairwise sentence similarities. Internal similarity solely corresponds to similarity #1. Left external similarity is the sum of similarity #2, #3, #4, and #5. Right external similarity is the sum of similarity #6, #7, #8 and #9.

sentence similarities within the segment. The second component is left external similarity which is the sum of pairwise similarities of each sentence within the segment to a fixed number of preceeding sentences. Finally, the third component is right external similarity. Similarly, it is the sum of pairwise similarities of each sentence in the segment to a fixed number of sentences following the segment. Given those components, the score of a segment is simply calculated by extracting two external similarities from internal similarity. Then, these scores are ranked for each position in the text and hypothesized segments are found by dynamic programming. Furthermore, through a Gaussian length model obtained from a training set of 85 segments, a weight score is also associated with each possible segment. It is shown that the length model improves the performance.

2.1.4 C99

Choi [37] extends the idea of Reynars dotplotting algorithm [1] by using the cosine similarity measure and a ranking scheme instead of exact word matches as creating dotplot (similarity) matrix. The proposed methodology begins with tokenizing sentences in the text. Uninformative words and punctuations are then eliminated from each sentence. Next, the remaining tokens are stemmed by Porter

stemming algorithm. The core step in *C99* is to create similarity matrix by computing cosine similarity for each pair of sentences in the text and applying the adapted ranking scheme to the matrix. It is noticed that cosine similarity values are unreliable for sentences in short length segments thus highlighting the need of a ranking scheme. Further, computing similarity for each pair of sentences is found to be inappropriate since a multi-topic document typically contains a varying language. After filling the similarity matrix with cosine scores, the ranking scheme simply replaces each value in the matrix by its rank, which is the proportion of the number of neighboring elements with lower similarity value to the total number of neighboring elements according to some predefined mask size. The final step is to determine segment boundaries. For this purpose, Choi exploits the maximization algorithm proposed by Reynar.

The number of boundaries is automatically determined by observing the differentiation in total inside density of current segments and segments with the inclusion of a newly hypothesized segment. For this reason, some threshold function based on mean and variance of inside density differentiations is defined. A large difference indicates that optimal number of segments is reached.

C99 is experimented on samples of concatenated documents from Brown corpus. It is concluded that *C99* outperforms both *Dotplotting* and *TextTiling* techniques. A final note about *C99* is that it is still considered to be one of the state-of-the-art algorithms for this task.

2.1.5 LCSeg

Galley et al. [13] present two alternative solutions to the segmentation problem. The first solution (*LCSeg*) is exclusively relies on lexical cohesion where the second solution integrates conversational features (e.g. cue phrases, silences, speaker overlaps) with lexical cohesion information by a probabilistic classifier (*C4.5*). In both alternatives, common preprocessing steps (tokenization, stop word elimination and stemming) are applied for each text as an initial step.

LCSeg defines lexical cohesion as identifying lexical chains (simply consisting of term repetitions) occurred in the text to be segmented. The insight behind this definition is that these chains begin and end at topic boundaries thus helping to induce the topical structure of the text. The first step is thus to identify all lexical chains appeared in the text. To avoid weakly linked chains, chains that have a hiatus of h consecutive sentences with no appearance of its term are divided into subchains (experimented as h = 11). The second step is to weight each identified chain by defining a metric similar to well known tf-idf [34] measure. Weighting score of a chain is a product of its frequency and compactness values. Here, frequency refers to the number of repeated terms for a particular chain. Compactness is about the length of the chain, and shorter chains receive a higher score than longer ones. It is defined as the logarithm of the proportion of the text length to chain length in terms of number of sentences. After assigning a weighting score to each identified chain, the algorithm computes cosine similarity between two adjacent windows of fixed size k in terms of sentence breaks or speaker turn breaks (experimented as k = 2). Here, the similarity metric uses weighting score of overlapped chains with windows instead of simply using word count information over those windows. The last step is then to plot the similarity scores against sentence breaks and search for local minima breaks by computing the depth of each such break. To avoid misleading boundaries, the plot is smoothed by using a moving average filter. Finally, the algorithm hypothesizes boundaries with the highest depth scores with the knowledge of number of boundaries. As there is no knowledge about the number of boundaries, the algorithm estimates it by considering a threshold function based on the mean and variance of all potential boundaries. Then, the boundaries above that threshold are selected as the hypothesized boundaries. *LCSeg* is compared to two state-of-the-art algorithms, *U00* and *C99*. Those systems are evaluated on both synthetic (Brown, TDT and WSJ) and real data sets (ICSI Meeting corpus). As a result, *LCSeg* shows comparable performance to the state-of-the-art algorithms.

In the second solution, segmentation is considered as a classification problem where each break corresponds to a topic boundary or not. As classifiers, *C4.5* and *C4.5rules* are chosen. The general strategy is to initially create an unpruned

decision tree by *C4.5* and then analyze it by *C4.5rules* to obtain the most useful subset of pruned production rules. As few labeled meeting samples available, a specific set of features is chosen instead of using a large set of features to avoid overfitting the known data. These features are cue phrases, silences, speaker overlaps, speaker changes and lexical cohesion information gathered from LCSeg. The classifier is evaluated merely on the meeting corpus by performing 25-fold cross validation. In general, it is observed that lexical cohesion has more importance than other features in identifying topic boundaries. For comparison, *LCSeg* and *U00* are chosen and *C4.5* outperforms these methodologies.

2.1.6 InfoSimba

Dias et al. [38] propose a language-independent unsupervised solution to the problem. They state that word frequency is not the only indicator of the importance of a word in a document. Thus, they propose three new heuristics to define the importance of a word. Defining those heuristics is the initial step of their methodology. The first heuristic is the well-known tf.idf measure [34]. It calculates the relevance of a word in a document in terms of its frequency and its distribution across a collection of documents. However, not all relevant words are useful for segmentation. In other words, it is important to find topic-relevant words which helps to segment the text into coherent topics. For that purpose, they introduce a new heuristic called tf.isf. It evaluates each word based on its frequency within a specific sentence and its distribution across all the sentences within the document. However, tf.isf does not have a closer look on the distribution of a word in a document. More specifically, it is also important to evaluate whether a word appears in consecutive, near-consecutive or distant sentences. For that reason, they propose a new density measure to compute the degree of closeness between the appearances of a word in a document. Eventually, the importance of a word can be thought as the weight of that word which is computed through the combination of those heuristics.

The next step is to determine similarities between a focus sentence and its neighboring groups of sentences. Cosine similarity can be used for that purpose,

however, it fails when two sentences do not share any common word but they are semantically related. Therefore, they propose a new similarity measure called InfoSimba. It is the integration of weight and cohesiveness information into the cosine formula. The degree of cohesiveness between two words is represented by Equivalence Index Association measure.

The final step is to detect topic boundaries. The idea is to determine whether the focus sentence is more similar to the previous block of sentences or to the following block of sentences. Therefore, a preference score is computed for each focus sentence. If it is positive, the focus sentence is more similar to the preceding block of sentences, otherwise, it is more similar to the following block of sentences. If there exists a downhill between the scores of consecutive sentences, the presence of a new topic is identified. However, not all downhills indicate the presence of a topic. Indeed, deeper ones indicate more correct topic shifts thereby identifying them is important. Deeper downhills are recognized through a threshold, which is a function of average and standard deviation of downhill depths.

2.2 Generative Approaches

2.2.1 Hidden Markov Model Segmentation

Yamron et al. [39] introduce an approach based on Hidden Markov Models (HMMs) and language models to extract story boundaries and retrieve stories related to a specific event. Here, the hidden states are topics and the observations are words or sentences. The main idea is to treat a story as an instance of some underlying topic, and to model a text stream as an unlabeled sequence of these topics. The first step is to construct topic clusters along with their associated smoothed unigram language models. It is done by clustering a set of training texts using Kullback-Leibler distance metric. The selected clustering methodology is multi-pass k-means algorithm (k = 100). Instead of learning full set of topic transition probabilities, finding the probability of changing between topics is sufficient. It can be easily estimated by using the number of segments in the

training data. After constructing the model, an unbroken text is processed sentence by sentence at a time. In this model, finding topic boundaries correspond to finding topic transitions. In other words, segmentation and topic assignment are done simultaneously. They evaluate the methodology on the Topic Detection and Tracking (TDT) Pilot Study Corpus which consists of broadcast news stories.

Blei and Moreno [40] form an Aspect Hidden Markov Model (AHMM) by integrating Hoffman's aspect model, which is a probabilistic latent concept model (PLSI), into the HMM framework. They experiment their approach on unbroken streams of New York Times articles and noisy radio speech transcripts.

2.2.2 U00

Utiyama and Isahara [41] propose an unsupervised generative approach to the segmentation problem. The text to be segmented is a sequence of topics and each topic has a specific language model associated with it. Given these compact models, the ultimate goal is to find a segmentation (a set of boundaries) which maximizes the likelihood of the data. In other words, the maximum-probability segmentation is found by comparing likelihood values of different segmentations via dynamic programming.

Likelihood of the data is computed as multiplying the posterior probability of words given the segmentation, $P(W|S)$, by the prior probability of the segmentation, $P(S)$. This posterior probability solely depends on the selected model(s). In the proposed model, it is assumed that different topics are statistically independent of each other. Another assumption is that the words within a topic are also independent of each other. Under these assumptions, the posterior probability $P(W|S)$ corresponds to the multiplication of individual posterior probability of words given each segment, $P(W_i|S_i)$, which is a product across the probability of each word occurred within segment S_i. The probability of a word observed within a particular segment is computed by employing the Laplace law (Laplacian smoothing). The only prior information to estimate $P(S)$ is the description length which gives larger values to the segmentations having less number of segments.

Consequently, the algorithm selects the best segmentation which maximizes the likelihood of the data via dynamic programming. Another issue is that the algorithm automatically determines the number of boundaries. Yet, the number of boundaries can be given as input to the system. Estimated number of segments by the algorithm is relatively stable as opposed to the variation in the text length.

For experimentation, artificial dataset created by Choi is used. It is statistically shown that the proposed system (*U00*) is more accurate than *C99*. The proposed system can handle the segmentation of documents with varying domains which makes it applicable to be used for summarization and retrieval purposes.

2.2.3 BayesSEG

Eisenstein and Barzilay [32] introduce a Bayesian approach (*BayesSEG*) to unsupervised topic segmentation by generalizing the model proposed by Utiyama and Isahara (*U00*). Further, they examine the impact of cue phrases by integrating the proposed cue phrase model into the Bayesian framework. The general objective is to maximize the overall observation (data) likelihood thereby resulting to a cohesive segmentation. In the study of Utimaya and Isahara, one particular smoothed language model is estimated for each segment. In this study, however, each model is estimated by marginalizing (integrating) over all possible language models, and by leveraging Dirichlet Compound Distribution (DCM). After defining marginalized models for each segment, high likelihood segmentation is inferred via dynamic programming. In the proposed model, it is assumed that there is no prior information about segmentations therefore treating each prior probability regarding to a possible segmentation as uniform. To examine the effectiveness of cue phrases, a modified approach is proposed which claims that some of the text is topic-specific whereas other terms are topic-neutral (i.e. cue phrases). For this purpose, a cue phrase model shared by among all documents is derived by using a symmetric Dirichlet prior. Then, the likelihood calculation is varied only for sentences at segment boundaries. For a segment-initial sentence, the likelihood is computed by leveraging both models integrated into the Bayesian framework. The proposed approach both with and without cue phrases is evaluated on text

and transcribed speech datasets. For textual corpus, they create documents each of which is a chapter from some medical textbook. The task is to segment chapters into coherent sections defined by the author. For speech, they exploit ICSI corpus of meeting transcripts. The proposed system is compared against *U00*, *LCSeg* and *MinCut* [21] systems. *BayesSEG* assumes that the number of desired segments is provided as input. The introduced approach both with and without cue phrases outperforms compared systems. It is observed that cue phrases improve the results on the meeting corpus but not on the textbook corpus thus implying that effectiveness of cue phrases depends on consistent usage of them.

2.3 Overview

Approaches based on lexical cohesion are typically distinguished from each other by the way of how they quantify cohesion and search for hypothesized segmentations.

As mentioned earlier, lexical cohesion can be characterized by employing lexical repetitions, lexical chains as well as semantic knowledge obtained from either solely considering the local context or a specific collection, or using knowledge and concept sources like dictionaries and thesauruses.

The search for segmentation is described by answering the following questions: Where to search the segmentation, and how to find it over there. The first question is related to search space, and it is generally the space of all putative segmentations for a given document. The latter is about the search technique which is often a greedy or a dynamic programming (DP) technique. Greedy approaches look for the segmentation which has the local minimum or maximum of some specified cohesion function (metric) as searching the space of segmentations. DP approaches, on the other hand, seek for the segmentation which has the global optimum due to some predefined objective function. However, the drawback is that such approaches are computationally expensive.

As the language may vary among different domains of interest, existing approaches typically used stemmers of respective languages. Some approaches also employ external linguistic resources so that they can extract semantic knowledge among units of text. Supervised approaches need a suitable number of training samples to avoid underfitting or overfitting the data. Those approaches may not present a universal solution for all domains since there may not be adequate sample data in some domains. Therefore, a domain-independent solution is more desirable. Lastly, some techniques can predict the number of desired segments instead of getting additional external knowledge about it. However, estimating the correct number of segments is a challenging task.

Table 2.1: Overview of the selected related work on text segmentation.

Work	**Lexical Cohesion Indicator**	**Search Technique**	**Search Space**	**Linguistic Resource**	**Learning Model**	**Segment Count Estimation**
Dotplotting [1]	Term Repetition	Divisive Clustering	Segmentations	Stemmer	Unsupervised	Manual
TextTiling [2]	Term Repetition	Greedy	Segmentations	Stemmer	Unsupervised	Automatic
LCA [35]	Semantic Knowledge	Dynamic Programming	Segmentations	Stemmer, LCA	Unsupervised	Automatic
C99 [37]	Term Repetition	Divisive Clustering	Segmentations	Stemmer	Unsupervised	Automatic
LCSeg [13]	Lexical Chains	Greedy	Segmentations	Stemmer	Unsupervised	Manual, Automatic
InfoSimba [38]	Lexical Distribution	Greedy	Segmentations	None	Unsupervised	Automatic
HMM [39]	Language Models	Greedy	Topic Clusters	Stemmer	Supervised	Automatic
U00 [41]	Language Models	Dynamic Programming	Segmentations	Stemmer	Unsupervised	Automatic
BayesSEG [32]	Language Models	Dynamic Programming	Segmentations	Stemmer	Unsupervised	Automatic
Our Work	HAL	Greedy	Segmentations	Stemmer	Unsupervised	Automatic

Chapter 3

Proposed Approach

This chapter describes the local context based approach proposed for the task of linear text segmentation. The initial step is to pre-process the document to be segmented. The following step is then to build HAL vectors for the document terms based on their co-occurrence in the document. The final step is to apply the segmentation algorithm, *BTS*, which iteratively discovers segment boundaries based on sentence relevancies by considering a certain block of consecutive sentences.

3.1 Pre-processing

As for any NLP task, pre-processing is also an essential stage for segmenting text. It loosely aims to divide text into linguistically meaningful units, and provide a lexical analysis of a given raw text. The level of analysis depends on the application. For our task, it aims to extract vocabulary of terms and sentences as well as learning referenced segment boundaries for a given raw text document. Pre-processing includes the following steps: tokenization, stopword elimination and stemming, respectively.

Tokenization is the process of splitting character streams into tokens. In our

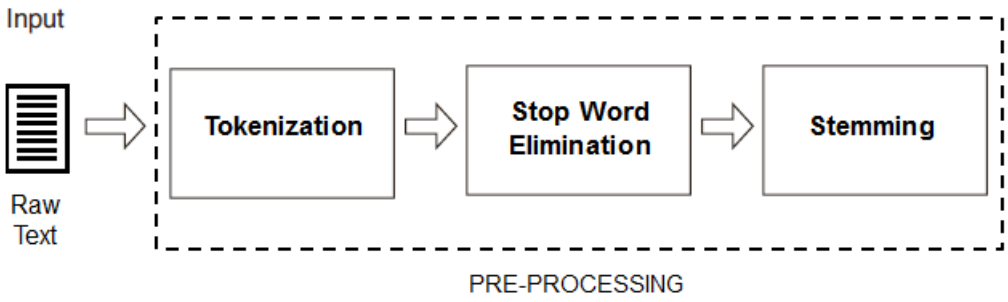

Figure 3.1: Pre-processing steps for text segmentation.

task, a particular raw text is chopped on white space to obtain its tokens. Non-alphabetical characters (e.g. punctuations, numbers) are then removed from each token.

Stopword elimination is the task of discarding words which are frequently appeared in texts and often carry little semantic meaning about them. For this purpose, stopword lists for Turkish and English are exploited. In more detail, Turkish stop list [18] contains 216 words where English stop list [37] includes 322 words.

	Examples of Stopwords
English	about, because, the, a, during, either, for, in, only, not, other
Turkish	ama, bu, ayrıca, kadar, olarak, ve, veya, hem, herhangi, zaten

Table 3.1: Examples of stopwords for English and Turkish.

Stemming, in one word, is reduction. By definition, it is to reduce words in different forms to their stem or root form. Roughly speaking, words are typically in either derivational or inflectional form through the addition of affixes into their base. Derivation often tends to alter the meaning of a word while inflection does not result in such difference. Stemming enables reliable term comparisons by providing the base of words instead of dealing with such different forms. In our experiments, we exploit Porters stemmer [42], the most commonly used algorithm for English, and Zemberek [43] stemmer for Turkish.

3.2 Building HAL Vectors for Document Terms

Semantic information among textual units can be extracted either by analyzing a set of documents (i.e. a text corpora) [44, 11, 45] or by exploiting an external linguistic resource such as WordNet [46]. Following the first notion, our aim is to capture semantic information, or deriving term relationships, by only analyzing the local context of the interested document. For this purpose, we employ HAL (Hyperspace Analogue to Language) [47, 48] to induce relationships among terms of the document to be segmented. HAL is a high-dimensional semantic space in the sense that each word in the text is represented by an individual vector (i.e. HAL vector) that contains co-occurrence information in terms of other words in the text. More generally, HAL leverages lexical co-occurrence information to examine relationships between document terms.

Looking closer, as building HAL vectors for document terms, a two-dimensional co-occurrence matrix (C) is formed by passing a window of fixed length over the text one word at a time. Each dimension of the matrix comprises unique words seen in the text, and each cell stores an accumulated count of co-occurrence score of the respective word pair. As moving a fixed-length window across the text, words appeared in each window are assumed to be correlated with each other by a score inversely proportional to the distance separating them. Given a text, at the very first window, co-occurrence scores are computed for each possible pair of words, (w_i, w_j), positioned at i and j such that w_j precedes w_i in that window (i.e. $i > j$). The number of pairs satisfying this condition is therefore $(windowSize \times (windowSize - 1)/2)$. The co-occurence score of a word pair (w_i, w_j) is then computed as follows:

$$cs(w_i, w_j) = windowSize - (i - j) \tag{3.1}$$

Any such word pair receives a maximum score of $windowSize - 1$ if there is no word separating the words in it, and a minimum score of 1 if the words in that pair resides at the two ends of the window.

After analyzing the first window, $C(w_i, w_j)$ values are then updated accordingly given these scores, $cs(w_i, w_j)$. At each remaining window, score calculation is done for word pairs in which the last word in each window is paired with each other word in that window. Then, respective cells in the matrix are accumulated regarding to these scores. The number of pairs formed at each window except the initial one is $(windowSize - 1)$. Note that the number of windows is $(N - windowSize + 1)$ where N is the total number of words in the text.

	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8
w_1								
w_2	5			1	2	3	4	5
w_3	4	5						
w_4	3	4	5					
w_5	2	3	4	5				
w_6	1	2	3	4	5			
w_7		1	2	3	4	5		
w_8			1	2	3	4	5	

Table 3.2: An example co-occurrence matrix built for the processed text "metasearch engin architectur queri broadcast multipl web search engin" where w_1: metasearch, w_2: engin, w_3: architectur, w_4: queri, w_5: broadcast, w_6: multipl, w_7: web, w_8: search after analyzing all windows. Empty cells in the matrix indicate a zero value.

Table 3.2 shows an example co-occurence matrix built for the text, "In a metasearch engine architecture, the query is broadcast to multiple web search engines.", for a given window size of 6 *after analyzing all windows*. As removing stop words and applying Porter's stemming, the text is now "metasearch engin architectur queri broadcast multipl web search engin". The co-occurrence matrix is actually built for this processed text. In this example, there are four windows in total, whose start and end word positions are $1 - 6$, $2 - 7$, $3 - 8$ and $4 - 9$. Table 3.3 shows the matrix *after analyzing the first window*.

In the matrix, each column is a HAL vector that holds co-occurrence information following a word while each row is also a HAL vector that stores such information preceding a word. Therefore, co-occurrence information regarding to a word is stored in two separate vectors by considering word ordering. However, order of words is not important for us. Thus, for a given word, its HAL vector is

	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8
w_1								
w_2	5							
w_3	4	5						
w_4	3	4	5					
w_5	2	3	4	5				
w_6	1	2	3	4	5			
w_7								
w_8								

Table 3.3: An example of co-occurrence matrix built for the processed text "metasearch engin architectur queri broadcast multipl web search engin" where w_1: metasearch, w_2: engin, w_3: architectur, w_4: queri, w_5: broadcast, w_6: multipl, w_7: web, w_8: search after analyzing the first window which starts with w_1 and ends with w_6. Empty cells in the matrix indicate a zero value.

simply found by adding up its row and column vectors. For example, the HAL vector of the term "w_7 : web", is estimated by adding up its row vector, [0 1 2 3 4 5 0 0], and its column vector, [0 4 0 0 0 0 0 5]. Therefore, HAL vector of the term is [0 5 2 3 4 5 0 5].

For each HAL vector, feature reduction is done by simply setting a threshold at the mean of the its co-occurrence scores, and features below this threshold are discarded from the respective vector. The remaining features for a vector is called *quality properties* of that vector. As considering the same example, for the term, "w_7 : web", mean of the scores is 3 and its HAL vector now becomes [5 3 4 5 5] with the features "engin, queri, broadcast, multipl, search", respectively.

The next and final step following feature reduction is to normalize each HAL vector by the sum of its feature values so that the sum of the scores is 1. Moving on the same example, the normalized HAL vector for the term, "w_7 : web", is [5/22 3/22 4/22 5/22 5/22] which corresponds to [0.23 0.13 0.18 0.23 0.23] with the features "engin, queri, broadcast, multipl, search", respectively.

Feature	engin	queri	broadcast	multipl	search
Value	0.23	0.13	0.18	0.23	0.23

Table 3.4: HAL vector for the term, "w_7 : web"after feature reduction and normalization.

3.3 Concepts of Correlation and Relevancy

We define new concepts regarding to how to measure the correlation and relevancy among text units.

TC(t_i, t_j) gives the term correlation between t_i and t_j by exploiting their HAL vectors as follows:

$$TC(t_i, t_j) = \begin{cases} 1 & \text{if } t_i = t_j \\ max(HAL_{t_i}(t_j), HAL_{t_j}(t_i)) & \text{otherwise} \end{cases} \tag{3.2}$$

TSR(t_i, s) calculates the term–sentence relevancy between term t_i and sentence s as follows:

$$TSR(t_i, s) = \begin{cases} 1 & \text{if s contains } t_i \\ \sum\limits_{t_j \in s} TC(t_i, t_j) & \text{otherwise} \end{cases} \tag{3.3}$$

SR(s_1, s_2) measures the sentence relevancy between sentences s_1 and s_2 by normalizing the total TSR relevancy by the number of non-zero TSR relevancy values.

$$SR(s_1, s_2) = \frac{\sum\limits_{t_i \in s_1} TSR(t_i, s_2)}{\sum\limits_{t_i \in s_1} [TSR(t_i, s_2) > 0]} \tag{3.4}$$

3.4 Segmentation Algorithm

The proposed algorithm, *BTS* (Block-based Text Segmentation), produces the segmentation for a given text by following the greedy strategy which iteratively forms a new segment by considering a certain block of consecutive sentences. In

other words, it makes local decisions about forming the contents of a segment by only examining the sentences within a particular block. Such an approach is particularly appropriate as the streams of text received online and has less computational cost as compared to the globally informed methods. The computational complexity of the algorithm is *O(mn)* where m is a constant which indicates the number of blocks and n is the block size in sentences.

The algorithm takes the document to be segmented and the block size as input. Block size determines the number of consecutive sentences in a certain block. The output of the algorithm is clearly the list of predicted segment boundaries. It is important to note that such boundaries correspond to the beginnings of predicted segments in terms of sentence ids. For each document, id of the initial sentence is one while the id of the last sentence is the total number of sentences in that document. The algorithm requires that input document is preprocessed and HAL vectors for document terms are built before running it.

The algorithm begins with creating the following empty lists to be utilized as predicting boundaries: *hypBoundaries*, *sentencePairs*, *relevancyScores* and *segmentContent*. The list *hypBoundaries* holds predicted segment boundaries by the algorithm. The list *sentencePairs* contains pairs of sentences formed by examining a certain block of consecutive sentences. Such pairs are formed by storing each sentence in a block along with the most related sentence to it in that block as a pair. The list *relevancyScores* holds relevancy scores associated with the pairs stored in *sentencePairs*. Lastly, the list *segmentContent* includes the ids of the sentences in a certain predicted segment.

The algorithm then proceeds as initializing the beginning sentence id of the first block as one ($blockStart = 1$), and the endmost sentence position as the id of the last sentence in the document, which is the number of sentences in the document. Next, the algorithm iteratively forms a new segment by examining a block of consecutive sentences. Each iteration starts with adding the beginning sentence id of a qualified block into the list of predicted boundaries. For a block to be qualified, its beginning sentence id should be less than the id of the last sentence in the document. If the block is not qualified, then the algorithm terminates by

returning the list of hypothesized boundaries (*hypBoundaries*). Intuitively, the first block is explicitly qualified since it begins with the first sentence in the document. A predicted segment contains a minimum number of two sentences and a maximum number of sentences that a block holds at maximum, which is the block size given as input. The pseudocode of the algorithm is shown in Figure 3.2.

An iteration of the algorithm (lines #8 – 47 in the pseudocode) proceeds as applying the following steps:

Step-1: Insert the beginning sentence id of the block into the list of predicted boundaries since it indicates the beginning of a new segment (line #8).

Step-2: Determine the end of the block. In other words, find the id of the last sentence to be included into the block (lines #9 – 12).

Step-3: Form sentence pairs and store each such pair associated with its relevancy score in the respective lists, *sentencePairs* and *relevancyScores*. More specifically, form sentence pairs by storing each sentence s_i in the block with the most relevant sentence s_j in that block where $i \neq j$. The relevancy between two sentences is computed by the *SR* function defined in Section 3.3. The most relevant sentence s_j to s_i is the sentence where *SR* gives the highest value for the pair (s_i, s_j) among all possible pairs. Then, also hold the relevancy scores in the list *relevancyScores* for the pairs in *sentencePairs* (lines #13 – 17).

Step-4: Eliminate sentence pairs which do not satisfy the threshold condition. Assuming that the relevancy scores of the sentence pairs are normally distributed, the threshold is based on the mean (μ) and standard deviation (σ) of the scores. More specifically, it is the difference of the mean and deviation ($\mu - \sigma$). Then, pairs having scores under this threshold are eliminated from the list of *sentencePairs* (lines #18 – 20).

Step-5: Form a new segment by considering the list of sentence pairs: Examine each sentence and its most related sentence in the list of *sentencePairs* (lines #21

– 45). This step starts with initializing the first contents of the segment. For this purpose, retrieve the first pair(s_i, s_j) from the list *sentencePairs*, and built the initial content of the segment by inserting sentence ids between the beginning sentence id of the block (*blockStart*) and maximum of the sentence ids, i and j. If the content of the segment contains the id of the last sentence in the document (*endmost*), then do not search new content for this segment, and go to *Step-6*. Otherwise, first retrieve the next sentence pair (s_i, s_j) from the list *sentencePairs*, and then iteratively search new content for the segment being currently built. An iteration of this process then proceeds as follows (lines #33 – 44):

- If the content of the segment contains the id of the most related sentence to s_i — the most related sentence is s_j and its id is j — then add sentence ids between the last element of content and the id of the currently examined sentence s_i, if possible (lines #33 – 39).
- Retrieve the next sentence pair (s_i, s_j) from the list *sentencePairs*. If there is no such pair left to be retrieved from the list *sentencePairs*, then do not search any new content anymore — go to *Step–6* — since all pairs have been examined. Otherwise, go back to the previous iteration step of searching new content for the segment (lines #40 – 44).

Step-6: Set the beginning of the new block to be considered in the next iteration by adding one to the last element contained in the list *segmentContent* (line #46).

Step-7: Clear the contents of the lists to be reused in the next iteration (line #47). That is, clear out the contents of *sentencePairs*, *relevancyScores* and *segmentContent*.

Step-8: Go back to *Step–1* if the beginning of the new block qualifies. That is, if it is less than the last sentence id of the document, then go back to *Step–1*; otherwise stop iterating (line #48).

Algorithm *BTS* (Document *d*, Integer *blockSize*)

Require: Preprocess *d* and construct HAL vectors for terms in *d*

Create a list *hypBoundaries* to hold predicted segment boundaries
Create a list *sentencePairs* to hold most related sentences in a certain block
Create a list *relevancyScores* to hold relevancy scores of sentence pairs in a certain block
Create a list *segmentContent* to hold contents of a certain predicted segment
blockStart ← 1
endmost ← number of sentences in *d*
repeat
    Add *blockStart* to *hypBoundaries*
    blockEnd ← *blockStart* + *blockSize* – 1
    if *blockEnd* > *endmost* **then**
        blockEnd ← *endmost*
    end if
    for each sentence s_i in the current block *B* **do**
        Find the most relevant sentence s_j in *B* to s_i ($i \neq j$)
        Add pair (s_i, s_j) to *sentencePairs*
        Add relevancy score of the pair (s_i, s_j) to *relevancyScores*
    end for
    Compute mean (μ) and standard deviation (σ) of *relevancyScores*
    relevancyThreshold ← (μ - σ)
    Eliminate pairs from *sentencePairs* with score less than *relevancyThreshold*
    Retrieve the first pair (s_i, s_j) from *sentencePairs*
    farIndex ← max (*i*, *j*)
    for *k* ← *blockStart* to *farIndex* **do**
        Add *k* to *segmentContent*
    end for
    searchContent ← *true*
    if *segmentContent* contains *endmost* **then**
        searchContent ← *false*
    else
        Retrieve the next pair (s_i, s_j) from *sentencePairs*
    end if

```
32:     while searchContent
33:         if segmentContent contains j then
34:             nextItem ← (Last element in segmentContent) + 1
35:             while nextItem <= i
36:                 Add nextItem to segmentContent
37:                 nextItem ← nextItem + 1
38:             end while
39:         end if
40:         if there is no remaining pair left to be retrieved from sentencePairs then
41:             searchContent ← false
42:         else
43:             Retrieve the next pair (si, sj) from sentencePairs
44:         end if
45:     end while
46:     blockStart ← (Last element in segmentContent) + 1
47:     Clear the contents of sentencePairs, relevancyScores, segmentContent
48: until blockStart >= endmost
49: return hypBoundaries
```

Figure 3.2: Pseudocode for the proposed segmentation algorithm, *BTS*.

Chapter 4

Evaluation Measures

This chapter presents methods for quantitatively evaluating and comparing the performance of different segmentation techniques. The initial attempt was to employ the same measures derived for evaluating performance in information retrieval and various classification systems. Despite those standard metrics were used in earlier analyzes, it was later noticed that such measures were inconvenient for the task of segmentation, and some alternative measures were proposed for more reliable assessments. Thus, evaluation metrics for this task basically fall into two categories, namely, classification-based and segmentation-based measures.

4.1 Classification-Based Measures

A typical evaluation based on binary classification is to count the number of correctly and incorrectly predicted class labels for instances given ground truth (or gold standard) information about their classes. A more detailed analysis is further possible through such knowledge, for instance, results in terms of accuracy, precision and recall. Segmentation can be thought of as binary classification problem where instances are all potential boundaries, and class labels are defined as boundary and non-boundary. Therefore, the initial approach for evaluating

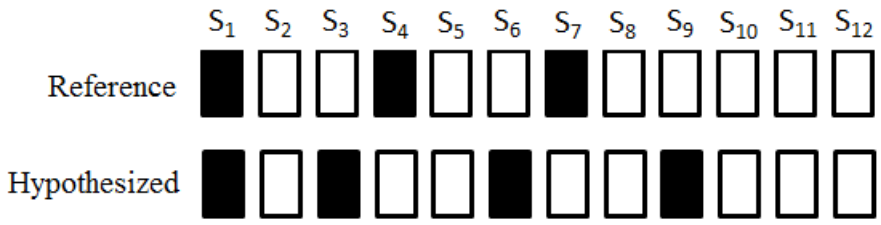

Figure 4.1: An example of reference and hypothesized segmentation. Each sentence is depicted as a box, and black boxes represent boundaries.

segmentation techniques was exactly the same as the evaluation in binary classification.

Given a particular document, all sentences except from the first and last sentence are potential boundaries. Obviously, the first sentence indicates the beginning of the initial segment, which is therefore explicitly marked as boundary, and the last sentence denotes the end of the last segment, which is an explicit non-boundary. As a result, each sentence is predicted as either a boundary (positive) or non-boundary (negative).

	Hypothesized Boundary	Hypothesized Non-boundary
True Boundary	True Positive (TP)	False Negative (FN)
True Non-boundary	False Positive (FP)	True Negative (TN)

Table 4.1: Contingency table for the possible outcomes of a segmentation system.

In such an evaluation, the first step is therefore to find the number of correctly and incorrectly hypothesized as well as non-hypothesized boundaries by a particular system. More specifically, correctly predicted boundaries are true positives where incorrectly predicted boundaries are false positives (false alarms). Similarly, true negatives are correctly hypothesized non-boundaries whereas false negatives (misdetections) are incorrectly hypothesized non-boundaries. Table 4.1 clearly summarizes these notions. Once getting these counts, the following step is to compute standard effectiveness measures (precision, recall and F-measure).

Precision [49, 50] is the fraction of hypothesized boundaries that are indeed true boundaries (Equation 4.1).

$$Precision = \frac{\#(Correctly \quad hypothesized \quad boundaries)}{\#(Hypothesized \quad boundaries)} = \frac{TP}{TP + FP} \tag{4.1}$$

Recall [49, 50] is the proportion of true boundaries that are predicted by the system (Equation 4.2).

$$Recall = \frac{\#(Correctly \quad hypothesized \quad boundaries)}{\#(True \quad boundaries)} = \frac{TP}{TP + FN} \tag{4.2}$$

F-measure [51, 50] is another metric which summarizes the effectiveness in a single number in terms of precision and recall. By definition, it is the weighted harmonic mean of precision and recall (Equation 4.3).

$$F = \frac{(\beta^2 + 1) \times Precision \times Recall}{\beta^2 Precision + Recall} \tag{4.3}$$

$$F_{\beta=1} = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{4.4}$$

Early evaluations for the task of segmentation were based on those classification-based measures. However, it was later realized that such measures were not appropriate for evaluating segmentation systems [3, 52]. The reason is that such measures yield similar results for both a quite terrible system and a system which predicts boundaries almost close to the referenced ones. In this situation, these standard measures heavily penalize systems which nearly miss the actual boundaries. Therefore, Beeferman et al. [3] states that closely estimated boundaries should count for something. For this purpose, one possible solution is to allow matches within a predefined sized window as well as the exact ones [53, 35]. However, this solution yet remains problematic in the sense that a predicted boundary just outside the window is still counted as bad as a further one.

Moreover, it is not possible to differentiate a perfect system from an almost-close system. For these reasons, such metrics are not reliable for evaluating the real performance of segmentation systems, and alternative measures are proposed to address these problematic issues.

4.2 Segmentation-Based Measures

This type of evaluation measures becomes prominent for evaluating segmentation systems, and solves issues regarding to initial evaluations based on standard classification measures. In other words, they are more sensitive to near misses, incorrectly detected segments and misdetections. Moreover, they do not enable trivial unprincipled techniques (e.g. identifying boundaries everywhere, or identifying no boundaries) to receive high evaluation scores.

Such measures are actually error (penalty) metrics. As they evaluate segmentations in terms of error, low scores indicate accurate segmentations. Roughly speaking, these measures find disagreements between the referenced and hypothesized segmentations by employing a fixed-sized sliding window across the text. In more detail, they share the following evaluation strategy: Pass a constant-sized window through the document and evaluate each time whether the two ends of the window are properly segmented given the knowledge of hypothesized and referenced segmentations. Lastly, normalize the total observed error by the number of windows. The difference lies in how each measure tends to examine the two ends of the window.

It is also remarkable that such metrics can also be analyzed in terms of missed boundaries and false alarms (incorrectly identified boundaries) to present a performance tradeoff as analogous to precision and recall for particular applications interested in such detail.

4.2.1 P_k: Probability of Segmentation Error

The probabilistic measure, P_k, is the probability that a randomly chosen pair of words, which are k-words apart from each other, is incorrectly classified by the hypothesized segmentation [3]. In other words, such segmentation either incorrectly predicts the pair in the same segment or falsely detects words in the pair as belonging to different segments. As P_k defines segmentation error in terms of probability, it ranges between 0 and 1. A perfect segmenter, which correctly hypothesize all boundaries, receives a score of 0 whereas a completely inaccurate segmenter may receive a score of 1.

P_k is computed as follows: Slide a fixed-sized (width of k) window across the text, and for each window position, compare the decision of hypothesized segmentation to the referenced one about whether they agree on the separation of two ends of the window. For a certain window with start i and end j, $\delta_R(i,j)$ is equivalent to one if the referenced segmentation (R) considers i and j in the same segment, and zero otherwise (Equation 4.5).

$$\delta_R(i,j) = \begin{cases} 1 & \text{if } R \text{ assigns } i \text{ and } j \text{ to the same segment} \\ 0 & \text{otherwise} \end{cases} \tag{4.5}$$

Similarly, $\delta_H(i,j)$ evaluates to one if the hypothesized segmentation (H) predicts i and j in the same segment, and zero otherwise (Equation 4.6).

$$\delta_H(i,j) = \begin{cases} 1 & \text{if } H \text{ assigns } i \text{ and } j \text{ to the same segment} \\ 0 & \text{otherwise} \end{cases} \tag{4.6}$$

Then, for a single window with start i and end j, the error is computed as $\delta_R(i,j) \oplus \delta_H(i,j)$ where $\oplus$ is the XOR operator. It means that error becomes one if segmentations disagree on the seperation of the two ends of the window, and otherwise zero. Lastly, total error is computed by summing the errors observed over all windows, and it is normalized by the number of windows (Equation 4.7).

$$P_k = \frac{\sum_{i=1}^{N-k} \delta_R(i, i+k) \oplus \delta_H(i, i+k)}{(N-k)} \tag{4.7}$$

where N is the number of words in the text.

The window size parameter, k, is typically determined as half the average segment length (in terms of words) in the referenced segmentation thereby ensuring that all trivial algorithms (hypothesizing boundaries everywhere, no boundaries, randomly placed boundaries and uniformly placed boundaries) almost receive the score of $P_k = 0.5$.

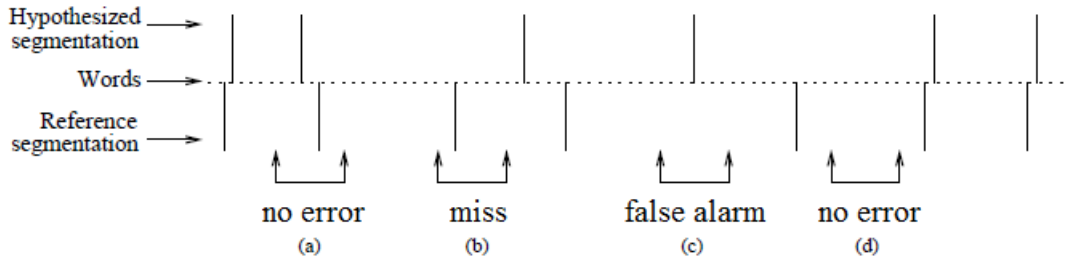

Figure 4.2: P_k evaluation for certain windows [3].

Figure 4.2 shows possible outcomes of the evaluation based on particular windows. Note that there exists k potential boundaries between two ends of a window. Intuitively, hypothesized and referenced boundaries either agree or disagree about the seperation of the two edges of the window. As initiated by the TDT Program [4], further analysis based on disagreements can be achieved for particular applications interested in such detail. Specifically, disagreements or failures can be viewed as false alarms (false positives) and misses (false negatives). False alarms are incorrectly identified boundaries by the hypothesized segmentation, and misses correspond to boundaries that could not be predicted by the hypothesized segmentation. Therefore, P_k can be expressed as the probability of misses and the probability of false alarms (Equation 4.8). P_{Miss} is the probability of observing no hypothesized boundary among windows including real boundaries (Equation 4.9), and $P_{FalseAlarm}$ is the probability of observing a hypothesized boundary among windows including no real boundaries (Equation 4.10).

$$P_k = P_{Miss} + P_{FalseAlarm} \tag{4.8}$$

where

$$P_{Miss} = \frac{\sum_{i=1}^{N-k} \delta_H(i, i+k).(1 - \delta_R(i, i+k))}{\sum_{i=1}^{N-k} (1 - \delta_R(i, i+k))} \tag{4.9}$$

$$P_{FalseAlarm} = \frac{\sum_{i=1}^{N-k} (1 - \delta_H(i, i+k)).\delta_R(i, i+k)}{\sum_{i=1}^{N-k} \delta_R(i, i+k)} \tag{4.10}$$

4.2.2 WD: WindowDiff

Despite P_k is the standard measure for evaluating segmentations, it penalizes misses (false negatives) more heavily than false alarms (false positives). Specifically, the total penalty for false negatives is always k (width of the window) where it is $k/2$ on average while assuming a uniform distribution of boundaries across the document [54]. Moreover, it does not consider the number of reference and hypothetical boundaries within a window which may cause some errors being unpenalized. For these reasons, Pevzner et al. [54] proposes a new metric called WindowDiff (WD) which addresses these problematic issues.

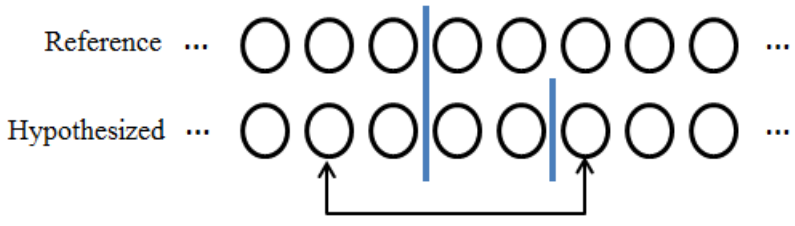

Figure 4.3: WD evaluation for a certain window of size $k = 4$. Circles represent words, and vertical lines indicate boundaries.

WD also employs a sliding window of fixed-length k to measure the error but in a slightly different way rather than P_k. For a certain window with start i and end j, it evaluates the segmentation by comparing the number of hypothetical boundaries, $H(i,j)$, to the number of referential boundaries, $R(i,j)$, lie within that window, indeed by taking the absolute difference of them. Then, it computes overall error by summing errors observed across all windows, and normalize it by the number of windows (Equation 4.11).

$$WD = \frac{\sum_{i=1}^{N-k} [|R(i,i+k) - H(i,i+k)| > 0]}{(N-k)} \tag{4.11}$$

WD is also expressed as misses (false negatives) and false alarms (false positives):

$$WD = WD_{Miss} + WD_{FalseAlarm} \tag{4.12}$$

where

$$WD_{Miss} = \frac{\sum_{i=1}^{N-k} [|R(i,i+k) > H(i,i+k)|]}{(N-k)} \tag{4.13}$$

$$WD_{FalseAlarm} = \frac{\sum_{i=1}^{N-k} [|R(i,i+k) < H(i,i+k)|]}{(N-k)} \tag{4.14}$$

WD and P_k are the most-widely used metrics as reporting the performance of segmentation techniques.

4.2.3 Pr_{error}

Georgescul et al. [55] propose another probabilistic measure, Pr_{error}, to solve the problem with WD, and also allow a tradeoff between misses and false alarms. It is noticed that WD penalizes misses less than false alarms. Both penalties are normalized by the total number of windows although it does not hold for misses. The reason is that misses occur only in the windows which contain true boundaries. Therefore, the number of misses is normalized by the number of windows including real boundaries.

$$Pr_{Miss} = \frac{\sum_{i=1}^{N-k} [|R(i, i+k) > H(i, i+k)|]}{\sum_{i=1}^{N-k} [|R(i, i+k) > 0|]} \tag{4.15}$$

$$Pr_{FalseAlarm} = \frac{\sum_{i=1}^{N-k} [|R(i, i+k) < H(i, i+k)|]}{(N-k)} \tag{4.16}$$

Misses and false alarms may be considered in varying degrees of importance depending on the application. For this purpose, Pr_{error} has cost variables associated with both type of penalties so that it allows a tradeoff between them (Equation 4.17). The cost of a miss is C_{Miss} where $0 \leq C_{Miss} \leq 1$, and the cost of a false alarm is $C_{FalseAlarm}$ where $0 \leq C_{FalseAlarm} \leq 1$. These cost values add up to one to ensure that Pr_{error} ranges between zero and one. For equal importance, each cost takes a value of 0.5.

$$Pr_{error} = C_{Miss}.Pr_{Miss} + C_{FalseAlarm}.Pr_{FalseAlarm} \tag{4.17}$$

Chapter 5

Datasets and Experiments

5.1 Datasets

There commonly exist two types of datasets to evaluate the performance of segmentation systems. These are text (synthetic) and speech datasets. Synthetic collections include documents typically created by concatenating different topical segments. In contrast, speech datasets include documents inherently containing several topical segments. Such collections are typically consists of speech transcripts produced by either manually or automatically through automatic speech recognition (ASR) systems. However, transcripts produced by ASR often include word errors and sentence segmentation errors.

	Document Count	Segment Count
Brown	200	2000
TRT	54	520
BilSeg	100	1000

Table 5.1: Statistics about the collections used for performance evaluation.

We evaluate the performance of the segmentation algorithms on three datasets: *Choi* in English, *TRT* and *BilSeg* in Turkish. Table 5.1 presents some statistics regarding to these collections.

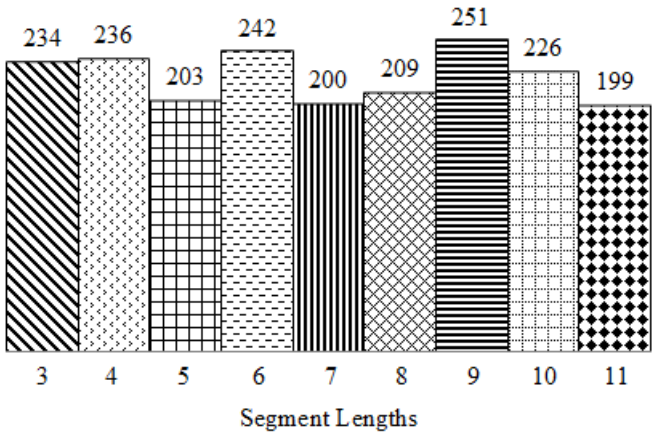

Figure 5.1: Distribution of segment lengths in the *Choi* collection.

Choi dataset is a synthetic corpus [37], and commonly used in the evaluation of segmentation systems. It includes a set of documents, each built by concatenating randomly sampled segments from the Brown corpus. We employ the subset of the *Choi* collection, which contains 200 documents where each document consists of ten segments. In this subset, there are 50 documents containing segments each with the length of 3–5, 6–8, 9–11 and 3–11 sentences (Table 5.2).

	3–5	6–8	9–11	3–11
#(documents)	50	50	50	50

Table 5.2: Number of documents in the *Choi* subcollection regarding to various segment length intervals.

TRT is a speech corpus containing news broadcast transcripts produced by manually [56]. Each transcript corresponds to a different news broadcast program, and includes distinct news stories. The length of segments in this collection ranges from two to thirteen sentences.

BilSeg is a synthetic corpus that includes documents created by concatenating randomly selected news articles from BilCol [18]. It is built in the same manner as the collection created by Ercan [11]. The length of the segments in this collection ranges from two to twenty three sentences.

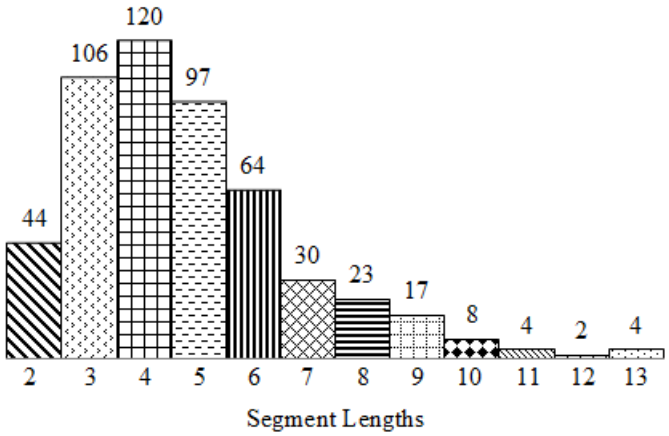

Figure 5.2: Distribution of segment lengths in the *TRT* collection.

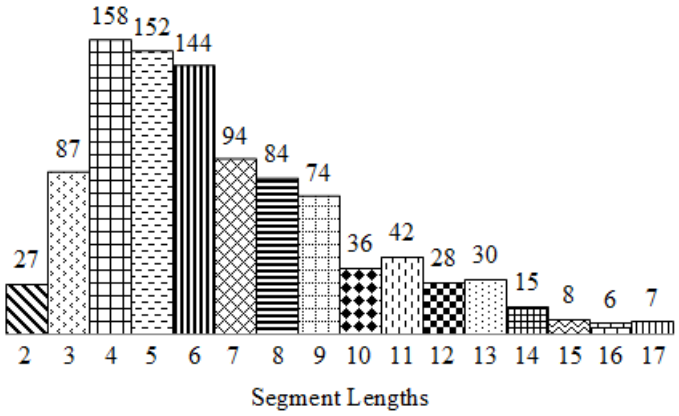

Figure 5.3: Distribution of segment lengths in the *BilSeg* collection.

5.2 Experimental Results and Discussion

We conducted experiments on the datasets mentioned in the previous section in order to evaluate and compare the performance of the proposed algorithm, *BTS*, with the two state-of-the-art algorithms, *C99* and *TextTiling*.

The proposed algorithm depends on several parameters. Specifically, *windowSize* and *featureReduction* are HAL-space parameters, and *blockSize* is the only

parameter of the algorithm. We performed searches for the optimal parameter settings that achieved the best performance in each dataset.

The first subsection investigates the effect of HAL space parameters while the parameter *blockSize* is set as the maximum segment length in sentences for each document in the respective collection. The next subsection analyzes the impact of the parameter *blockSize* as there is no knowledge regarding to the maximum segment length for a document. As analyzing the *blockSize* parameter, note that the HAL space parameters are determined as the parameter settings that achieve the best performance on each dataset in the first subsection. Lastly, the final subsection compares the best performance of the *BTS* algorithm with the performance of the specified state-of-the art algorithms.

5.2.1 Maximum Segment Length is Known

This section examines the effect of HAL space parameters, *windowSize* and *featureReduction* while setting the algorithm parameter *blockSize* as the maximum segment length in sentences for the respective document given the knowledge of such information. The *windowSize* parameter determines the size of the co-occurrence window as building co-occurrence matrix, and *featureReduction* determines the decision about whether to eliminate some features of the HAL vectors built for the document terms based on the predefined threshold. The values experimented for the parameter *windowSize* ranges from 3 to 10 with/without the application of feature reduction to the HAL vectors. These parameters are analyzed in terms of P_k and WD in each dataset, and the parameter settings that achieve the best performance for each dataset are reported together with the detailed evaluation results obtained by applying the proposed algorithm under these settings.

For the *Choi* collection, the best performance is achieved by the parameter setting where *windowSize* is 3 and *featureReduction* is applied to HAL vectors of document terms. As the *windowSize* increases, HAL vectors contain more words which may cause introducing spurious relatedness information, and therefore, the correlation between terms degrades accordingly. Therefore, the segmenter does

not perform well under such settings for the *windowSize* parameter.

	Precision	Recall	F-mes	P_k	*WD*
PS1	0.478	0.587	0.522	0.227	0.283
PS2	0.496	0.582	0.532	0.220	0.262

Table 5.3: Best results achieved for *Choi* collection where the parameter setting PS1:{$windowSize = 3$, $featureReduction = false$} and PS2:{$windowSize = 3$, $featureReduction = true$}.

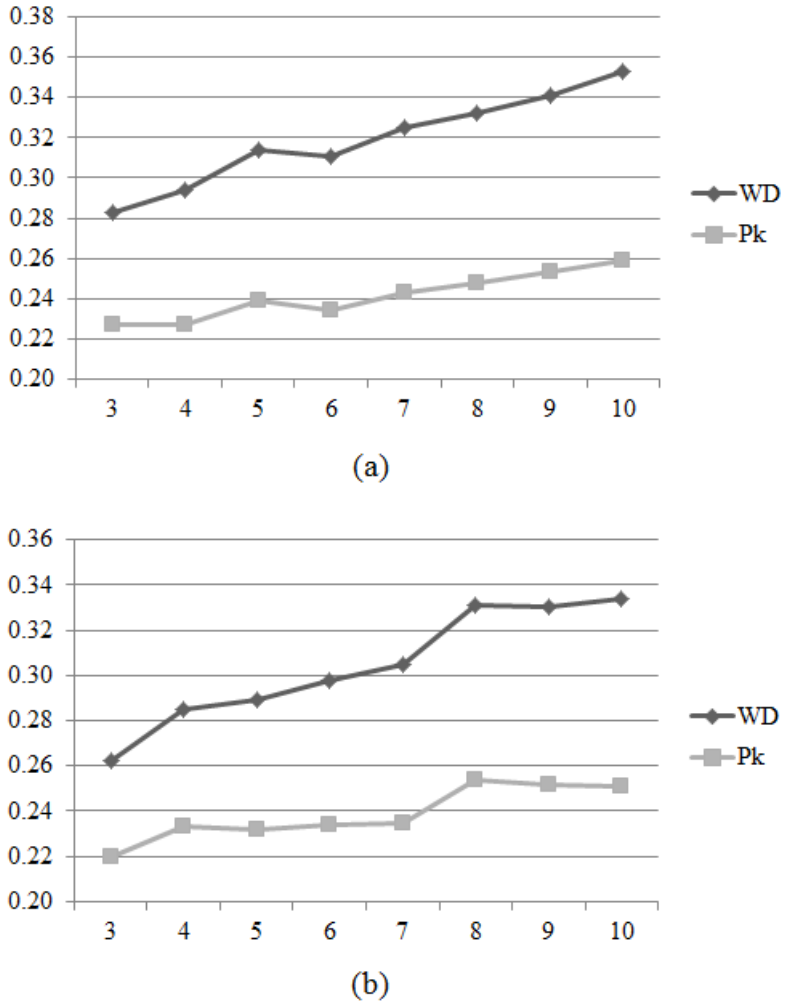

Figure 5.4: The plots of WD and P_k for *Choi* collection due to selected co-occurrence window sizes as feature reduction is (a) not applied (b) applied.

For the *TRT* collection, the best performance is achieved by the parameter setting where *windowSize* is 4 and *featureReduction* is not applied to HAL vectors of document terms. However, notice that the setting where *windowSize* is 7 and *featureReduction* is applied attains results closer to the best results.

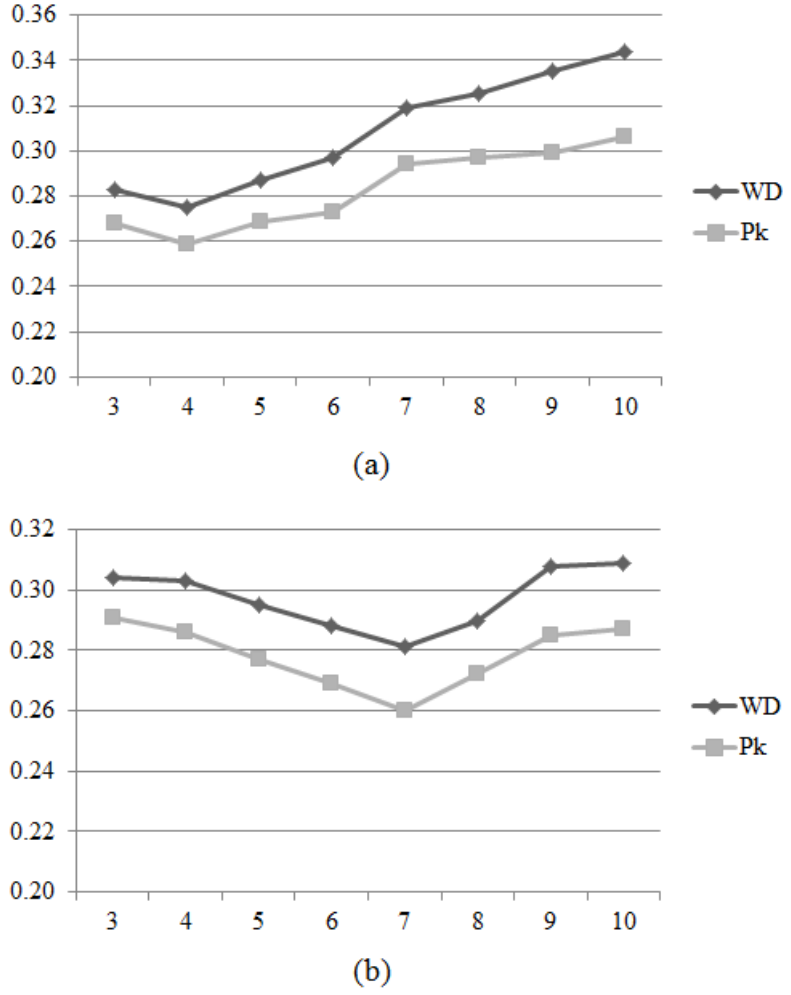

Figure 5.5: The plots of WD and P_k for *TRT* collection due to selected co-occurrence window sizes as feature reduction is (a) not applied (b) applied.

	Precision	Recall	F-mes	P_k	WD
PS1	0.519	0.487	0.502	0.259	0.275
PS2	0.521	0.501	0.511	0.260	0.281

Table 5.4: Best results achieved for TRT collection where the parameter setting PS1:$\{windowSize = 4,\ featureReduction = false\}$ and PS2:$\{windowSize = 7,\ featureReduction = true\}$.

For the *BilSeg* collection, the best performance is achieved by the parameter setting where *windowSize* is 5 and *featureReduction* is applied to HAL vectors of document terms. However, notice that the same *windowSize* setting with *featureReduction* is not applied obtains results very close to the best results.

	Precision	Recall	F-mes	P_k	WD
PS1	0.552	0.618	0.583	0.207	0.245
PS2	0.570	0.587	0.578	0.214	0.244

Table 5.5: Best results achieved for *BilSeg* collection where the parameter setting PS1:$\{windowSize = 5,\ featureReduction = false\}$ and PS2:$\{windowSize = 5,\ featureReduction = true\}$.

One common observation is that as we examine the plots regarding to P_k and WD, P_k values are always less than WD values obtained for a particular collection. It shows that the proposed algorithm often tends to produce false alarms, and P_k fails to detect some of them.

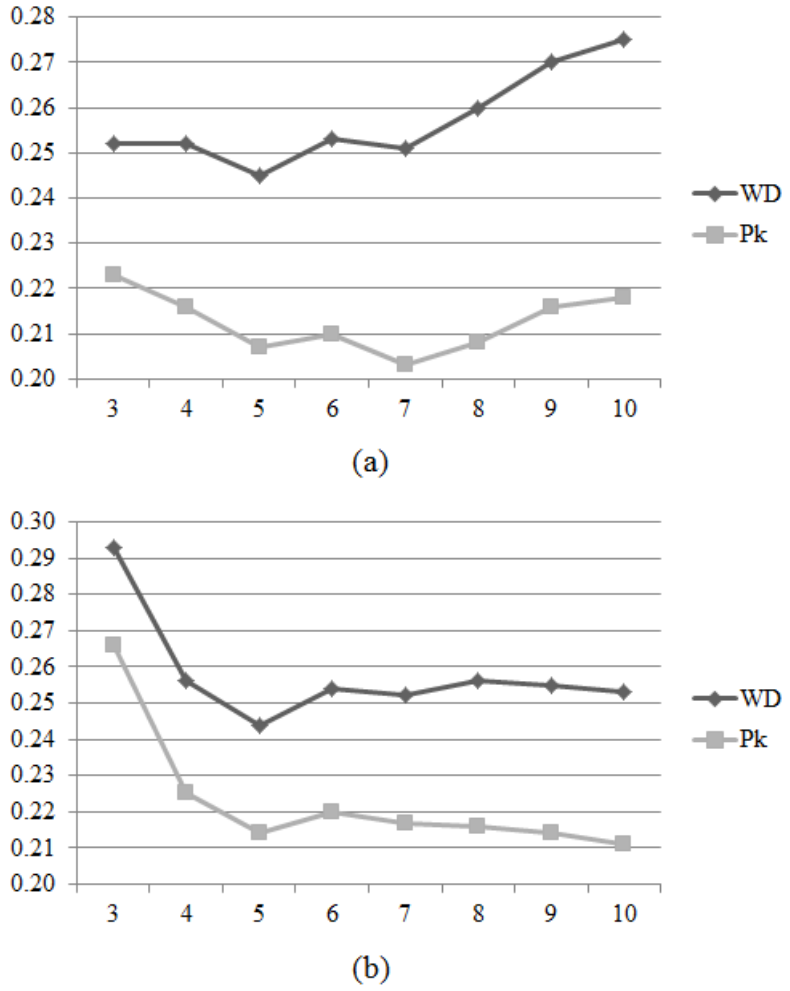

Figure 5.6: The plots of WD and P_k for *BilSeg* collection due to selected co-occurrence window sizes as feature reduction is (a) not applied (b) applied.

5.2.2 Maximum Segment Length is Unknown

This section analyzes the impact of the *blockSize* parameter as there is no knowledge regarding to the maximum segment length for a document. As analyzing the *blockSize* parameter, note that the HAL space parameters are determined as the parameter settings that achieve the best performance on each dataset in the previous section.

For the *Choi* and *TRT* collections, the best performance is observed when *blockSize* is 10, and for the *BilSeg* collection, the best performance is achieved when *blockSize* is 15. Thus, the proper selection of *blockSize* may depend on the characteristics of the documents in a collection, particularly the variation in segment lengths in the documents. The results achieved by these parameter settings are worse than the results obtained by the parameter settings in the previous section.

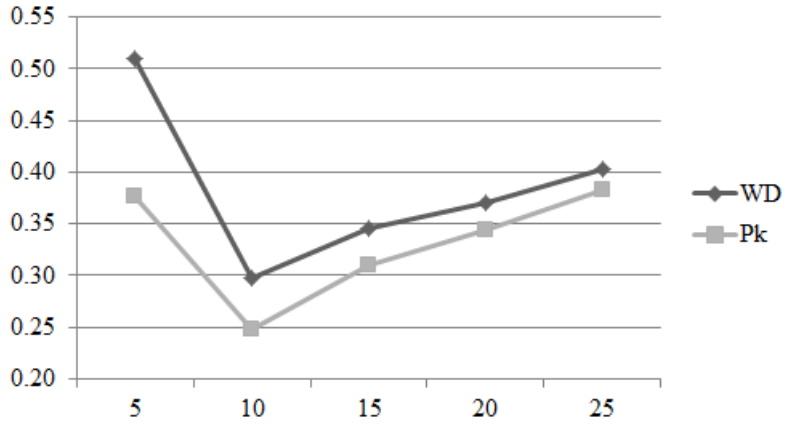

Figure 5.7: The plots of WD and P_k for *Choi* collection due to selected block sizes given that $windowSize = 3$ and $featureReduction$ is applied.

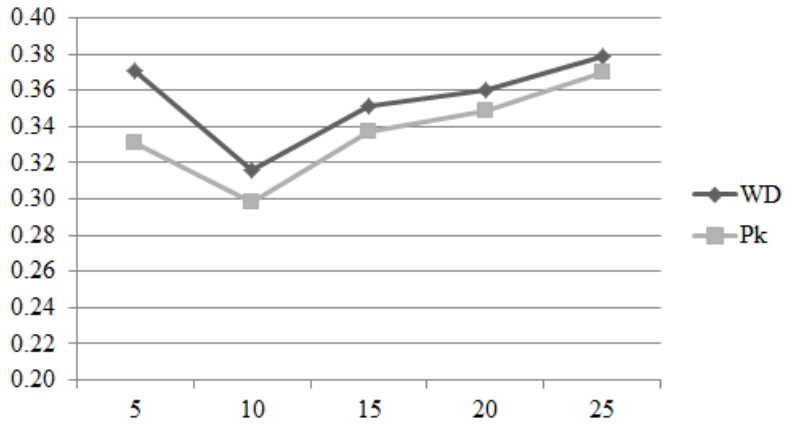

Figure 5.8: The plots of WD and P_k for *TRT* collection due to selected block sizes given that $windowSize = 4$ and $featureReduction$ is not applied.

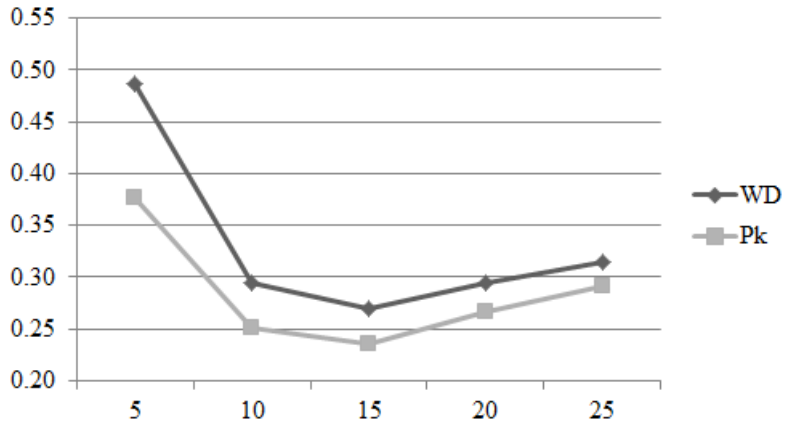

Figure 5.9: The plots of WD and P_k for *BilSeg* collection due to selected block sizes given that $windowSize = 5$ and $featureReduction$ is applied.

	Precision	Recall	F-mes	P_k	WD
PS	0.452	0.503	0.476	0.248	0.298

Table 5.6: Best results achieved for *Choi* collection where the parameter setting PS:$\{windowSize = 3,\ featureReduction = true,\ blockSize = 10\}$.

	Precision	Recall	F-mes	P_k	WD
PS	0.493	0.424	0.456	0.298	0.316

Table 5.7: Best results achieved for TRT collection where the parameter setting PS:$\{windowSize = 4,\ featureReduction = false,\ blockSize = 10\}$.

	Precision	Recall	F-mes	P_k	WD
PS	0.540	0.533	0.537	0.235	0.269

Table 5.8: Best results achieved for *BilSeg* collection where the parameter setting PS:$\{windowSize = 5,\ featureReduction = true,\ blockSize = 15\}$.

5.2.3 Performance Comparison

This section compares the best performance achieved by the proposed algorithm, *BTS*, with the performance of the two state-of-the art algorithms, *C99* and *TextTiling*. As drawing general conclusions by simply looking at the results, *C99* outperforms the other methods on all datasets. On the *TRT* collection, *C99* outperforms *BTS* just by 3% in terms of WD. On the *BilSeg* collection, the *BTS* is ranked as the third algorithm just by 1% difference in terms of WD due to the secondly ranked method *TextTiling*.

The performance of the algorithms are also statistically evaluated in terms of the evaluation metric *WD* by applying ANOVA, which is the common statistical method for testing the differences between multiple means [57, 58, 59]. For this purpose, each document is considered as an observation, and the documents are assumed to be independent from each other. Another assumption is that WD values are approximately normally distributed and there exists homogeneity of variances. As the post–hoc test, we choose Tukey [60] with the significance level of 0.05 to find out which algorithms actually differ. For statistical testing, we employ IBM SPSS Statistics Version 22.0.0 [61].

For the *Choi* collection, there is a statistically significant difference between the means as determined by one–way ANOVA. A Tukey post–hoc test then revealed that the difference is statistically significant for all algorithms.

For the *TRT* collection, there is a statistically significant difference between the means as determined by one–way ANOVA. A Tukey post–hoc test then revealed that the difference is statistically significant between the algorithms *C99* and *TextTiling*, and *BTS* and *TextTiling*. There is no statistically significant differences between *C99* and *BTS*.

For the *BilSeg* collection, there is a statistically significant difference between the means as determined by one–way ANOVA. A Tukey post–hoc test then revealed that the difference is statistically significant between the algorithms *C99* and *TextTiling*, and *C99* and *BTS*. There is no statistically significant differences

between *TextTiling* and *BTS*.

Our proposed algorithm shows nearly consistent performance for each dataset without being affected by the type of the datasets. However, *C99* is strongly affected as producing segmentations for the *TRT* collection. Thus, the performance of *C99* varies according to the selected dataset. We can further hypothesize that our method finds many false alarms as we examine the precision values, and these may correspond to subtopic shifts in a segment. Therefore, it is possible that our proposed method can be more successful as producing fine-grained (hierachical) segmentations.

	Precision	Recall	F-mes	P_k	*WD*
C99	0.726	0.752	0.739	0.116	0.135
TT	0.495	0.606	0.545	0.247	0.308
BTS	0.496	0.582	0.532	0.220	0.262

Table 5.9: Performance comparison of C99, TT(TextTiling) and BTS for the *Choi* collection.

	Precision	Recall	F-mes	P_k	*WD*
C99	0.652	0.501	0.567	0.235	0.242
TT	0.353	0.200	0.256	0.352	0.376
BTS	0.519	0.487	0.502	0.259	0.275

Table 5.10: Performance comparison of C99, TT(TextTiling) and BTS for the *TRT* collection.

	Precision	Recall	F-mes	P_k	*WD*
C99	0.756	0.746	0.751	0.127	0.142
TT	0.499	0.504	0.502	0.206	0.230
BTS	0.570	0.587	0.578	0.214	0.244

Table 5.11: Performance comparison of C99, TT(TextTiling) and BTS for the *BilSeg* collection.

Chapter 6

Conclusions and Future Work

We present a novel local-context based approach depending on lexical cohesion for linear text segmentation, which is the task of dividing text into a linear sequence of coherent segments. As the lexical cohesion indicator, the proposed technique exploits relationships among terms induced from semantic space called HAL (Hyperspace Analogue to Language), which is built upon by examining co-occurrence of terms through passing a fixed-sized window over text. We define new concepts regarding to term correlation and sentence relatedness. The algorithm iteratively discovers topical shifts by examining the most relevant sentence pairs in a block of sentences considered at each iteration. The proposed technique is evaluated on both error-free speech transcripts of news broadcasts and documents formed by concatenating different topical regions of text. A new corpus for Turkish is automatically built where each document is formed by concatenating different news articles. For performance comparison, two state-of-the-art methods, TextTiling and C99, are leveraged, and the results show that the proposed approach has comparable performance with these two techniques.

We consider the following directions for future work:

- Determining the algorithm parameter *blockSize* automatically by examining the document to be segmented,

- Embedding different term correlation and sentence relatedness measures into the proposed algorithm, *BTS*,
- Building HAL vectors for terms over the collection instead of the document to be segmented,
- Applying the proposed algorithm for the task of hierarchical text segmentation.

Bibliography

[1] J. C. Reynar, "An automatic method of finding topic boundaries," in *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, pp. 331–333, Association for Computational Linguistics, 1994.

[2] M. A. Hearst, "Texttiling: Segmenting text into multi-paragraph subtopic passages," *Computational linguistics*, vol. 23, no. 1, pp. 33–64, 1997.

[3] D. Beeferman, A. Berger, and J. Lafferty, "Statistical models for text segmentation," *Machine Learning*, vol. 34, no. 1-3, pp. 177–210, 1999.

[4] J. Allan, J. G. Carbonell, G. Doddington, J. Yamron, and Y. Yang, "Topic detection and tracking pilot study: Final report," *DARPA Broadcast News Transcription and Understanding Workshop*, pp. 194–218, 1998.

[5] R. Barzilay, M. Elhadad, *et al.*, "Using lexical chains for text summarization," in *Proceedings of the ACL workshop on Intelligent Scalable Text Summarization*, vol. 17, pp. 10–17, Madrid, Spain;, 1997.

[6] R. Barzilay and L. Lee, "Catching the drift: Probabilistic content models, with applications to generation and summarization.," in *HLT-NAACL*, pp. 113–120, 2004.

[7] J. Renkema, *Introduction to discourse studies*. John Benjamins, 2004.

[8] J. Morris and G. Hirst, "Lexical cohesion computed by thesaural relations as an indicator of the structure of text," *Computational Linguistics*, vol. 17, no. 1, pp. 21–48, 1991.

[9] M. A. Halliday and R. Hasan, "Cohesion in English," 1976.

[10] M. Halliday, "An introduction to functional grammar," 1994.

[11] G. Ercan, *Lexical cohesion analysis for topic segmentation, summarization and keyphrase extraction.* PhD thesis, Bilkent University, Computer Engineering, 2012.

[12] B. J. Grosz and C. L. Sidner, "Attention, intentions, and the structure of discourse," *Computational Linguistics*, vol. 12, no. 3, pp. 175–204, 1986.

[13] M. Galley, K. McKeown, E. Fosler-Lussier, and H. Jing, "Discourse segmentation of multi-party conversation," in *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pp. 562–569, Association for Computational Linguistics, 2003.

[14] A. Gruenstein, J. Niekrasz, and M. Purver, "Meeting structure annotation," in *Recent Trends in Discourse and Dialogue*, pp. 247–274, Springer, 2008.

[15] Y. Yaari, "Segmentation of expository texts by hierarchical agglomerative clustering," *arXiv preprint cmp-lg/9709015*, 1997.

[16] P.-Y. Hsueh, J. D. Moore, and S. Renals, "Automatic segmentation of multiparty dialogue.," in *EACL*, 2006.

[17] J. Eisenstein, "Hierarchical text segmentation from multi-scale lexical cohesion," in *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 353–361, Association for Computational Linguistics, 2009.

[18] F. Can, S. Kocberber, O. Baglioglu, S. Kardas, H. C. Ocalan, and E. Uyar, "New event detection and topic tracking in turkish," *Journal of the American Society for Information Science and Technology*, vol. 61, no. 4, pp. 802–819, 2010.

[19] G. Youmans, "A new tool for discourse analysis: The vocabulary-management profile," *Language*, pp. 763–789, 1991.

[20] M. Mohri, P. Moreno, and E. Weinstein, "Discriminative topic segmentation of text and speech," in *International Conference on Artificial Intelligence and Statistics*, pp. 533–540, 2010.

[21] I. Malioutov and R. Barzilay, "Minimum cut model for spoken lecture segmentation," in *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pp. 25–32, Association for Computational Linguistics, 2006.

[22] A. Kazantseva and S. Szpakowicz, "Linear text segmentation using affinity propagation," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 284–293, Association for Computational Linguistics, 2011.

[23] H. Misra, F. Yvon, O. Cappé, and J. Jose, "Text segmentation: A topic modeling perspective," *Information Processing & Management*, vol. 47, no. 4, pp. 528–544, 2011.

[24] M. Georgescul, A. Clark, and S. Armstrong, "A comparative study of mixture models for automatic topic segmentation of multiparty dialogues.," in *IJCNLP*, pp. 925–930, 2008.

[25] H. Misra, F. Yvon, J. M. Jose, and O. Cappe, "Text segmentation via topic modeling: An analytical study," in *Proceedings of the 18th ACM conference on Information and knowledge management*, pp. 1553–1556, ACM, 2009.

[26] M. Riedl and C. Biemann, "Text segmentation with topic models," *The Journal for Language Technology and Computational Linguistics*, pp. 47–69, 2012.

[27] J. Hirschberg and D. Litman, "Empirical studies on the disambiguation of cue phrases," *Computational linguistics*, vol. 19, no. 3, pp. 501–530, 1993.

[28] G. Tür, D. Hakkani-Tür, A. Stolcke, and E. Shriberg, "Integrating prosodic and lexical cues for automatic topic segmentation," *Computational Linguistics*, vol. 27, no. 1, pp. 31–57, 2001.

[29] J. Arguello and C. Rosé, "Topic segmentation of dialogue," in *Proceedings of the HLT-NAACL 2006 Workshop on Analyzing Conversations in Text and Speech*, pp. 42–49, Association for Computational Linguistics, 2006.

[30] M. Georgescul, A. Clark, S. Armstrong, *et al.*, "Exploiting structural meeting-specific features for topic segmentation," in *Actes de la 14eme Conférence sur le Traitement Automatique des Langues Naturelles*, pp. 15–24, 2007.

[31] M. Dowman, V. Savova, T. L. Griffiths, K. Kording, J. B. Tenenbaum, and M. Purver, "A probabilistic model of meetings that combines words and discourse features," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 16, no. 7, pp. 1238–1248, 2008.

[32] J. Eisenstein and R. Barzilay, "Bayesian unsupervised topic segmentation," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 334–343, Association for Computational Linguistics, 2008.

[33] K. W. Church, "Char_align: a program for aligning parallel texts at the character level," in *Proceedings of the 31st annual meeting on Association for Computational Linguistics*, pp. 1–8, Association for Computational Linguistics, 1993.

[34] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information processing & management*, vol. 24, no. 5, pp. 513–523, 1988.

[35] J. M. Ponte and W. B. Croft, "Text segmentation by topic," in *Research and Advanced Technology for Digital Libraries*, pp. 113–125, Springer, 1997.

[36] J. Xu and W. B. Croft, "Query expansion using local and global document analysis," in *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 4–11, ACM, 1996.

[37] F. Y. Choi, "Advances in domain independent linear text segmentation," in *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pp. 26–33, Association for Computational Linguistics, 2000.

[38] G. Dias, E. Alves, and J. G. P. Lopes, "Topic segmentation algorithms for text summarization and passage retrieval: an exhaustive evaluation," in *American Association for Artificial Intelligence*, vol. 7, pp. 1334–1339, 2007.

[39] J. P. Yamron, I. Carp, L. Gillick, S. Lowe, and P. van Mulbregt, "A hidden markov model approach to text segmentation and event tracking," in *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*, vol. 1, pp. 333–336, IEEE, 1998.

[40] D. M. Blei and P. J. Moreno, "Topic segmentation with an aspect hidden markov model," in *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 343–348, ACM, 2001.

[41] M. Utiyama and H. Isahara, "A statistical model for domain-independent text segmentation," in *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pp. 499–506, Association for Computational Linguistics, 2001.

[42] M. F. Porter, "An algorithm for suffix stripping," *Program: Electronic Library and Information Systems*, vol. 14, no. 3, pp. 130–137, 1980.

[43] "Zemberek: An Open Source Natural Language Processing Java Library for Turkish." `http://code.google.com/p/zemberek`. Accessed: 2011-03-17.

[44] C. Carpineto and G. Romano, "A survey of automatic query expansion in information retrieval," *ACM Computing Surveys (CSUR)*, vol. 44, no. 1, p. 1, 2012.

[45] J. Bai, D. Song, P. Bruza, J.-Y. Nie, and G. Cao, "Query expansion using term relationships in language models for information retrieval," in *Proceedings of the 14th ACM international conference on Information and knowledge management*, pp. 688–695, ACM, 2005.

[46] G. A. Miller, "Wordnet: A lexical database for english," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[47] K. Lund and C. Burgess, "Producing high-dimensional semantic spaces from lexical co-occurrence," *Behavior Research Methods, Instruments, & Computers*, vol. 28, no. 2, pp. 203–208, 1996.

[48] C. Burgess, K. Livesay, and K. Lund, "Explorations in context space: Words, sentences, discourse," *Discourse Processes*, vol. 25, no. 2-3, pp. 211–257, 1998.

[49] G. Salton, *Automatic Text Processing: The Transformation, Analysis, and Retrieval of*. Addison-Wesley, 1989.

[50] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to information retrieval*, vol. 1. Cambridge University Press Cambridge, 2008.

[51] C. J. V. Rijsbergen, *Information Retrieval*. Newton, MA, USA: Butterworth-Heinemann, 2nd ed., 1979.

[52] R. J. Passonneau and D. J. Litman, "Discourse segmentation by human and automated means," *Computational Linguistics*, vol. 23, no. 1, pp. 103–139, 1997.

[53] J. C. Reynar, *Topic segmentation: Algorithms and applications*. PhD thesis, University of Pennsylvania, 1998.

[54] L. Pevzner and M. A. Hearst, "A critique and improvement of an evaluation metric for text segmentation," *Computational Linguistics*, vol. 28, no. 1, pp. 19–36, 2002.

[55] M. Georgescul, A. Clark, and S. Armstrong, "An analysis of quantitative aspects in the evaluation of thematic segmentation algorithms," in *Proceedings of the 7th SIGdial Workshop on Discourse and Dialogue*, pp. 144–151, Association for Computational Linguistics, 2006.

[56] D. Küçük, *Exploiting information extraction techniques for automatic semantic annotation and retrieval of news videos in Turkish*. PhD thesis, Middle East Technical University, Computer Engineering, 2011.

[57] J. Zar, "Biostatistical analysis (4th edition)," *Princeton Hall, New Jersey*, 1999.

[58] D. J. Sheskin, *Handbook of parametric and nonparametric statistical procedures*. crc Press, 2003.

[59] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *The Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

[60] J. W. Tukey, "Comparing individual means in the analysis of variance," *Biometrics*, pp. 99–114, 1949.

[61] "IBM SPSS Statistics Software." `http://www-01.ibm.com/software/analytics/spss/products/statistics/`. Accessed: 2014-03-05.