Evaluating Text Segmentation

by

Christopher Fournier

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the M.A.Sc. degree in
Electrical and Computer Engineering

Ottawa-Carleton Institute for Electrical and Computer Engineering
School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

This thesis investigates the evaluation of automatic and manual text segmentation. Text segmentation is the process of placing boundaries within text to create segments according to some task-dependent criterion. An example of text segmentation is topical segmentation, which aims to segment a text according to the subjective definition of what constitutes a topic. A number of automatic segmenters have been created to perform this task, and the question that this thesis answers is how to select the best automatic segmenter for such a task. This requires choosing an appropriate segmentation evaluation metric, confirming the reliability of a manual solution, and then finally employing an evaluation methodology that can select the automatic segmenter that best approximates human performance.

A variety of comparison methods and metrics exist for comparing segmentations (e.g., WindowDiff, P_k), and all save a few are able to award partial credit for nearly missing a boundary. Those comparison methods that can award partial credit unfortunately lack consistency, symmetricity, intuition, and a host of other desirable qualities. This work proposes a new comparison method named *boundary similarity* (B) which is based upon a new minimal *boundary edit distance* to compare two segmentations. Near misses are frequent, even among manual segmenters (as is exemplified by the low inter-coder agreement reported by many segmentation studies). This work adapts some inter-coder agreement coefficients to award partial credit for near misses using the new metric proposed herein, B.

The methodologies employed by many works introducing automatic segmenters evaluate them simply in terms of a comparison of their output to one manual segmentation of a text, and often only by presenting nothing other than a series of mean performance values (along with no standard deviation, standard error, or little if any statistical hypothesis testing). This work asserts that one segmentation of a text cannot constitute a "true" segmentation; specifically, one manual segmentation is simply one sample of the population of all possible segmentations of a text and of that subset of desirable segmentations. This work further asserts that an adapted inter-coder agreement statistics proposed herein should be used to determine the reproducibility and reliability of a coding scheme and set of manual codings, and then statistical hypothesis testing using the specific comparison methods and methodologies demonstrated herein should be used to select the best automatic segmenter.

This work proposes new segmentation evaluation metrics, adapted inter-coder agreement coefficients, and methodologies. Most important, this work experimentally compares

the state-or-the-art comparison methods to those proposed herein upon artificial data that simulates a variety of scenarios and chooses the best one (B). The ability of adapted inter-coder agreement coefficients, based upon B, to discern between various levels of agreement in artificial and natural data sets is then demonstrated. Finally, a contextual evaluation of three automatic segmenters is performed using the state-of-the art comparison methods and B using the methodology proposed herein to demonstrate the benefits and versatility of B as opposed to its counterparts.

Acknowledgements

Along this journey that have I undertaken—culminating in this thesis, a great many people have aided me. They have all shaped my mind, and made me the researcher that I am today. This work is a reflection of how far I have come, and I am eternally grateful to those who have helped me along the way.

Thank you to my supervisor, Prof. Diana Inkpen, for her patience with me as a student. Prof. Stan Szpakowicz, for introducing me to the field of natural language processing (NLP), and for helping me better my skills as a researcher. Anna Kazantseva, for providing the inspiration and impetus for this work; without her data, feedback, and encouragement, this work would never have even begun. Martin Scaiano, for his thorough feedback. Saif Mohammad, for suggesting experiments upon artificial data to evaluate the inter-coder agreement statistics proposed herein. Alistair Kennedy, for his feedback and spirited debate upon evaluation. David Nadeau, helping me to realize that my initial Master's topic was a folly, and extreme patience as I completed this work. Murray McCulligh, for teaching me the software development skills required for me to implement the software supporting this work. Prof. Shikharesh Majumdar, for convincing me to pursue graduate studies. Prof. Babak Esfandiari, for listening to my thoughts and then providing me with the single best piece of career counselling I have ever received: to drop my other notions and to instead pursue my desire to research NLP. Alan Petten, for instilling in me the desire for exploration that has gotten me this far, and giving me an environment in which my young mind could limitlessly explore the world of computer science.

Thank you to all of my friends for their patience, and especially to those who have annotated data for me. Last but not least, thank you to my parents and grandparents for their loving support.

Contents

Tables

Figures

Algorithms

Chapter 1

Introduction

In computational linguistics (CL) and natural language processing (NLP), text segmentation is the task of splitting text into a series of segments by placing boundaries within. The purposes for doing this vary greatly depending upon the task being undertaken. Segmentation is often used as a pre-processing step by other NLP systems such as those that perform video and audio retrieval (Franz et al., 2007), question answering (Oh et al., 2007), subjectivity analysis (Stoyanov and Cardie, 2008), and even automatic summarization (Haghighi and Vanderwende, 2009). One popular segmentation task it to perform topical segmentation (e.g., Hearst 1997), i.e., the separation of text into topically cohesive segments by some subjective parameter(s).

Segmentations can occur at a variety of levels of granularity, i.e., atomic units. This granularity could be structural (e.g., morphemes, words, phrases, sentences, paragraphs, etc.) or even related to vocalization (e.g., phonemes, syllables, etc.). The choice of what atomic unit to choose depends upon the task. For example, Figure 1.1 shows a hypothetical topical segmentation of a popular science article at the paragraph level—meaning that paragraphs are the atomic unit of text used by that study—with descriptions of the contents of each segment (Hearst, 1997, p. 33).

Text segmentation can take many forms. Segmentation can place a single type of boundary and create one which, for example, models where topics begin and end in text without overlap—referred to as linear segmentation. Multiple types of boundaries can also be placed in a linear segmentation to model characteristics of the boundaries themselves. These characteristics could include whether a boundary represents the end of an act or a scene in a play—referred to herein as multiple-boundary-type linear segmentation. In topical segmentation, segments can be thought of as belonging to larger topical segments

Paragraph	*Topic*
1–3	Intro—the search for life in space
4–5	The moon's chemical composition
6–8	How early earth-moon proximity shaped the moon
9–12	How the moon helped life evolve on earth
13	Improbability of the earth-moon system
14–16	Binary/trinary star systems make life unlikely
17–18	The low probability of nonbinary/trinary systems
19–20	Properties of earth's sun that facilitate life
21	Summary

Figure 1.1: A hypothetical topical segmentation at the paragraph level of a popular science article (Hearst, 1997, p. 33)

which span multiple smaller sub-segments, much like a thesis is often organized into sections contained within chapters. Such a segmentation resembles the hierarchy found in a table of contents, and is referred to as a hierarchical segmentation. This work concerns itself primarily with linear, and not hierarchical, segmentation, and although it makes provisions for dealing with multiple-boundary-type segmentation it does not experiment upon such data sets.

Segmentation can be a subjective task (Mann et al., 1992), especially when the segmentation task is only known to have been performed properly by a human. Topical segmentation has the potential to be wildly subjective depending upon not only the task but the type of text being analysed—different types of text will have different definitions of what constitutes a topic. In the topical segmentation a novel (e.g., Kazantseva and Szpakowicz 2012), does a topic shift occur when the setting changes? Or when character dialogue shifts topic? When characters enter or exit? In the intention-based segmentation of a monologue (e.g., Passonneau and Litman 1993), is a boundary denoted by a referential noun phrase, a cue word, or a pause? Because of the subjectivity of many segmentation tasks in CL, machine learning (ML) has been used to try to create artificial segmenters. ML has also been used to gain a greater understanding of the nature of a segmentation task or to simply automate the task for consistency.

There are a variety of automatic segmenters that exist for topical segmentation, including: TextTiling (Hearst, 1993, 1994, 1997), Bayesian Unsupervised Segmentation (BayesSeg; Eisenstein and Barzilay 2008), Affinity Propagation for Segmentation (APS; Kazantseva and Szpakowicz 2011), and Minimum Cut Segmenter (MinCut; Malioutov and Barzilay 2006). This work is concerned with creating and demonstrating tools and methodology to evaluate and select an ideal segmenter from a group of segmenters.

Segmentation evaluation may seem like a simple task, but it is subtly difficult.

The first difficulty that exists in determining which automatic segmenter performs best is that often it is difficult to obtain a "true" segmentation to use as training data or as a reference to compare against. Many studies report low agreement between humans even though they segment the same text using the same instructions (see Table 3.3 on page 56). The reason why many of these studies report low agreement is due to a fact, "known to researchers in discourse analysis from as early as Levin and Moore (1978), that while [coders] generally agree on the 'bulk' of segments, they tend to disagree on their exact boundaries" (Artstein and Poesio, 2008, p. 40). Why do they often disagree?

Answering why human segmenters often disagree must be done on a case-by-case basis, but the subjectivity of such a task is one potential cause of this phenomenon. Analysis of how they disagree shows that humans either completely miss placing a boundary that another human has (i.e., a full miss), or they place one near a boundary that another human has, but not at the exact location (i.e., a near miss). An evaluation methodology is required that can account for this difficulty, of which a few have been proposed by others (e.g., Passonneau and Litman 1993; Hearst 1997), but they have a variety of drawbacks as outlined in Chapter 3, and alternatives to them are proposed, demonstrated, and evaluated herein. Most important though, a method of comparing segmentations that can award some partial credit to an automatic segmenter that nearly misses boundaries is needed, because "in almost any conceivable application, a segmenting tool that consistently comes close—off by a sentence, say—is preferable to one that places boundaries willy-nilly" (Beeferman et al., 1997, p. 42).

A number of comparison methods that can award partial credit for near misses already exist—the most popular of which are P_k (Beeferman and Berger, 1999, pp. 198–200) and WindowDiff (Pevzner and Hearst, 2002, p. 10), but they too have a variety of shortcomings and deficiencies as experimentally demonstrated herein. To replace them, this work proposes a new minimal boundary edit-distance that can be normalized to produce a superior comparison method of segmentation evaluation. From the edit distance and comparison method proposed herein, methods of determining the reliability of manual segmentation data are adapted for segmentation. Additionally, an overall segmentation evaluation methodology is demonstrated and proposed.

1.1 Thesis Statement

This thesis claims that:

> *The minimal boundary edit-distance proposed herein can be normalized to create a segmentation comparison method that improves upon the state-of-the art, can be used to adapt inter-coder agreement coefficients to measure the reliability and replicability of manual segmentations, and with the evaluation methodology proposed herein can be used to evaluate the relative performance of automatic and/or manual segmenters.*

1.2 Research Questions

To support the thesis statement, the following research questions must be answered:

1. How can we best compare two arbitrary segmentations?
2. How can we determine the reliability of manual segmentations and replicability of how they are collected?
3. How can we select the best automatic segmenter for a task?

If, for a specific text item and segmentation task, a manual segmentation is compared to another segmentation, how do we quantify the differences between them? Quantifying the differences between two segmentations requires a distance function. There are many distance functions available to us, and the question is whether any of them can be mathematically considered as either a comparison method or a metric. The distance function we use to compare two segmentations should also differentiate between a variety of sources of dissimilarity that adapt to what a specific task would consider an error. Our ideal distance function would also be configurable enough to easily map to what a specific task would consider to be an error.

For a specific text item and segmentation task, does there exist one "true" reference segmentation that we can compare against? If a "true" reference segmentation does not exist, how can we measure the reliability of segmentations that are manually produced, and how do manual segmenters disagree? When we are training upon manual segmentations, we assume that we have elicited a specific effect from our human segmenters that we wish to emulate; how can we determine that our study is replicable, and that we have observed the effect that we wanted to? The answers to these questions serve as the foundation

upon which we can begin to evaluate the performance of an automatic segmenter: trust in the manual segmenters whom we wish to approximate.

Assuming that, for a task, we have found a reliable set of manual segmentations, how can we select the best automatic segmenter for this task? Beginning with our reliable manual segmentations, how can we utilize multiple manual segmentations of the same text item to measure how closely we can approximate human performance and chose an appropriate automatic segmenter that will operate best upon unseen data? With the answers to these questions in hand, we can begin to search for the best machine learning methods to perform automatic segmentation for a task.

1.3 Contributions

This work contributes:

1. A new edit distance that operates upon boundaries and potential boundary positions that can act as a distance function to compare two segmentations called *boundary edit distance*;

2. This edit distance is then normalized to produce a new comparison method, named *boundary similarity* (B), that is proposed to replace WindowDiff and P_k;

3. This comparison method is then used to adapt two inter-coder agreement coefficients (π and κ and their multi-coder counterparts) so that they can account for near misses and then be used to determine the reliability and replicability of segmentation data;

4. Using the segmentation evaluation tools contributed by this work (an edit distance, comparison method, and inter-coder agreement coefficients), a methodology is proposed and demonstrated for segmentation evaluation that allows for the selection of the best performing automatic segmenter for a specific task, short of performing an ecological evaluation.

Additionally, this research has resulted in two publications (Fournier and Inkpen 2012 and Fournier 2013) and a software package for the evaluation of segmentation called `segeval` (Fournier, 2012).

1.4 Outline

In this work, first background is presented in Chapter 2 and a review of state-of-the-art segmentation comparison methods, inter-coder agreement coefficients, and segmentation evaluation methodology is presented in Chapter 3. A complete overview of evaluation methodology is not presented, but the scope of the evaluation methodology proposed is defined and statistical hypothesis testing overviewed.

Chapter 4 begins by defining a method of conceptualizing and representing segmentation. Later, this representation is used to communicate and propose *boundary edit distance* along with four potential normalizations of the edit distance which are later tested for their suitability as segmentation comparison methods in Section 4.1 against WindowDiff. Inter-coder agreement coefficients are then adapted for segmentation in Section 4.2 which could use any of the four proposed comparison methods. Evaluation methodology are then proposed for segmentation in Section 4.3 which could be used for any segmentation comparison method.

Chapter 5 experimentally evaluates the proposals made in Chapter 4 and discusses the results of each individual experiment as they are performed. First, an experimental evaluation of WindowDiff and the four comparison methods proposed herein is detailed in Section 5.1, with the best comparison method then evaluated for its fitness as part of an adapted inter-coder agreement coefficient in Section 5.2. Next, comparison methods and agreement results from the previous experiments are used to evaluate automatic segmenters in an experiment to provide an overall experimental evaluation and a demonstration of which comparison method is most suitable for segmentation evaluation in Section 5.3.

Finally, Chapter 6 shows a discussion on how to report segmentation comparison methods and a conclusion and overview of future work is presented in Chapter 7 summarizing the results of Chapter 5 which reiterates the choice of which of the four boundary edit distance normalizations should be used to replace WindowDiff and P_k as a comparison method for segmentation.

Chapter 2

Background

2.1 Linear Segmentation Comparison Methods

To evaluate either automatic or human segmenters, a method of comparing one segmentation to another is required which demonstrates the differences between any two segmentations. When one of these segmentations is assumed to be a correct solution (i.e., reference) for a segmentation task, the difference between it and a hypothetical segmentation is considered to quantify the amount of error that a hypothetical segmenter has committed. This section explores the variety of metrics and methods that have been used to quantify the error between a reference and hypothesis segmentation (where one "true" segmentation is assumed to be a reference).

Information retrieval metrics Segmentation has been evaluated by many early studies[1] using information retrieval (IR) metrics such as precision, recall, and accuracy. To use IR metrics, the task of segmentation was conceptualized much like binary classification tasks are: two classes exist—boundary and non-boundary—for each potential boundary position. A potential boundary position is a space between two adjacent atomic textual units (e.g., words, sentences, paragraphs, etc.), where atomicity (i.e., granularity) of a unit is defined by the end task for which segmentation is being used—giving us $n-1$ potential boundaries for n atomic units of text (Passonneau and Litman, 1993, p. 150). Passonneau and Litman (1993) illustrated a confusion matrix which represented segmentation as a classification task (Table 2.1), showing how to count the number of true positives

[1] Passonneau and Litman (1993); Hearst (1994); Litman and Passonneau (1995); Hearst (1997); Reynar and Ratnaparkhi (1997); Yaari (1997) and Passonneau and Litman (1997).

		Actual	
		Boundary	Non-boundary
Predicted	Boundary	TP	FP
	Non-boundary	FN	TN

Table 2.1: Confusion matrix for potential boundary positions of linear segmentations; adapted from Passonneau and Litman (1993)

(TP), false positives (FP), false negatives (FN), and true negatives (TN). A TP occurs when both a hypothetical (i.e., automatic) and reference (i.e., manual) segmenter place a boundary at the exact same position in a document, whereas if only the hypothetical segmenter does so, a FP has occurred. A TN occurs when a boundary has not been labelled at a particular position by both the hypothetical and reference segmenter, whereas if only the reference segmenter does so, a FN has occurred. The sums of the various rows and columns of this confusion matrix then represent the number of boundaries ($|\mathrm{B}|$) and non-boundaries ($|\neg\mathrm{B}|$), respectively, with a total number of $n-1$ potential boundaries ($|\mathrm{B}| + |\neg\mathrm{B}| = n - 1$).

Using this confusion matrix, Passonneau and Litman (1993); Litman and Passonneau (1995) and Passonneau and Litman (1997) calculated a variety of IR metrics, including *precision*, *recall*, *fallout*, and *error* (Equations 2.1–2.4). These metrics served to try to quantify the types of errors that occurred. Precision measured how often a method would choose a correct boundary position out of those chosen whereas recall measured how often a method chose a correct boundary position out of all possible correct answers. Fallout measured how often incorrect boundaries were identified by a hypothetical segmenter, and error measured the total number of incorrect boundaries and non-boundaries identified.

$$\text{Precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \qquad (2.1) \qquad \text{Fallout} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}} \qquad (2.2)$$

$$\text{Recall} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \qquad (2.3) \qquad \text{Error} = \frac{\mathrm{FP} + \mathrm{FN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}} \qquad (2.4)$$

When evaluating performance, an ideal segmenter would balance both precision and recall (i.e., a segmenter should strive for high precision and high recall in many applications). Beeferman et al. (1997, p. 42) regarded the complementary nature of precision and recall as a flaw, and desired one number, not many. For this reason, a mean of the two metrics was provided by van Rijsbergen (1979, p. 174) in his "effectiveness measure", otherwise known as F_β-measure (Equation 2.5).

$$\begin{aligned} \mathrm{F}_\beta\text{-measure} &= (1+\beta^2)\cdot\frac{\text{Precision}\cdot\text{Recall}}{(\beta^2\cdot\text{Precision})+\text{Recall}} \\ &= \frac{(1+\beta^2)\cdot\mathrm{TP}}{(1+\beta^2)\cdot\mathrm{TP}+\beta^2\cdot\mathrm{FN}+\mathrm{FP}} \end{aligned} \tag{2.5}$$

Using F_β-measure, a single metric was able to be shown which balanced those two performance metrics, allowing for easier comparisons of segmenters to balance both precision and recall and compare only one performance value against another. Although F_β-measure and other IR metrics are widely applied as a performance metric in natural language processing, the usage usage of IR metrics present their own problems when applied to segmentation, unfortunately. When a number of manual segmentations were produced for a corpus, it was realized that manual segmenters often did not fully agree with each other (see Section 3.2). For tasks were this disagreement occurred, the validity of the assumption that there exists one "true" reference segmentation that can be compared against does not hold true, making these metrics (e.g., IR metrics) that rely upon having a single reference segmentation to compare against difficult to interpret when applied to multiple manual segmentations. Additionally, one type of error which IR metrics fail to account for was found to be very frequent: coders would place a boundary adjacent with, but not directly upon, the position chosen by another coder (i.e., nearly missing a boundary).

Near-miss errors Many studies report low inter-coder agreement coefficients due to a fact, "known to researchers in discourse analysis from as early as Levin and Moore (1978), that while [coders] generally agree on the 'bulk' of segments, they tend to disagree on their exact boundaries" (Artstein and Poesio, 2008, p. 40). This effect is often referred to as a near-miss error. This realization, and the knowledge that coders often have high disagreement with each other, has led to the development of a variety of segmentation comparison methods which attempt to award partial credit for near-misses, because "in almost any conceivable application, a segmenting tool that consistently comes close—off by a sentence, say—is preferable to one that places boundaries willy-nilly". IR metrics do not award partial credit for such near-misses, and instead can award to arguably better segmenters "worse scores than an algorithm that hypothesizes a boundary at every [potential boundary] position. It is natural to expect that in a segmenter, close should count for something." (Beeferman et al., 1997, p. 42).

Comparison to a majority solution Many studies[2] that used IR metrics to evaluate automatic segmenters used multiple human annotators (i.e., coders) to code boundaries, and found that their coders did not completely agree with each other, demonstrating that there is often no one "true" reference segmentation to compare against. A solution to the problem of near-misses of boundary locations was still required, and the ability to have one metric to summarize segmentation results was desired. To solve this issue, a reference solution was composed by these studies of boundaries upon which a majority of coders agreed. From this majority, all IR metrics could be computed against it as a reference. For each boundary position, if the majority of the coders observed a boundary at that position, then a boundary was placed in the majority solution coding; otherwise, a boundary was not. This proved problematic, as Passonneau and Litman (1993) and Passonneau and Litman (1997) first defined a majority as 4 of 7 coders, but then re-defined it as 3 of 7 in Litman and Passonneau (1995). This change was explained as being warranted because the differences between coders for combinations where the number of coders was ≥ 3 were not statistically significantly different—using Cochran's test, which Cohen (1960) points out is an inappropriate test to use for such a determination (see Section 3.2). Hearst (1993, p. 6) also used a majority solution to compare against, which she called a 'consensus' opinion, which was comprised of boundaries which the majority ($> 50\%$) of coders agreed upon, but added the condition that the average agreement between coders should be $\geq 90\%$ to trust this majority segmentation.

Regardless of whether statistical significance is demonstrated for whether coders that comprise a majority solution significantly differ from each other in their segmentations or not, a majority solution will invariably produce results difficult to interpret when the threshold of what constitutes a "majority" has had so many different interpretations ($^4/_7$, $^3/_7$, $> 50\%$). Hearst (1994, p. 30) points out that the conflation of a number of codings does not constitute an objectively determined, or 'real', boundary for segmentations tasks such as topic modelling. Hearst (1994, p. 30) renounced the usage of a majority reference segmentation for segmentation evaluation, stating that "a simple majority does not provide overwhelming proof about the objective reality of the subtopic break. Since readers often disagree about where to draw a boundary marking for a topic shift, one can only use the general trends as a basis from which to compare different algorithms".

It has been demonstrated that comparison to a majority solution is too fraught with issues to be a viable evaluation method to adapt IR metrics or other segmentation

[2]Passonneau and Litman (1993); Hearst (1994); Litman and Passonneau (1995); Hearst (1997) and Passonneau and Litman (1997), etc.

comparison methods. With no consistent definition of what constitutes an acceptable majority and with the issue of over-inflated values, the use of a majority solution has been predominantly abandoned in literature.

The difficulty of using a majority solution was also realized by researchers trying to determine the reliability of their coding schemes by measuring inter-coder agreement. The percentage agreement statistic (See Section 3.2) required a comparison against a majority solution to measure inter-coder agreement. Between pairs of coders, this metric could also be used as a simple comparison metric between coders. The usage of a majority solution, unfortunately, grossly inflated the statistic's values, because "the statistic itself guarantees at least 50% agreement by only pairing off coders against the majority opinion"(Isard and Carletta, 1995, p. 63).

Artificial segmentation data To attempt to circumvent the issue of creating a single reference segmentation from many manual codings, some authors constructed artificial segmentation data. Phillips (1985) used the subtopic structure of chapters in science textbooks to represent reference boundaries, Richmond et al. (1994, pp. 51–53) concatenated articles from an edition of The Times newspaper and attempted to reconstruct the boundaries between articles, Beeferman et al. (1997) also used concatenated articles but from the Wall Street Journal (WSJ) corpus and the Topic Detection and Tracking corpus collected by Paul and Baker (1992) and Allan et al. (1998) respectively, Reynar (1998) used a subset of the WSJ corpus with concatenation, and Choi (2000) used 700 artificial texts, where a text is comprised of ten text segments selected from the first n sentences of a randomly selected document in the Brown corpus (Francis, 1964). This article concatenation is, however, artificial, and with that property come some undesirable side effects.

The usage of artificially constructed segmentation data attempts to provide a reference segmentation to compare against, but this artificial segmentation data did not guarantee that the articles chosen were wholly unrelated to each other (one article could naturally lead into another in its choice of topic). There is also no guarantee that the properties of artificially concatenated text correlate with that of the natural text it attempts to emulate, e.g., the subtlety between topic and subtopic breaks that occur in natural text (meaning that artificial data may significantly simplify a segmentation task). This concatenation may alter features of the text which are inconsistent with how natural text is composed and the resulting probabilities of a manual coder placing a boundary at a particular position (e.g., for topical segmentation, artificial segmentation data may

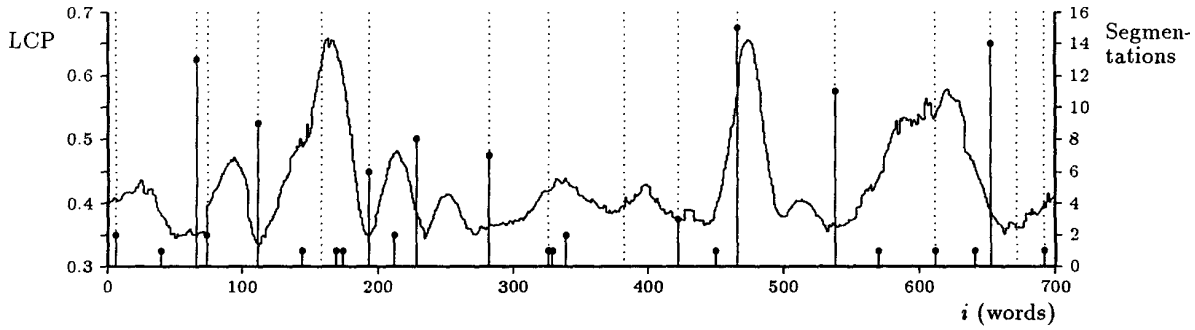


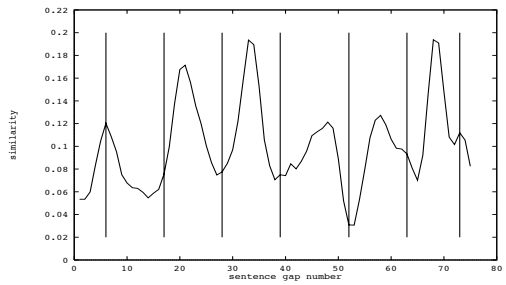
Figure 2.1: Correlation between a histogram of 16 manual segmentations and the lexical cohesion profile of the simplified version of O. Henry's 'Springtime á la Carte' (Kozima, 1993, p. 288)

contain a differing frequency of a particular type of cue phrase that coders may use to determine where topical shifts occur). Additionally, for artificial corpora where boundaries are defined as chapter or section breaks denoted by a document's author, these boundaries represent only one individual's segmentation of the text, or one coding, and they are not strictly artificial[3]. Multiple codings of the same text by a variety of individuals may produce different segmentations, meaning that we must thoroughly study the subjective notion of a boundary before we decide upon a single reference for usage in an application.

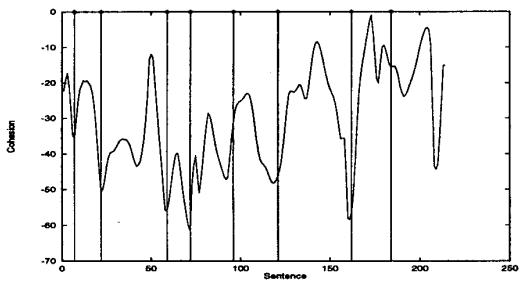
Qualitative analysis of manual codings Given the drawbacks of using artificial data, and the issues involved when using a majority solution against which to calculate statistics and perform a quantitative evaluation of a segmentation method, Kozima (1993) used the same sort of data, but performed a qualitative evaluation of a proposed segmentation method. Kozima (1993, p. 288) collected 16 segmentations of the simplified version of O. Henry's 'Springtime á la Carte' (Thornley, 1816). A frequency representation of the boundaries that these coders placed at particular positions were represented as a histogram upon which was graphed the cohesion of the text using Kozima's (1993) segmentation method, lexical cohesion profile (LCP), as shown in Figure 2.1.

As stated by Hearst (1994, p. 30), this sort of analysis does not inform us of whether an automatic segmenter is able to identify objective topic or subtopic breaks, but instead is able to inform us of whether a method correlates with some general trends. Kozima (1993, p. 288) presented this qualitative analysis of the text cohesion 'valleys' as correlating

[3]More often than not, a document's headings and subheadings structures denoting topical sections are the conflation of both an author and editor's boundary decisions, or many authors and editors, whose rationale for such boundary choices is neither controlled nor does it represent one single human opinion of what constitutes a segment for subjective tasks such as topical segmentation.



(a) Consensus topic boundaries chosen by human coders (vertical lines) and TextTiling's text similarity for a 77-sentence popular science article "Magellan" (Hearst, 1993, p. 2)



(b) Topic boundaries chosen by a paper's author (vertical lines) and text cohesion calculated for a 200 sentence psychology paper (Richmond et al., 1994, p. 53)

Figure 2.2: Correlations between manual segmentations (vertical lines) and text similarity/cohesion metrics from (Hearst, 1993, p. 2) and (Richmond et al., 1994, p. 53)

with many of the high frequency manually annotated boundaries, but finally came to the conclusion that not all boundaries appear to correlate with LCP.

A similar qualitative analysis was performed by Hearst (1993, p. 2) when she compared the text similarity metric used by TextTiling against manual segmentations. Instead of using a frequency representation of coder boundaries, Hearst used only those boundaries that represented a "consensus" opinion ($> 50\%$ of the coders agreed upon a boundary, where the average coder agreement was $> 90\%$), represented as vertical lines plotted with TextTiling's text similarity metric for a 77-sentence popular science article referenced as "Magellan" in Figure 2.2a. The same qualitative analysis was also performed by Richmond et al. (1994, p. 53) when they compared their text cohesion metric with manual segmentations for a 200-sentence psychology paper segmented by its author, as shown in Figure 2.2b.

Beeferman et al. (1997, p. 45) also utilized a very similar form of qualitative visual correlation analysis as those previously mentioned, as shown in Figure 2.3. They produced many such charts for the artificial segmentation data sets they used and employed two sets of vertical lines: one to indicate their segmenter's boundaries and another to indicate the artificial (i.e., reference) segment boundaries. They then plotted their method's confidence score for each potential boundary as a horizontal line.

These qualitative analyses are very informative when developing a text segmentation method which is based upon measuring the change in different properties of text over varying text locations. These comparisons can be used to tune segmentation method parameters (Hearst, 1997) and to try to offer a visual explanation of how some methods

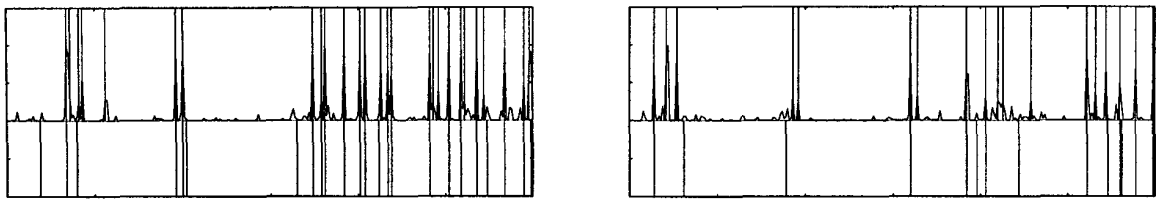


Figure 2.3: Correlations between hypothetical segmentations (upper vertical lines), reference segmentations (lower vertical lines), and the probability of the automatic segmenter placing a boundary on WSJ data (Beeferman et al., 1997, p. 45)

work and how they correlate with manual segmentations, but in the end, they do not offer anything more than qualitative comparison against a majority trend. In addition to these sorts of analyses, methods of quantifying the degree to which a correlation exists, or the degree to which automatic segmenters disagree with manual segmentations are still needed to make clear conclusions about the performance of an automatic segmenter in terms of human performance.

Near-miss errors within a window The definition of what constitutes a matching boundary, or a true positive, for IR metrics can be adjusted to account for near-miss errors. Hearst (1993, p. 6) redefined a matching boundary as a hypothetical boundary that falls within 2 sentences of a consensus boundary and Reynar (1994) redefined it as a boundary which falls within 3 sentences of a reference boundary (where an artificial corpus of concatenated articles from The Times was used), and reported IR metrics with and without the inclusion of these near-misses. Richmond et al. (1994) reported results upon a variety of sentence distances of a reference boundary (where an artificial corpus of concatenated WSJ articles was used). During this evaluation, Richmond et al. found that their method "located 53% of the article boundaries precisely and 95% of the boundaries to within an accuracy of a single sentence. Every article boundary was identified to within an accuracy of two sentences." Reynar (1998, Chapter 5) goes even further, and reports results for a variety of methods where the definition of a match has a tolerance of 0, 20, 40, 60, 80, or 100 words (again using an artificial corpus of concatenated WSJ articles).

Windows around reference boundaries where a boundary was considered to be correct provided a useful way to adapt IR metrics to account for near misses. This technique is not without issues, however. These windows were inconsistently defined and must take into account the granularity chosen (e.g., being off by 1 unit in a paragraph-level segmentation does not translate well to an application where the desired segmentation unit differs, e.g.,

		Boundary Errors			
		Deleted	Inserted	Off by > 2	
Texts	Magellan	1	1	1	3
	Earth	0	1	2	3
	Sequoia	1	2	2	5
		2	4	5	11

Table 2.2: Errors produced by TextTiling when compared to manual consensus segmentations with a hypothetical boundary within 2 reference sentences being considered a match; adapted from Hearst (1993, p. 6)

as a sentence or word unit). These methods also meant that if an automatic segmenter placed multiple boundaries within the window around only one reference boundary, they would be awarded a match, or true positive, for each surrounding boundary, making it easy for degenerate segmenters to achieve inflated scores using some IR metrics.

Errors as addition/deletion operations Another way to quantify the errors that a segmentation method produces, but to also account for near misses, is to measure the number of boundaries that are erroneously deleted, inserted, or off by a number of units. Hearst (1993, p. 6) included a frequency count of the number of times her segmentation method TextTiling differed from the majority opinion of manual segmentations for 3 articles (shown in Table 2.2). The frequency counts of deleted versus inserted boundaries corresponds to the number of false negative and false positive boundaries placed, respectively, and the inclusion of the number of boundaries placed that are off by > 2 sentences together present a detailed picture of the types of errors that segmenters make, and their relative frequencies. Richmond et al. (1994) also quantified errors in this way when comparing their automatic and manually annotated psychology paper in the form of a table where columns represent boundaries and rows indicate whether they were placed by the manual or automatic segmenter, with symbols in the bottom row to indicate whether it was an additional (+), omitted (−), or off by n (digit) boundary (shown in Table 2.3). By showing the alignment chosen for boundaries in the associated table, Richmond et al. (1994) also demonstrated the need to make a judgement as to whether a boundary is off by n, or is actually an added boundary, which is a distinction that would be required to calculate a minimum edit distance between two segmentations. Overall, these three types of errors are valuable when identifying how to improve a segmenter, showing which errors are most pronounced, and which should be minimized in subsequent iterations of the development of an automatic segmenter.

Boundary	Hypothesis	7	22	42	58	72	77	92	118	137	156	161	177	184	191
positions	Reference	7	22	-	59	72	-	96	121	-	-	162	-	184	-
	Error	0	0	+	1	0	+	4	3	+	+	1	+	0	+

Table 2.3: Topic boundary positions chosen by a paper's author and Richmond et al.'s (1994) automatic segmenter upon a 200 sentence psychology paper denoting additional (+), omitted (−), or off by n errors; adapted from Richmond et al. (1994, p. 53)

Beeferman et al. (1997) went further than using edit-like operations to discuss types of errors, and mused that one could "come up with an explicit alignment between the segments proposed by [an] algorithm and the reference segments, and then to combine insertion, deletion, and substitution errors into an overall penalty." (Beeferman et al., 1997, p. 42). Unfortunately, they did not pursue this type of an error metric, probably because it offered no obvious way to award partial credit for near-miss errors (because off-by-n errors as in Richmond et al. 1994, p. 53 were not considered). To partially credit near-miss errors, and to create a penalty-based measure, Beeferman et al. revisited the concept of a window-based analysis of segmentations to account for near-miss errors.

Quantifying segmentation error in terms of windows Beeferman et al. (1997, p. 43) proposed a measure with the premise that an automatic segmenter should optimize the likelihood that two units of text (i.e., words, sentences, paragraphs, etc.) are "correctly labelled as being related or being unrelated". To that end, they proposed a measure named P_μ, which is "the probability that two sentences drawn randomly from the corpus are correctly identified as belonging to the same document or not belonging to the same document", or, more formally, for a reference segmentation (R) and hypothesis segmentation produced by an automatic segmenter (H), where P_μ is calculated as shown in Equation 2.6.

$$P_\mu(R, H) = \sum_{1 \le i \le j \le n} D_\mu(i, j) \cdot \big(\delta_R(i, j) \mathbin{\bar{\oplus}} \delta_H(i, j)\big) \tag{2.6}$$

In Equation 2.6, δ_R "is an indicator function which is 1 if the two corpus indices specified by its parameters belong in the same document, and 0 otherwise"; δ_H "is 1 if the two indices are hypothesized to belong in the same document, and 0 otherwise". "The $\bar{\oplus}$ operator is the XNOR function ('both or neither')" applied to its operands, δ_R and δ_H, which essentially serves as a coefficient of 1 when indices in both the reference and hypothesis solution are in the same document (meaning a correct adjacent text unit was

identified) or both are not, and zero otherwise. The function D_μ is a "distance probability distribution over the set of possible distances between sentences chosen randomly from the corpus", which is used to represent the text in a document. For the purpose of focusing upon small distances such as pairs of sentences, they define D_μ to be "an exponential distribution with a mean $^1/_\mu$, a parameter that [is fixed] at the appropriate mean document length for the domain" and γ_μ is chosen so that D_μ "is a probability distribution over the range of distances it can accept", as shown in Equation 2.7.

$$\mathrm{D}_\mu(i,j) = \gamma_\mu e^{-\mu|i-j|} \tag{2.7}$$

P_μ is defined as a probability, producing values with a range of $[0,1]$, and only awards a value of 1 for a perfect score. Using its distance probability distribution, it awards partial credit for near-misses (i.e., returning groups of sentences with a high probability of occurring in the same document). This method presents an overall assessment of the number of errors that were committed by an automatic segmenter as one single number while awarding partial credit for near-missed boundaries, but the usage of a probabilistic distance was difficult to rationalize; "one weakness of the [measure] as we have presented it here is that there is no principled way of specifying the distance distribution D_μ".

Doddington (1998) attempted to solve this issue of better specifying the distance probability distribution in his modification of the penalty-based measure for the Topic Detection and Tracking TDT Phase 2 (TDT2) evaluation plan. Two important changes where made: "the boundary test (as to whether the two sentences/words/times belong to the same story) is made at a fixed distance rather than a probabilistic distance", and the error measure was reformulated as the probability that a segmenter would omit a boundary (FN) or insert an additional boundary (FP). This formulation did not come into wide usage, but it informed a later adaptation of P_μ which did become widely adopted.

2.2 Inter-Coder Agreement

Most segmentation tasks in natural language processing—at the beginning of the investigations into a specific task—have only one known type of segmenter: human. After investigation, humans remain as the only natural type of segmenter—supplemented, eventually, by automatic segmenters. To emulate human segmenters it is necessary to study how humans perform in natural language segmentation tasks. This study requires the analysis of examples of segmentation, which are produced during sessions where

humans are issued segmentation instructions, corpora, and time to complete such a task.

Experiments studying phenomena present in text are comprised of a variety of components and stages. First, the phenomenon of interest is identified, e.g., the nature of what constitutes a topic in writing. A specific and concrete question is defined which would denote this phenomenon, e.g., what denotes a topic shift in popular scientific writing? This question gives us a particular effect to elicit and study: the subjective placement of boundaries denoting topical shifts in a specific document. Our goal then is to see whether this subjective effect can be elicited in a group of individuals. If it can be elicited reliably, then one has discovered something which may be a general effect common amongst the population of individuals one has asked to participate in our study (i.e., coders).

Given a specific question to answer, and knowing the effect one wishes to elicit, a method of doing so must be devised. First, one develops a coding scheme to attempt to recognize when or where the phenomenon occurs and to identify or label its various parts (e.g., boundaries denoting topic shifts), and distinguish between various types/components of larger parts (e.g., minor versus major topic boundaries, or hierarchical topic structures). One then needs a set of instructions that when followed will cause a coder to produce the specific phenomenon one desires (e.g., read an article keeping in mind when topic shifts occur) and inform them of how to apply the coding scheme (while reading, place a mark between sentences/paragraphs where you believe that a topic shift has occurred). These instructions define our task. One also needs to select a representative corpus from which to elicit this effect which is appropriate for our available coders and task. The results of a set of coders applying a coding scheme from given instructions upon a corpus should be a set of data which represents the effect one wishes to observe.

When collecting observations in this manner, one must be very careful in defining the effect that one wishes to observe, crafting the instructions to not overly bias the result or detract from the subjectivity one wishes to observe, and defining a coding scheme that will completely and unambiguously label the effect. How does one know that the instructions conveyed the task sufficiently? How can one trust that the human segmenters each reliably produced examples of the phenomena that one is studying? How does one know that they were capable of applying the coding scheme? How does one know that the experiment can be reproduced by another researcher using the same coding scheme and that the experiment has not been overly biased by the instructions, coding scheme, interpretations of the initial researchers, or another unknown bias? To begin to answer these questions, one must look at the agreement that occurs between humans performing these tasks (i.e., coders), and asses their reliability at applying a specific coding scheme

(Craggs and McGee Wood, 2005, p. 289).

What does reliability mean? The measurement of agreement to assess the reliability of text annotations provided by coders has had a variety of interpretations in computational linguistics (Craggs and McGee Wood, 2005, p. 290). It has been claimed that such measurements "assess labelling accuracy," (Jurafsky et al., 1997), that they demonstrate in coders "the objectivity of [their] decisions" (Di Eugenio and Glass, 2004), or the degree of our understanding of coder decisions (Carletta, 1996). Craggs and McGee Wood (2005, p. 290) instead assert that the...

> "...intended meaning of reliability should refer to the degree to which the data generated by coders applying a scheme can be relied upon. If we consider the coding process to involve mapping units of analysis onto categories, data are reliable if coders agree on the category onto which each unit should be mapped."
>
> – Craggs and McGee Wood (2005, p. 290)

In short, if coders are given the same instructions and apply the same coding scheme differently, then the coding scheme may not truly represent a fully constructed description of the phenomenon being studied. A contrived example of this is the hypothetical coding of the interpretation of the word 'Okay' by a number of coders. If one applies the coding-scheme categories 'Acknowledgement' and 'Accept', but coders cannot reliably code both of these categories, then the coding scheme may not be appropriate (e.g., there may be a third, as of yet unpredicted interpretation of 'Okay' which is inconsistently being labelled as either 'Acknowledgement' or 'Accept' such as 'Interrogative'[4]). Craggs and McGee Wood (2005, p. 290) assert that if reliability of a coding scheme can be demonstrated, that establishes that the data has two important properties:

1. "The categories onto which the units are mapped are not inordinately dependent on the idiosyncratic judgements of any individual coder"; and
2. "There is a shared understanding of the meaning of the categories and how data are mapped onto them."

The first property is required to ensure that a coding scheme and annotation effort is reproducible, and that agreement is not overly biased (negatively or positively) by a

[4]The inflection given to the word 'Okay' in speech modify 'Okay' to be an interrogative question.

minority within a group of coders. The second property is required to be able to trust any analysis derived from the codings, and it means that the categories are an accurate representation of the phenomenon under study.

How can one trust that each coder reliably produces examples of the phenomena that one is studying? One cannot completely trust that coders have understood the instructions given to them. One can, however, infer the degree to which one can reasonably trust one's coders given their expertise. If a coder is asked to topically segment a document, and in addition to their instructions also has also a high level of expertise in the task (they have studied or even published articles on text segmentation, topic modelling, etc. which suggests their great understanding of the entire task which they have been asked to perform). "Expert" coders can be compared to those with unknown, or little expertise (i.e., naïve coders), to qualitatively assess the general replicability of the instructions and coding scheme used (Isard and Carletta, 1995, p. 64). Additionally, by comparing expert coders to naïve coders, one can spot any "odd men out", one's outliers, which one can then exclude to obtain a highly uniform (thought not necessarily high quality) set of codings (Isard and Carletta, 1995, p. 64). Looking solely at the naïve coders themselves, "if the naïve coders agree among themselves better than they agree with the expert, it could be that the instructions mean something, but not what the expert intended!" (Isard and Carletta, 1995, p. 64). All of these comparisons are made by comparing the agreement between one group and the next, which requires a statistic which is comparable from group to group regardless of whether the number of coders in a group differs or not.

Percentage agreement A need to ascertain the agreement between coders for segmentation was recognized by both Passonneau and Litman (1993) and Hearst (1993). In their 1993 work, Passonneau and Litman identified that "the correspondence between discourse segments and more abstract units of meaning is poorly understood. . . ". Beginning with 20 narrative monologues from a movie, they asked 7 coders[5] to segment this discourse "using speaker intention as a criterion" (Passonneau and Litman, 1993, p. 149), or more specifically, to place a boundary when a speaker finished a communicative task and began a new one. Their coders produced one phrase-level[6] linear segmentation for each of the 20

[5]Psychology students at the University of Connecticut.

[6]Though "restricted to placing boundaries between prosodic phrases" (Passonneau and Litman, 1993, p. 149).

monologues of approximately 700 words. They considered that using such a phrase-level granularity for their task was perhaps too coarse, in that it would make the coders more likely to agree upon each other, but that "using smaller units would have artificially lowered the probability for agreement on boundaries." (Passonneau and Litman, 1993, p. 149).

To measure agreement, Passonneau and Litman (1993, p. 150) adapted the percentage agreement metric of Gale et al. (1992, p. 254) for usage in segmentation.[7] In this metric, agreement is the ratio of the total observed agreement of a coder with the majority opinion[8] for each boundary position over the total possible agreements with the majority (i.e., Equation 2.8).

$$\text{Percentage agreement} = \frac{|\text{Boundary Positions}_{\text{coder}} \cap \text{Boundary Positions}_{\text{majority}}|}{|\text{Boundary Positions}_{\text{majority}}|} \quad (2.8)$$

This agreement measure unfortunately fails to take into account chance agreement, which by Passonneau and Litman's admission could have been caused by the granularity level chosen versus that which the coders may have intuitively used. Another issue with percentage agreement is that it does not explicitly try to account for the prevalence of near-misses of boundary placements that often occur during segmentation. This issue was probably thought to have been mitigated by performing comparisons against a set of majority, agreed upon, boundaries (of which the segmentation being compared to this majority voted for), and also by normalizing this number of matching boundaries by the number of boundaries in the majority set.

Statistical significance of coder difference Passonneau and Litman (1993, pp. 150–151) appeared to not be wholly satisfied with their percentage agreement measure, and embarked upon determining whether the coders statistically significantly differed from each other in their boundary placements. To accomplish this, they represented the segmentations of each coder in an $i \times j$ matrix "where $i = 7$ coders and width $j = n - 1$" prosodic boundaries.[9] In each cell of the matrix a 1 was placed when a coder indicated a boundary at that potential boundary position, and a 0 if they did not (see Table 2.4).

To determine statistical significance, Cochran's test (Cochran, 1950) was applied

[7] Hearst (1993) appears to have used a similar if not identical metric.

[8] If $^4/_7$ coders agreed upon a boundary it was deemed to be the majority, but in Litman and Passonneau (1995) this was relaxed to $^3/_7$ coders.

[9] The total number of potential boundary positions is $n - 1$, where for their task n is the number of prosodic phrases in a transcript(Passonneau and Litman, 1993, p. 150).

	Potential Boundary Positions					
Coder	1	2	3	...	$n-2$	$n-1$
1	0	0	0	...	0	0
2	0	1	0	...	0	0
3	0	1	0	...	0	0
4	0	1	0	...	0	0
5	0	0	1	...	0	0
6	0	1	0	...	0	0
7	0	1	0	...	0	0

Table 2.4: Hypothetical matrix of potential boundary positions per coder for a segmentation as used to determine statistical significance using Cochran's test (Cochran, 1950) in Passonneau and Litman (1993, pp. 150–151)

"to evaluate the significance of differences across columns of the matrix" of which there are c columns, row totals are u_i, and the column totals are T_j in Equation 2.9. In this formulation, "$\mathcal{Q}$ approximates the χ^2 distribution with $j-1$ degrees of freedom (Cochran, 1950)" (Passonneau and Litman, 1993, p. 151). Using this test, Passonneau and Litman (1993, p. 151) determined that agreement between their coders was "extremely highly significant" ($p < 0.1 \times 10^{-5}$).

$$\mathcal{Q} = \frac{j(j-1)\sum T_j - \bar{T}^2}{c\big(\sum u_i\big) - \big(\sum u_i^2\big)} \tag{2.9}$$

This agreement came with the caveat that the proportion of non-boundaries agreed upon by most coders was higher than the proportion of boundaries that they agreed upon. This caveat is troubling because it indicates that this significance test of agreement is potentially artificially inflated by combining the two agreements, upon boundary and non-boundary positions, together. It is also regarded as an inappropriate measure of whether coders agree because, as pointed out by Cohen (1960), the χ^2 distribution which it approximates is a measure of association as opposed to agreement (Artstein and Poesio, 2008, p. 5).

Hypothesis testing, confidence intervals, and standard error of coefficients Inter-coder agreement coefficients such as κ and π do not reflect sampling error, but are intended to "generalize the findings of a reliability study to a population of raters" (Sim and Wright, 2005, p. 265). They do not allow us to perform hypothesis testing to determine whether coders perform better than chance. It is, however, possible to determine whether

such coefficients differ significantly from a value of zero (Bakeman and Gottman, 1997, pp. 64–65). Cohen (1960, pp. 43-44) implies that when the number of items coded is large (i.e., 100 items or greater), then "the sampling distribution of κ will approximate normality", allowing us to set confidence intervals, and a hypothesis test can be performed between two independent κ values.

> However, "it needs pointing out that it is generally of little value to test κ for significance as it is for any other reliability coefficient—to know merely that κ is beyond chance is trivial since one usually expects more than this in the way of reliability".
>
> – Cohen (1960, p. 44)

Even demonstrating statistical significance from random agreement (a coefficient value of zero) is often uninteresting because "quite low values of κ often turn out to be significant" (Bakeman and Gottman, 1997, p. 66), and such coefficients are instead meant to ensure "that the coders do not deviate too much from perfect agreement (Krippendorff, 2004a, p. 237)", not that they do deviate from chance agreement (Artstein and Poesio, 2008, p. 29). Reporting such, however, should not be harmful as long as it is not the sole statistic calculated.

The variance, standard error, and confidence intervals of the coefficients described have been defined in a variety of works. The standard error of a coefficient is the standard deviation of the sampling distribution of a coefficient (Everitt, 2003), and it gives us an indication of the variance of a coefficient, or at least the lower bound, which can be used to establish confidence intervals. Standard error for Cohen's κ has been described by Fleiss et al. (1969), and for Fleiss' multi-π by Siegel and Castellan (1988, Section 9.8.2) in their description of K and by Gwet (2008, pp. 40–42) and by using the jackknife technique (Miller, 1964) by Kraemer (1980, pp. 210–211). Krippendorff (2004a, Section 11.4.2) states that the distribution of α is unknown, so confidence intervals must be obtained by bootstrapping. Bootstrapping could also provide a means for estimating standard error on other coefficients, though it comes with its own set of drawbacks and assumptions, and, if enough data is already present, parametric methods should be preferred.

What coefficient value indicates a reliable coding scheme? When interpreting these coefficients, besides the obvious interpretation of the zero value and the greater than or less than zero ranges, some arbitrary guidelines have been proposed. Krippendorff (2004a, pp. 241–242) prescribes that, for content-analysis coding schemes, only

variables having a coefficient value[10] of ≥ 0.800 should be relied upon for usage, and for values > 0.667 and < 0.800 that only tentative conclusions should be drawn about the relationship between the phenomenon measured and some other characteristic of the data. Krippendorff gave these ranges with considerable hesitation, and noted that any guidelines for interpretation should be related to the "the costs of drawing wrong conclusions" (Krippendorff, 2004a, p. 242), and that the choice of a reliability criterion (i.e., coefficient value above which is deemed to be reliable) should "correlate with the conditions under which one is willing to rely upon imperfect data" (Krippendorff, 2004b, p.6). Though the costs of drawing incorrect conclusions are not great in the field of computational linguistics (such an error is of greater consequence in a medical domain), any guidelines used should be informed by the specific task performed and the application of the codings. Other authors have given general arbitrary interpretation-guidelines for α including Landis and Koch (1977) and Fleiss (1981, p. 218), but each field—and sometimes each task in a field—generally establishes their own guidelines.

Despite Krippendorff's arbitrary guideline of ≥ 0.800 for reliability and > 0.667 and < 0.800 for tentative conclusions being espoused by many works in computational linguistics (Carletta, 1996; Di Eugenio and Glass, 2004; Krippendorff, 2004a), it is regarded as overly simplistic and naive by Craggs and McGee Wood (2005, p. 293), and unhealthy by Artstein and Poesio (2008, p. 58). Like Krippendorff, Craggs and McGee Wood (2005) asserted that studies should be more concerned with whether coefficients report a level of reliability that is suited to the degree to which "one is willing to reply upon imperfect data". Craggs and McGee Wood (2005, pp. 293–294) look at two use cases for coded data: training a machine-learning algorithm, and corpus analysis. For training, because machine-leaning algorithms depend upon the ability to recognize patterns in data, any level of "noise" that would obscure such patterns in less perfect (i.e., less than reliable) data would degrade performance considerably. Low reliability in a coding scheme which is to be used to produce data for training upon is then intolerable (Craggs and McGee Wood, 2005, p. 294). Content analysis, where a relationship is inferred between the phenomenon investigated and some property of the data, can perhaps tolerate less reliability in data, though "the conclusions that are gleaned from the analysis must be tempered according to the level of agreement achieved" (Craggs and McGee Wood, 2005, p. 294).

[10] In this case the coefficient is Krippendorff's α, but many works have applied this criterion to other coefficients and even outside of content analysis.

Averaging coefficients Although average inter-coder agreement coefficient values are often reported in computational linguistics, "it is a serious mistake to average the reliabilities of the variables of a complex instrument and take this average as a measure of overall data reliability. Computing averages assumes that higher values compensate for lower ones" Krippendorff (2004a, p. 242). This is usually taken to mean that the overall agreement of a study that measures agreement over multiple variables should not micro/macro average them, but also that the micro/macro averaging of multiple coefficients that even represent the same variable from documents of different length, differing numbers of coders, or number of items, is also an error. Micro averaging could be performed if the same coders coded different items for each statistic, but this would be redundant because in this situation multi-π, multi-κ, or Krippendorff's α all handle the combination of multiple coders and items, and Krippendorff's α can even handle missing items or coders.

2.3 Evaluating Segmenters

In an overall system which requires as one intermediate step the automatic segmentation of text, how does one choose the best automatic segmenter in this context? A method is needed which allows one to decide which segmenter best benefits an overall natural language processing system. To perform this selection, there are a variety of concerns that one must take into account, and empirical methods that can be applied to contextually determine which automatic segmenter is best for our task at hand.

Context when selecting a segmenter One could be providing an information retrieval (IR) system for audio and video broadcasts which helps a user select the correct broadcast by displaying small segments of a larger broadcast that are topically related (Franz et al., 2007), improving the semantic coherence of topical passages used for information extraction in a question answering (QA) system (Oh et al., 2007), using opinion topic identification to aid in fine-grained automatic opinion analysis (Stoyanov and Cardie, 2008), or using segmentation to aid multi-document summarization (Haghighi and Vanderwende, 2009). In each of these applications, multiple automatic topic segmenters can be evaluated, and it is not necessarily true that the same automatic topic segmenter would suffice for each application. One often desires a segmenter that is ideal in the context in which it is used and that one is confident will be able to generalize to unseen data after an evaluation is performed.

Ecological validity Ideally, when evaluating an automatic segmenter for a specific task one would evaluate its performance *in situ*. Instead of measuring the performance of an automatic segmenter in terms of how closely it replicates a given set of manual segmentations, one would place it into the overall system and assess the performance of the overall system with the variable being the segmenter chosen and controlling for all else. When measuring the end task performance one should ideally strive in our experiments to attain *ecological validity*, or "... the ability of experiments to tell us how real people operate in the real world" (Cohen, 1995, p. 102).[11] One should look not at how well the overall system performs upon replicating a manual solution during an experiment where the variable is the segmenter chosen, but instead one should assess the system at work in the environment where it is to be used. One should be watching real users conduct queries on an audio and video IR system, using a QA system, performing fine-grained opinion analysis, or using multi-document summarizers.

Ideally, both a quantitative and qualitative study should be performed to inform us of how such a system involving an automatic segmenter is used, or improves the daily work of its users. How does the system affect the users? Does it increase/reduce productivity? How do the users perceive the performance of the system? Are the users happier as a result of using the system (which can be affected by how frustrating some errors or difficulties in using the system are)? This form of evaluation is gaining popularity, especially in the area of automatic summarization, where evaluation of such systems have always been a difficult problem (e.g., He et al. 1999; Liu and Liu 2010; McCallum et al. 2012).

Unfortunately, evaluating an entire system that contains an automatic segmenter, and varying that segmenter, is a time-consuming process. Evaluating an entire system in place, with real users of the system, and in sufficient quantity and duration to make sound conclusions is even more expensive. Often there is too much time involved to test each subsystem (e.g., an automatic segmenter) and how it affects the overall performance of a system, let alone testing the entire system itself in such a rigorous and real-world battery of experiments and applications. In light of this expense, the best that can be done is to evaluate subsystems such as automatic segmenters upon their ability to reproduce a large number of manually produced solutions that are contextually relevant to the overall system (i.e., actual input and output expected to be encountered in daily use with sufficient variety to assess generalizability to different input).

[11] An ecologically valid evaluation form of task-based evaluation which essentially requires that human subjects provide the evaluation in an actual, i.e. real world and in no way artificial, setting.

Dataset	*Automatic Segmenters*		
	BayesSeg	MinCutSeg	APS
AI (Malioutov and Barzilay, 2006)	0.443	0.437	0.404
Clinical (Eisenstein and Barzilay, 2008)	0.353	0.382	0.371
Fiction (Kazantseva and Szpakowicz, 2011)	0.377	0.381	0.350

Table 2.5: Mean WindowDiff values of three topic segmentation methods over three data sets; adapted from Kazantseva and Szpakowicz (2011, p. 292)

Comparing segmenters by their reproduction of manual segmentation By evaluating whether an automatic segmenter can reproduce a large number of contextually and manually produced solutions (i.e., samples), one can begin to understand how such a segmenter may perform on unseen data, and how the various segmenters compare to each other. First, a method of comparing two arbitrary segmentations (e.g., P_k, WindowDiff, etc.) needs to be chosen. Using the method chosen, one can compute the method's value for each manual segmentation, where each text has one hypothetical segmentation produced by an automatic segmenter and one or more manually produced reference segmentations.[12] For each method, these statistic values can be combined into a sample mean—as in Table 2.5–which approximates how each method could be expected to perform according to the statistic used over the overall population containing unseen data (see Equation 2.10, where n is the number of independent samples, or manual segmentations, over which the mean is calculated).

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{2.10}$$

What traditionally constitutes a sample in segmentation evaluation is one manual segmentation of a document. If one has two segmentations of a document, these exist two samples. If one has two segmentations of two documents each, there exists four samples. A document is a very coarse unit of measurement, however, and if it is possible, the most ideal unit at which to measure performance would be per boundary pair (i.e., one member of this pair may be empty and the pair is comprised of components from two segmentations from two different segmenters). Since the task of a segmenter is to place boundaries, it would make sense to evaluate their placement of boundaries compared to

[12]Remember that finding one segmentation is often a subjective task with multiple interpretations, meaning that finding one "true" segmentation to compare against is unlikely, and so for a single piece of input text, multiple manual segmentations may have to be compared against.

another segmentation. A more coarse and abstract unit such as windows of segmentations is not as intuitive.

For the mean WindowDiff values shown in Table 2.5 (where the number of samples is equal to the number of documents because only one manual segmentation was provided for each document), note that the smallest (best performing) value is obtained by APS for two out of the three data sets. Kazantseva and Szpakowicz (2011, p. 292) identified, however, that only upon the fiction data set was their method (APS) statistically significantly better[13] than the other methods listed. What is statistical significance, and why is it important? To understand this, one must further discuss how a mean is calculated.

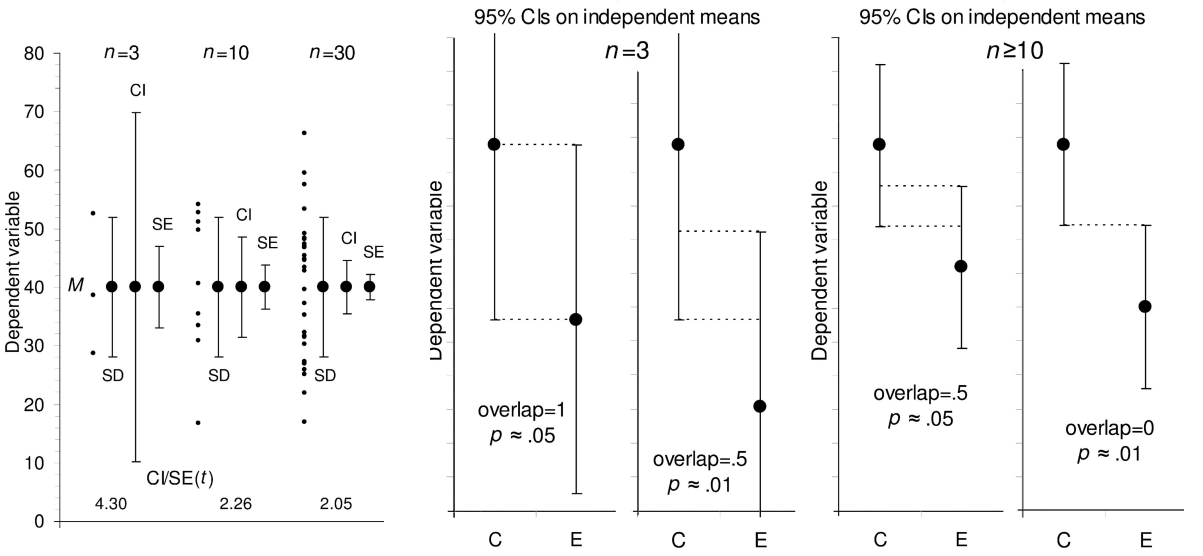
The means in Table 2.5 are simply estimates of what the real population mean μ would be if one were to calculate the mean using every possible topically segmented item (i.e., document), and these estimates are widely used in recent literature introducing new topic segmentation methods (e.g., Malioutov and Barzilay 2006; Eisenstein and Barzilay 2008; Kazantseva and Szpakowicz 2011). Unfortunately, this estimate is not accurate, and contains error because it is a small sample of a larger population. One can estimate the error of this data by calculating the standard error (SE) of this data set using Equations 2.11–2.12. As shown by standard error's formulation (Equation 2.12), as one increases the number of samples the error will also decrease, as shown in the SE bars in Figure 2.4a. This is because one has increased the proportion of the population that one has sampled, thereby giving us a more representative sample mean ($\bar{x}$).

$$\mathrm{SD}(\bar{x}) = \sigma(\bar{x}) = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}} \qquad (2.11) \qquad \mathrm{SE}(\bar{x}) = \frac{\sigma}{\sqrt{n}} \qquad (2.12)$$

The presence of this error in terms of the mean indicates that instead of having a single numerical value, one instead has a range, or interval within which one is confident that the true population mean is present. These confidence intervals (CI) are calculated as in Equation 2.13, where one can choose the degree of our confidence that the population falls within the specified range (e.g., 95% chance, or 95% CI) by choosing the relevant value in a table of critical values ($t_{(n-1)}$) for our given degrees of freedom (n) of Student's t distribution (Student (William Sealy Gosset), 1908).

$$\mathrm{CI}(\bar{x}) = \bar{x} \pm t_{(n-1)} \cdot \mathrm{SE} \qquad \text{where } t_{(n-1)} \text{ is a critical value of } t \qquad (2.13)$$

[13] Using a one-tailed paired t-test obtaining $p < 0.05$.



(a) Three different means (M, or $\bar{x}$) with varying n showing SD, 95% CI, and SE bars for comparison

(b) "Estimating statistical significance using overlap for 95% CI bars." Two different means with varying n are shown; each mean shows a baseline method and a competing method, and the overlap between the CIs is shown.

Figure 2.4: Sample error bars (Cumming et al., 2007, pp. 9–10)

To determine whether the mean of one related population is significantly different from the mean of another, one needs to ensure that the confidence intervals of our sample means either do not overlap, or do not overlap to a large degree. This a method of estimating the degree to which two population means are different. One can compare these two means to determine whether one is significantly different from the other, which is referred to as statistical hypothesis testing.

Comparison of mean segmenter performance When comparing automatic segmenters, one can look to the task of model selection for intuition as to how to select one segmenter out of many.

> "Many model selection algorithms have been proposed in the literature (Zucchini, 2000). The existing procedures can roughly be categorized as analytical or resampling based. Analytical approaches require certain assumptions of the underlying statistical model. Resampling based methods involve much more computation, but they remove the risk of making faulty statements due to unsatisfied assumptions (Feelders and Verkooijen, 1996). With the computer

> power currently available, this does not seem to be an obstacle.
>
> Standard methods of model selection include classical hypothesis testing (Zucchini, 2000), maximum likelihood (Bishop, 1995), Bayes method (Schwarz, 1978), cross-validation (Stone, 1974), Akaike's information criterion (Akaike, 1974) and many more. Probably the most widely accepted procedure is the use of an information criterion based on choosing the model with the maximized log-likelihood function minus a penalty. However, there is little agreement about what the form of the penalty function should be. Although, there is active debate within the research community regarding the best method for comparison, statistical model selection is a reasonable approach (Mitchell, 1997). We consider the general problem of determining which of a set of competing models is better.
>
> A statistical approach to model selection should try to find out which model is better on average."
>
> —Pizarro et al. (2002, pp. 155-156)

Disregarding other factors such as the difficulty of training a segmenter, the expense of collecting adequate training data, and the speed at which training/segmentation is performed, one can determine the comparative performance of two automatic segmenters by estimating their mean performance upon a data set and performing statistical hypothesis testing to determine whether there is a significant difference between the two segmenters in spite of the error of the mean estimates. This procedure determines whether "the results of an experiment are significant or accidental" (Duda et al., 2000, p. 628).

Statistical hypothesis testing In statistical hypothesis testing, a null (H_0) and alternative hypothesis (H_1) are chosen. H_0 states that no difference between two distributions exists, whereas H_1 states that there exists a difference between two distributions (Cohen, 1995, pp. 106–109). An appropriate test statistic and comparison procedure are chosen for the number of distributions (i.e., automatic segmenters) that does not violate any required assumptions of sample independence or normality common to many statistical tests[14]. A method of choosing an appropriate test statistic and multiple comparison

[14] An investigation into a variety of test statistics and procedures that are applicable to common machine learning scenarios can be found for a single data set in Vázquez et al. 2001; Pizarro et al. 2002 and for multiple data sets in Demšar 2006; García and Herrera 2008.

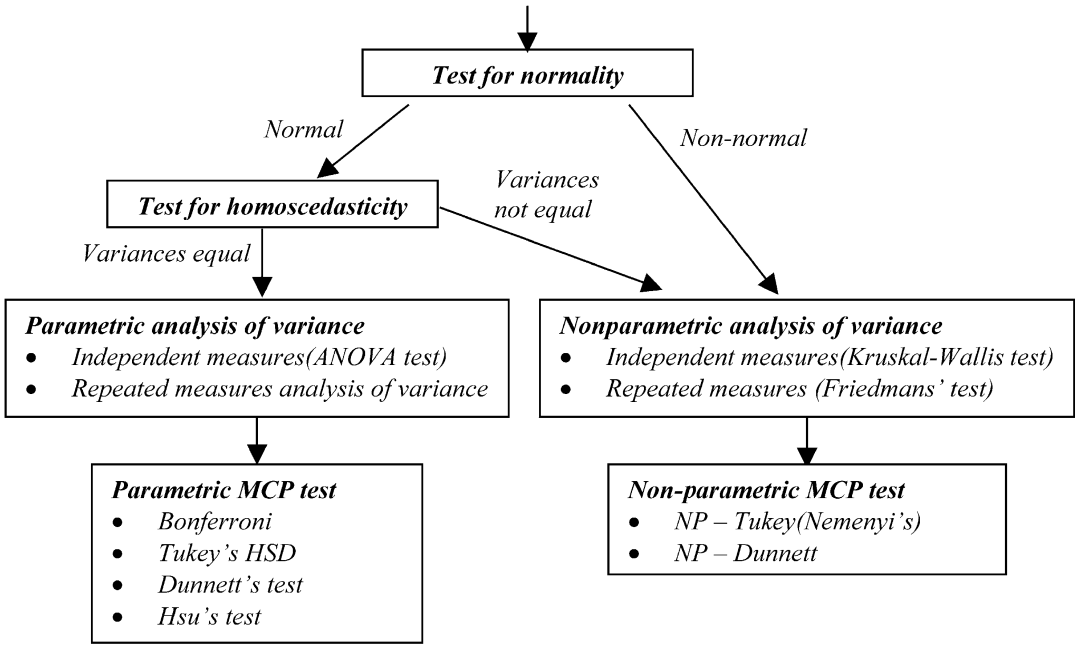


Figure 2.5: A procedure for statistical hypothesis testing for model selection wherein tests are performed upon the per-subject performance of each model to decide upon appropriate procedures (Vázquez et al., 2001, p. 163)

procedure (MCP, or post-hoc test) is outlined by Vázquez et al. (2001) succinctly in their flowchart as reproduced in Figure 2.5 for tests appropriate for more than two segmenters under test.

Applying the procedure outlined in Figure 2.5 requires the application of tests for normality and homoscedacity to verify the assumptions of an analysis of variance (ANOVA). A quantile-quantile (Q-Q) plot or a histogram could be used to observe the distribution of data and visually test for significant deviations from normality, but for reproducibility and brevity such visualizations are not used in this work. Instead, the Shapiro-Wilk test (Shapiro and Wilk, 1965) for normality is used to test the null hypothesis that a distribution does not statistically significantly deviate from normality. If this normality is satisfied, then Bartlett's test (Snedecor and Cochran, 1989) is applied to test the null hypothesis that a set of distributions do not statistically significantly deviate from homoscedacity.

Chapter 3

Literature Review

This thesis aims to improve upon existing methods of evaluating text segmentation with a focus upon issues related to collecting manual segmentations and using segmentation comparison methods to evaluate automatic segmenters. This focus leads to the review of: *a*) methods of comparing arbitrary segmentations; *b*) manual segmentation collection methods and inter-coder agreement coefficients to determine the reliability of collected segmentations; and *c*) methods of using multiply coded corpora to evaluate automatic segmenters. In this chapter, a review of the relevant literature of all of these concepts is presented with an emphasis on how they are applied to text segmentation, with examples specifically of topic segmentation.

3.1 Linear Segmentation Comparison Methods

Beeferman and Berger (1999, pp. 198–200) adapted their earlier measure P_μ, removing the probabilistic distance measure and incorporated the fixed distance of k units as proposed by Doddington (1998). The most intuitive explanation of P_k is given by Pevzner and Hearst (2002, pp. 3–4), which has been adapted and re-formulated here, wherein a window of size k, where k is defined as half of the mean reference segment size, is slid across a text containing $n-1$ potential boundaries spanning n units (Pevzner and Hearst, 2002, p. 5). For each window, and for both the reference (R) and hypothesis (H) segmentation individually, an algorithm determines whether the edges (i.e., probes), are in different segments or not; if R and H disagree, then a penalty of 1 window is assigned. This is accomplished using the same $\delta_R(i,j)$ and $\delta_H(i,j)$ as before, but the positions over

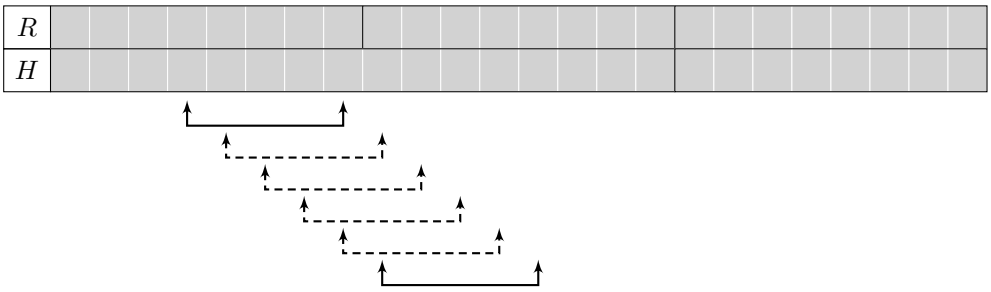


Figure 3.1: How P_k handles false negatives in reference (R) and hypothesis (H) segmentations with windows of size $k = 4$ where dashed lines with arrows represent penalized windows (4 windows of error; adapted from Pevzner and Hearst (2002, p. 5))

which they are evaluated are dependent upon k (Equation 3.1).

$$P_k(R, H) = \sum_{i=1, j=i+k}^{n-k} D_k(i, j) \cdot \left(\delta_R(i, j) \bar{\oplus} \delta_H(i, j)\right) \tag{3.1}$$

These penalties are then normalized by the number of windows measured, which is defined in the distance function, $D_k(i, j)$ (Equation 3.2).

$$D_k(i, j) = \frac{1}{n - k} \tag{3.2}$$

To illustrate how P_k works, an example of how it determines whether a window contains errors is shown in Figure 3.1. In this example, a false negative is shown where the automatic segmenter has left a boundary out of the hypothesis segmentation (H) that was expected in the reference segmentation (R). The lines with two arrows below the two segmentations indicate windows of size $k = 4$ which are being evaluated along the length of the segmentations, where solid lines represent windows contributing no error, and dashed lines representing windows contributing 1 window of error. The total penalty assigned is then k windows for false negatives.

P_k gracefully handles near-misses such as the off-by-one error shown in the first part of Figure 3.2. Here, the first error is penalized by one window of error as the starting probe passes, and later by another window of error (2 penalties total) as the ending probe passes the near miss (not shown). The second error, where the boundaries are off by two units, is penalized by two units of error as the starting passes, and again by another two (4 penalties total for this error) as the ending probe passes (not shown), giving us 6 total windows of error, which is divided by $n - k$ to give us $P_k(R, H) = \frac{6}{24-4} = 0.3$.

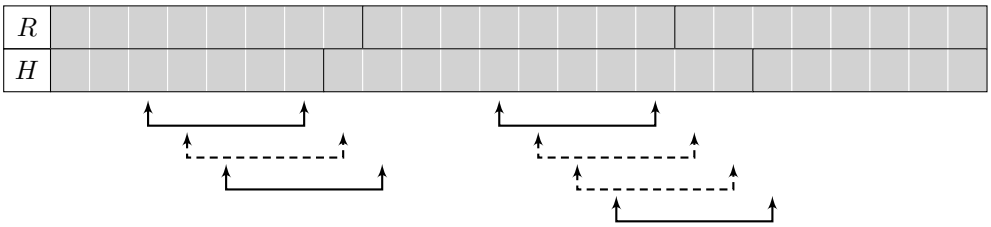


Figure 3.2: How P_k handles boundary near-misses; showing 3 windows of error upon the passage of the first probe with 6 windows in error total (notation identical to Figure 3.1)

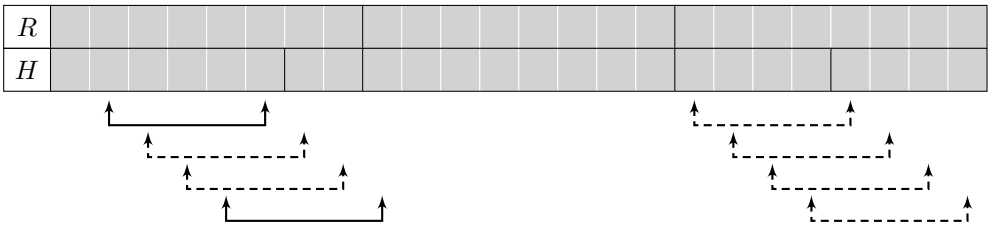


Figure 3.3: How P_k handles misses occurring near another boundary; the first false positive is under penalized at $\frac{k}{2}$, whereas the second false positive is penalized properly as k (notation identical to Figure 3.1, adapted from Pevzner and Hearst (2002, p. 6))

Issues with P_k Although P_k is able to award partial credit for near-misses, and is able to adequately penalize undesirable algorithms—those which place boundaries "everywhere, uniformly, randomly, and not at all" (Beeferman and Berger, 1999)–it is not without some faults. Pevzner and Hearst (2002) identified that P_k can penalize false negatives more than false positives. Assuming that we have a fixed reference segmentation, if a false positive is placed near a matching set of boundaries—as in the first false positive in Figure 3.3), then it will be under-penalized as $\frac{k}{2}$ windows of error, whereas it should be penalized as k windows of error as a false negative would be. When looking at the set of all hypothetical segmentations as compared to one reference, and "assuming uniformly distributed false positives, on average a false positive is noted $\frac{k}{2}$ times by [P_k]–half the rate for false negatives" Pevzner and Hearst (2002, pp. 5–6) (see Figure 3.1 for an example FN). The converse is true if we fix the hypothesis and compare all potential reference segmentations (assuming that there exists more than one reference segmentation).

Another issue with P_k is that in some scenarios it will not identify some errors, leaving them un-penalized. Specifically, it "does not take into account the number of segment boundaries between the two ends of the probe" (Pevzner and Hearst, 2002, pp. 6–7). This can be seen in Figure 3.4, where a false positive is inserted in a narrow space (containing 3 units) which is smaller than the window size ($k = 4$ units), leading the definition of

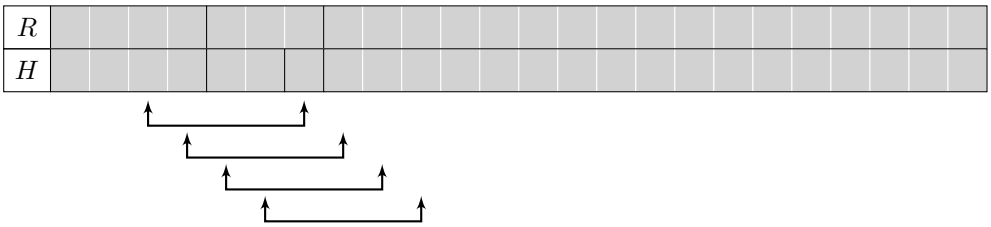


Figure 3.4: How P_k does not identify and fails to penalize misses occurring between probes (notation identical to Figure 3.1, adapted from Pevzner and Hearst (2002, p. 7))

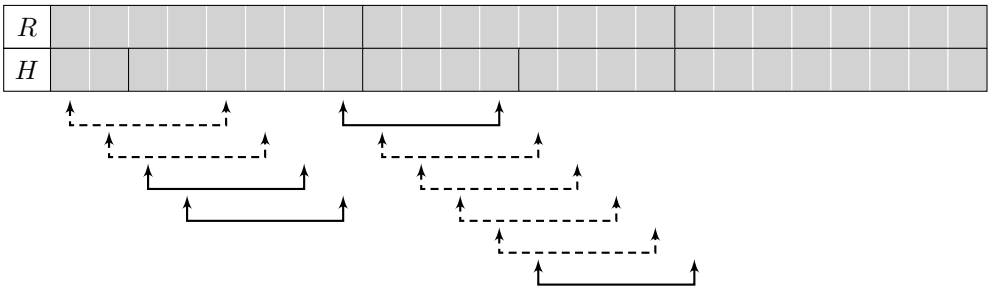


Figure 3.5: How P_k incorrectly penalizes misses occurring near the beginning or ending of segmentations (notation identical to Figure 3.1)

what constitutes disagreement between the reference and hypothesis segmentations to determine that there is in fact, perfect agreement.

The beginning and end of a segmentation provide another issue to methods which slide a window across a segmentation to count errors. If a false positive/negative is j units away from the beginning or end of a segmentation, then it will only be penalized $k - j$ windows of error, where normally they would be penalized as k windows of error (shown in Figure 3.5; Pevzner and Hearst (2002, p. 5)).

Because P_k will fail to penalize boundaries in situations where the segment sizes are smaller than k[1] (e.g., Figure 3.4), this makes P_k sensitive to variations in internal segment sizes; it does not produce consistent values between pairs of segmentations that contain equivalent errors (i.e., numbers of false negatives/positives). As seen earlier, each full miss (i.e., not a near miss) should be penalized with k windows of error (e.g., Figure 3.1), but for P_k this is only true for false negatives if the size of the two segments surrounding the boundary (referred to as A and B) are greater than twice the window size ($|A| + |B| > 2k$), otherwise the penalties assigned "will decrease linearly with $[|A| + |B|]$

[1] k is often determined per segmentation comparison, and if there is a high degree of internal segment size variance, then the probability of windows being smaller than k increases, making such penalization failures probable.

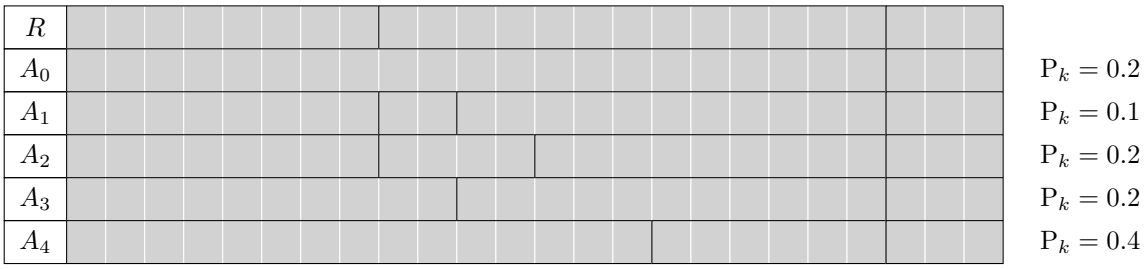


Figure 3.6: How P_k differentiates between a variety of automatically generated hypothetical segmentations (A_0—A_4; adapted from Pevzner and Hearst (2002, p. 9))

so long as $[k < |A| + |B| < 2k]$" (Pevzner and Hearst, 2002, p. 7).

Although P_k is able to award partial credit to many near misses, it does not do so for all, and it is often very harsh in its assessments. An example of this is illustrated by the P_k values for the automatically generated hypothetical segmentations (A_0—A_4) in Figure 3.6. In the contrived example, A_0 represents a false negative, A_2 a false positive, and A_3 a near miss, and all three of these errors are penalized to the same degree when measuring errors using P_k, but should they be? Pevzner and Hearst (2002, pp. 8–9) argue that they should not, and that of the three segmenters, A_3 should be preferred over A_0 and A_2 because that automatic segmenter recognizes that there is only a single boundary[2], and it has only nearly missed the reference boundary position in its placement of this sole boundary. A_1 presents an even worse failing of P_k to identify a correct segmentation, with the automatic segmenter placing a correct boundary but then a near boundary that is a full miss; it could arguably be considered a less egregious error than the additional boundary provided by A_2, it is also arguable that the score of A_3 and A_1 should be closer than A_2 and A_1. In this, and many analogous cases, P_k penalizes near misses too harshly in comparison to what should be greater or more equal errors.

P_k was a step forward; it allowed for degenerate algorithms to receive an error score of approximately 0.5, whereas good algorithms would be discernible by their low error score, and near misses were given partial credit. However, the issues surrounding its unequal penalization of FPs versus FNs, its tendency to overlook FPs and FNs within segments of lengths less than k, its sensitivity to variations in segment size, and its over-penalization of near-miss errors make it less than ideal as a method for comparing segmentations to quantify error.

[2]This interpretation of whether boundary cardinality should be preferred over boundary position placement is not discussed in depth by Pevzner and Hearst (2002), but the balance between these two aspects of segmentation error should be determined on an application by application basis when choosing an error counting method to evaluate segmentation.

Adapting P_k: WindowDiff To attempt to mitigate the shortcomings of P_k, Pevzner and Hearst (2002, p. 10) proposed a significant modification of P_k–named *WindowDiff* (*WD*)—such that "for each position of the probe, simply compare how many reference segmentation boundaries fall in this interval (r_i) versus how many boundaries are assigned by the algorithm (a_i)", and then penalize the algorithm if $r_i \neq a_i$ (computed as $|r_i - a_i| > 0$ where the result of this boolean expression is interpreted as an integer). In this modification, a window of size k is still slid across the text, but now penalties are attributed to windows where the number of boundaries in each segmentation differs (see Equation 3.3, where b(w) is a function that counts the number of boundaries in a window, b(R_{ij}) and b(H_{ij}) represents the number of boundaries within the segments in a window of size k from position i to j, and n the number of sentences), with the same normalization as P_k.

$$\mathrm{WD}(R, H) = \frac{1}{n-k} \sum_{i=1, j=i+k}^{n-k} (|\mathrm{b}(R_{ij}) - \mathrm{b}(H_{ij})| > 0) \tag{3.3}$$

WindowDiff had advantage over P_k because it essentially eliminated the unequal penalization of FPs versus FNs, and it can identify FPs and FNs in segments less than length k (which decreases its sensitivity to variation in internal segment size).

To validate that WindowDiff was an improvement over P_k, Pevzner and Hearst (2002, pp. 11–18) calculated the errors detected by each method (and some weighted variants) over large numbers of simulated segmentations while controlling for the rate at which certain errors were introduced and the distribution of internal segment sizes. For their simulations, each trial consisted of generating one reference segmentation containing 1000 segments that were randomly generated using a specified internal segment size distribution, and then similarly generating 100 hypothetical segmentations having a particular property (e.g., a specific FP error rate), and calculating the difference between these 100 hypothetical segmentations and the reference segmentation using WD and P_k, and averaging these results.

To verify that WindowDiff varies less due to internal segment sizes than P_k, Pevzner and Hearst (2002, pp. 11–18) performed 10 trials where hypothetical segmentations were generated that kept the mean internal segment size at 25 units in length for four different ranges: (20,30), (15,35), (10,40), and (5,45)—where the first number in each tuple represents the lower bound, and the second the upper bound (inclusively). The error rates of the hypothetical segmentations were also controlled for, and errors were

	(20,30)	(15,35)	(10,40)	(5,45)
P_k	0.245	0.245	0.240	0.223
WD	0.245	0.245	0.242	0.237

(a) False Negatives ($p = 0.5$)

	(20,30)	(15,35)	(10,40)	(5,45)
P_k	0.128	0.122	0.112	0.107
WD	0.240	0.241	0.238	0.236

(b) False Positives ($p = 0.5$)

	(20,30)	(15,35)	(10,40)	(5,45)
P_k	0.317	0.309	0.290	0.268
WD	0.376	0.370	0.357	0.343

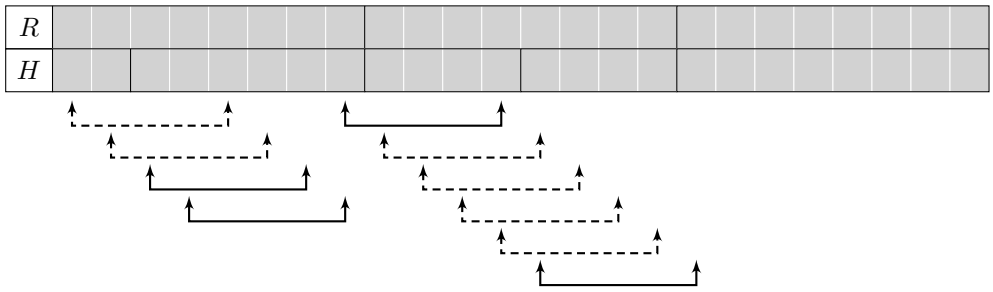
(c) False Negatives & False Positives ($p = 0.5$)

Table 3.1: Mean error as calculated by P_k and WindowDiff for four different error probabilities over four different internal segment size distributions whose mean internal segment size is kept at 25 units, demonstrating decreased variability in the values of WindowDiff within each scenario (i.e., table) for differing internal segment sizes (i.e., columns) in comparison to P_k; adapted from from Pevzner and Hearst (2002, p. 12)

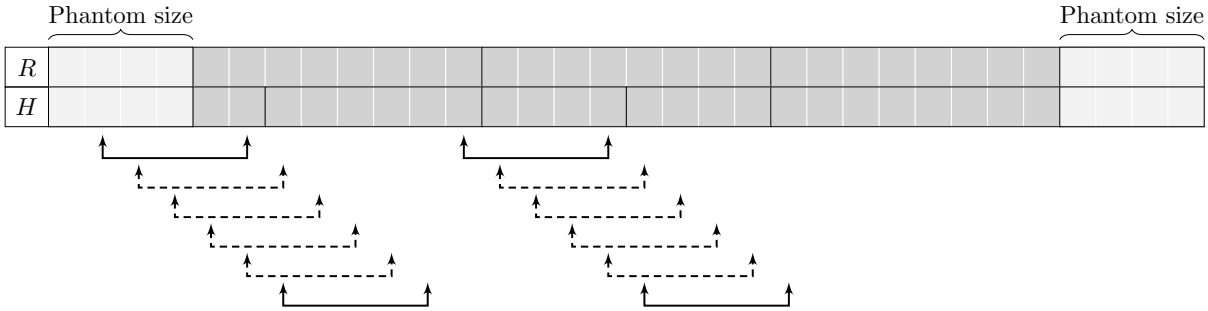
uniformly placed with three different types and probabilities: FNs with a probability of 0.5 ($p = 0.5$), FPs ($p = 0.5$), and both FPs and FNs ($p = 0.5$) as shown in Table 3.1.

In Table 3.1a, for the simulations involving FNs, both P_k and WindowDiff appear to be affected by variations in the internal segment sizes (columns) by similar amounts, but WD slightly less so (P_k ranges from 0.223 to 0.245, whereas WD varies from 0.237 to 0.245). Table 3.1b shows that when counting FPs that WD properly identifies FNs (whereas P_k's values dip below what should be expected if both FPs and FNs were to be equally penalized, which is illustrated by comparing these values to those in Table 3.1a, where P_k values should be similar in value across both tables but are not). WD also shows less variability due to internal segment sizes, ranging from 0.236 to 0.241, whereas P_k varies from 0.107 to 0.128. Finally, when combining these two types errors—with equal probability of occurrence, we see still see sensitivity to internal segment size present in both, but less so for WD in Table 3.1c where WD varies from 0.343 to 0.376 and P_k varies from 0.268 to 0.317. Overall, WD appears to vary less due to internal segment size ranges than P_k in these simulations.[3]

[3]One obvious criticism of this analysis is that the means reported are not accompanied by their variances, leaving us with less of an ability to determine whether these variations are truly significant. Because these means are approximate, statistical significance could be evaluated between the populations described by these means to determine whether WD significantly reduces sensitivity to internal segment sizes compared to P_k.



(a) How WD incorrectly penalizes misses (FPs or FNs) near the beginning of a segmentation



(b) How WD can correctly penalize misses occurring near the beginning of a segmentation by padding the ends by k units as suggested by Lamprier et al. (2007)

Figure 3.7: Effect of adding k units of phantom size at the beginning and end of a segmentation upon how WindowDiff counts errors (notation identical to Figure 3.1)

Issues with WindowDiff, and all window-based comparison methods WindowDiff was an improvement over P_k, and has become the standard method used for comparing linear segmentations. WindowDiff is not without its own set of issues, however. The method still varies depending upon variations in internal segment sizes and upon closer inspection has been found to have some less desirable properties, not all of which can be remedied.

Lamprier et al. (2007) demonstrated that WindowDiff penalizes errors less at the beginning and end of a segmentation due to how a window is slid across text (e.g., Figure 3.7a). This can be corrected by padding the segmentation at each end by k units, ensuring that the same number of windows are measured over errors at the beginning and end as are for errors $\geq k$ units from the edges of a segmentation, as shown in Figure 3.7b. Alternatively, Lamprier et al. (2007) also propose a reformulated version of WD which claims to solve this issue.

Georgescul et al. (2006, p. 148) identified that when calculating WindowDiff, both FPs and FNs are weighted by $^1/_{N-k}$. By this weight definition, both FPs and FNs are

penalized equally for each time they are encountered encountered, but although there are "equiprobable possibilities to have a [FP] in an interval of k units", "the total number of equiprobable possibilities to have a [FN] in an interval of k units is smaller than $(N - k)$", making the interpretation of a full miss as a FN less probable than as a FP. This contradicts the assertion by Pevzner and Hearst (2002) that WindowDiff handles FPs and FNs equally when considering the effect of how WindowDiff models errors, which slightly increases the probability of whether a miss is considered a FP over a FN.

Though not explicitly noted in the original describing papers for P_k or WindowDiff, k must be a positive integer because it is used to operate upon the set of atomic text units which themselves are separate and indivisible units whose cardinality is always expressed as whole numbers. For this reason, once half of the mean reference segment size is calculated, this value must be rounded to be utilized by the WindowDiff or P_k algorithms. The introduction of this rounding, and the variety of inconsistent rounding strategies anecdotally used by researchers has led to difficulty in interpreting methods that use sliding windows to count errors. Strategies that have been employed by researchers have included calculating k using the mean internal segment size for all reference segmentations or using only the current reference which they are performing comparisons against and adopting slightly different rounding strategies (e.g., simply rounding up when encountering the decimal number 5 versus statistically motivated rounding strategies that avoid bias towards rounding up by only doing so when a 5 is preceded by an odd number).

Niekrasz and Moore (2010), when analysing P_k and WindowDiff, found that both comparison methods have a bias towards hypothetical segmentations that exhibit:

- Fewer segments (i.e., precision is favoured over recall);
- Clumping (i.e., spurious boundaries are placed adjacent to correct boundaries); and
- Edge bias (i.e., the bias identified by Lamprier et al. (2007)).

Noting these biases, they proposed scaling P_k to account for chance agreement in an attempt to provide an unbiased comparison method in the form of what is essentially an inter-coder agreement coefficient. Unfortunately, the proposed scaling did not produce a comparison method, it produced an unintuitive inter-coder agreement coefficient loosely based upon Cohen's κ.

WindowDiff is regarded as not a very intuitive comparison method, and some work has been put into proposing reformulations which are easier to interpret and attempt to mitigate the issues with WindowDiff (e.g., Georgescul et al.'s (2006) Pr_{miss}, Lamprier et al.'s (2007) *reformulated WD* and *normalized WD*, and Scaiano and Inkpen's (2012)

WinPR which adapts information retrieval metrics to use WindowDiff-based confusion matrices as an error comparison method). Instead of modifying WindowDiff, others have proposed entirely unrelated methods and metrics to replace WindowDiff (e.g., Franz et al. 2007; Fournier and Inkpen 2012; Brooke et al. 2012, etc.[4]) and avoid its pitfalls.

In extending WindowDiff, Lamprier et al. (2007, pp. 20–21) remarked that when creating a reference segmentation coders can have divergent interpretations of what constitutes a topic in topical segmentation, citing Hearst (1997) as an example. Combining multiple sections of documents together to artificially create topical segmentation data as in Choi (2000) can lead to a lack of homogeneity in the text being segmented, and the effects of this lack of homogeneity have not been quantified with respect to manually created reference segmentations. Kazantseva and Szpakowicz (2012) collected manual segmentations from 4–6 coders over 20 chapters of a novel. Using mean P_k and WindowDiff, they remarked that their manual coders had divergent opinions. To mitigate this issue, Kazantseva and Szpakowicz (2012) proposed a modification of WindowDiff which attempted to penalize each boundary to the degree that each manual coder agreed that position constituted a boundary, calling the method *multiWindowDiff*. All of these modifications to WindowDiff account for the difficulty in utilizing divergent manual segmentations, but unfortunately suffer from all of the previously mentioned issues with WindowDiff, and the issue of combining multiple codings into one reference, or creating a artificial segmentation data, are unsuitable. Given the issues with window-based segmentation comparison methods, and the difficulty of using multiple divergent manual segmentations, which alternatives exist?

Alternatives to window-based comparison methods Franz et al. (2007), when evaluating text segmentation methods for usage in a system that segment audio and video transcriptions by topic, proposed measuring performance in terms of the number of words that a user will either miss, or be erroneously presented, when being shown a search result. Essentially, this performance metric measures the correct set of words representing a news story segment truncated by w words, or concatenated with additional w words, representing FNs and FPs respectively. Franz et al. (2007) proposed measuring a count of the missing or additional/false-alarm words for a segment normalized by the number of word positions present (see Equations 3.4–3.5).

[4]The comparison method proposed in Section 4.1.3 is published in Fournier and Inkpen (2012).

$$R_{FN} = \frac{1}{N} \sum_{w} FN(w) \qquad (3.4) \qquad\qquad R_{FP} = \frac{1}{N} \sum_{w} FP(w) \qquad (3.5)$$

R_{FN} and R_{FP} have the advantage of being intuitive and clearly linked to the end application of the segmentation method because they take into account the severity of an error in terms of segment size, allowing R_{FN} and R_{FP} to reflect the effects of erroneously missing or added words in a segment better than window based comparison methods. Unfortunately, R_{FN} and R_{FP} only really account for near misses, and are not applicable to segmentations containing more than two boundaries (a start and end).

While evaluating the performance of a large number of automatic segmenters upon manual segmentations of the speakers in T.S. Elliot's "The Wasteland", Brooke et al. (2012) found that both P_k and WindowDiff would favour automatic segmenters that produced very few breaks. To mitigate this, in addition to reporting P_k and WindowDiff, Brooke et al. (2012, p. 31) proposed another comparison method named *Break Distance* (BD) which "sums all the distances, calculated as fractions of the entire text, between each true break and the nearest predicted break." Although this penalty-based comparison method can easily underestimate error if an automatic segmenter places a large number of segments, when accompanied by P_k or WindowDiff, it contrasts the picture given by P_k and WD when they underestimate error presented by placing few boundaries and provided a clearer picture of the performance obtained (akin to the relationship between precision and recall when graphed against each other).

Current state of segmentation evaluation WindowDiff was an improvement over P_k, is able to less severely penalize near-misses, and is also less sensitive to variation in internal segment sizes. Unfortunately, WindowDiff (and all window-based comparison methods) has a number of drawbacks, including:

- Sensitivity to variations in internal segment sizes (Pevzner and Hearst, 2002);
- Errors at the beginning and end of segmentations are not fully penalized (although this can be mitigated as shown by Lamprier et al. (2007));
- Favouring hypotheses with fewer segments (i.e., precision is favoured over recall);
- Favouring hypotheses that contain clumping; and
- Window size calculation rounding is not explicitly specified and consistent;

- Window size is based upon only the reference segmentation, meaning that if the two segmentations compared swapped roles (a reference segmentation is considered a hypothesis, and vice versa), different WD values would result, making WD non-symmetric, violating the axiomatic requirements to say that WD operates in a metric space, and making it unsuitable for use as part of an inter-coder agreement coefficient; and

- WindowDiff, and quantifying error in terms of windows, is not intuitively related to the units being segmented.

There are modifications to WindowDiff to attempt to solve issues related to incorrect error counting at the edges of a segmentation and to improve the interpretability of the methods, but none of the other issues are fully addressed. There are a few replacements for WindowDiff that are highly suited for specialized applications (R_{FN} and R_{FP}) or as supplements to WindowDiff (BD), but neither set of solutions are necessarily suitable for topic segmentation or other subjective or contextual text segmentation tasks involving multiple divergent manual segmentations of the same text.

In Section 4.1.3, a number of comparison methods are proposed to attempt to replace WindowDiff which later chapters experimentally compare to determine whether they indeed can solve the current issues with segmentation comparison methods and improve upon the state-of-the-art (WindowDiff). A comparison method is of little use, however, without reference segmentations to compare against which are often collected from manual segmenters. For this reason, not only are methods of comparing two segmentations required, but also methods of determining the quality of reference segmentations such as measures of inter-code agreement.

3.2 Inter-Coder Agreement

Chance agreement in segmentation Rosé (1995) and Isard and Carletta (1995) both illustrated the need to report an agreement statistic that took chance-agreement into account to determine both the reliability of the codings collected and the reproducibility of segmentation studies. Percent agreement against a majority coding conflates coder solutions to ideally create a "true" reference segmentation against which to compare, but no studies present a clear rationale for the majority scheme used, and even the definition

of a majority is arbitrary.[5] Even if one was to adapt percentage agreement to take into account chance agreement, it would be difficult to do so without grossly inflating its results, because "the statistic itself guarantees at least 50% agreement by only pairing off coders against the majority opinion"(Isard and Carletta, 1995, p. 63).

Chance agreement can occur in segmentation when coders operating at slightly different granularities agree randomly due to the frequency or distribution of their codings, and not their own innate interpretation of what constitutes a segmentation boundary. In segmentation one can assume that each potential boundary position is a trial where a coder can choose to place a boundary, or not. If two coders were placing an equal proportion of boundaries, but chose to place boundaries randomly, one would expect that the coders would agree with each other 50% of the time given their 2 choices (or categories).

> "If instead, the two coders were to use four categories in equal proportions, we would expect them to agree 25% of the time (since no matter what the first coder chooses, there is a 25% chance that the second coder will agree). And if both coders were to use one of two categories, but use one of the categories 95% of the time, we would expect them to agree 90.5% of the time ($.952^2 + .05^2$)."
>
> – Isard and Carletta (1995, p. 63)

These chance agreement probabilities make it impossible to interpret agreement statistics which do not account for such chance agreement (Isard and Carletta, 1995, p. 63).

Inter-coder agreement coefficients To account for chance agreement, a variety of statistics have been developed that make statistical assumptions about what constitutes chance agreement, and modifies an actual agreement statistic to account for how much of that agreement is to be expected by chance, which are called inter-coder agreement coefficients. These coefficients each assume that different distributions of choices (proclivity towards coding using one or more categories more than others, or producing more codings than other coders) by coders characterize chance agreement. Rosé (1995); Isard and Carletta (1995) both suggest the usage of κ-like coefficients, with the general form shown in Equation 3.6.

$$\kappa = \frac{A_a - A_e}{1 - A_e} \tag{3.6}$$

[5] This is exemplified by Litman and Passonneau (1995, pp. 109–110) changing their arbitrary definition of a majority from what they defined in Passonneau and Litman (1993, p. 150).

These coefficients share the same form and are parameterized differently: actual agreement between coders is represented as A_a and the agreement that would be expected by chance as A_e, where A_a and A_e are defined by each specific coefficient. The coefficients range in value from $[{}^{-A_e}/_{1-A_e}$ to 1 (inclusively). A value of zero ($A_a = A_e$) indicates that coders are operating by chance alone, whereas a value above zero ($A_a > A_e$) indicates that there is some agreement occurring that is not simply by chance. A value below zero ($A_a < A_e$) indicates that coders are *dis*agreeing more than would be expected by chance.

There are a variety of inter-coder agreement coefficients, and throughout their usage a variety of names have been applied to each. Artstein and Poesio (2008) provides a broad evaluation, and standardization of the various names of the available inter-coder agreement coefficients as applied to computational linguistics. Initially, the computational linguistics and natural language processing communities were introduced to inter-coder agreement and reliability by Carletta (1996), whose work served as a general introduction to inter-coder agreement coefficients and argued against the usage of non-chance corrected percentage agreement. This introduction caused some confusion, however.

> "Carletta uses the term kappa for the coefficient of agreement, referring to Krippendorff (1980) and Siegel and Castellan (1988) for an introduction, and using Siegel and Castellan's terminology and definitions. However, Siegel and Castellan's statistic, which they call K, is actually Fleiss's generalization to more than two coders of Scott's π, not of the original Cohen's κ"[6].
>
> – Artstein and Poesio (2008, p. 1)

Three well known coefficients for pairs of coders include Bennett et al.'s (1954) S, Scott's (1955) π, and Cohen's (1960) κ. These coefficients can be generalized as shown in Equation 3.7, where they use the same calculation of actual agreement (A_a) and only differ in how they calculate expected agreement (A_e).

$$S,\ \pi,\ \kappa = \frac{A_a - A_e}{1 - A_e} \tag{3.7}$$

Artstein and Poesio (2008) outline differences between these coefficients in terms of how

[6]To make matters worse, in Fleiss' generalization of Scott's π, he introduced it as κ (Fleiss, 1971). For clarity, herein, this work will adopt the terminology used by Artstein and Poesio (2008), where Siegel and Castellan's (1988) coefficient will be referred to as K and interchangeably as Fleiss' multi-π (shown later), related coefficients which model chance agreement as being identical distributions as π, and any coefficient that models chance agreement using individual coder distributions as κ.

Name(s) used herein	*Publication*	*Previously used names*
π	Scott (1955)	
Multi-π, π^*, or K	Fleiss (1971)	Fleiss' κ, Siegel and Castellan's (1988) K, K or 'kappa' by Carletta (1996)
κ	Cohen (1960)	'kappa'
Multi-κ or κ^*	Davies and Fleiss (1982)	

Table 3.2: Coefficient used herein and by Artstein and Poesio (2008) and their synonyms

they characterize the chance that a particular coder will assign an arbitrary item to a category (i.e., how A_e is calculated):

> S: "This coefficient is based on the assumption that if coders were operating by chance alone, we would get a uniform distribution: that is, for any two coders c_m, c_n and any two categories k_j, k_l, $\mathrm{P}(k_j|c_m) = P(k_l|c_n)$. (Put it another way: chance does not distinguish between categories and coders.)"
>
> π: "If coders were operating by chance alone, we would get the same distribution for each coder: for any two coders c_m, c_n and any category k, $\mathrm{P}(k|c_m) = P(k|c_n)$. (i.e., chance distinguishes between categories, but not coders.)"
>
> κ: "If coders were operating by chance alone, we would get a separate distribution for each coder: chance distinguishes between both categories and coders."
>
> – Artstein and Poesio (2008, pp. 7–8)

For comparing more than two coders against each other, both π and κ described previously have been generalized to suit multiple coders as Fleiss' multi-π (Fleiss, 1971) and multi-κ (Davies and Fleiss, 1982), and can be generalized in a similar manner as shown in Equation 3.8 (Artstein and Poesio, 2008, p. 12).

$$\text{Multi-}\pi,\ \text{multi-}\kappa = \frac{A_a - A_e}{1 - A_e} \tag{3.8}$$

The differences between these coefficients are in how A_a and A_e are calculated to properly compare multiple coders, but the assumptions remain the same in terms of how they characterize chance agreement (Artstein and Poesio, 2008, pp. 13–14). These coefficients have been introduced multiple times in the past, and to attempt to standardize their names in relation to how they characterize chance agreement, Artstein and Poesio (2008) indicate some of the names which they have been referred to as (shown in Table 3.2).

One limitation of all versions of κ and π are that they treat all disagreements as being equal in severity (Artstein and Poesio, 2008, p. 14). For this reason, some studies

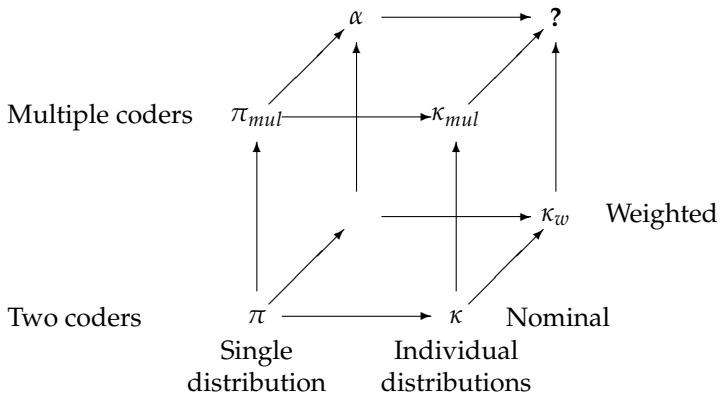


Figure 3.8: Inter-coder agreement coefficients arranged in three dimensions by properties (Artstein and Poesio, 2008, p. 19)

choose to use Krippendorff's α (Krippendorff 1980, Chapter 12, Krippendorff 2004a, Chapter 11), which is based upon similar assumptions as those that π makes about what constitutes chance agreement (Artstein and Poesio, 2008, p. 15), they use a weighted version of κ (Cohen, 1968). This work focuses primarily on the usage of unweighted version of π and κ for simplicity of interpretation, and only omits use of Krippendorff's α for brevity and due to its similarities to π. To summarize the differences between all of the coefficients that Artstein and Poesio (2008) survey, they place them upon the corners of a three dimensional cube (where the dimensions indicate whether a coefficient is: nominal/weighted, applies to only two coders/multiple coders, or uses single/individual distributions to model chance agreement) to visually depict their differences and to aid in coefficient selection, as shown in Figure 3.8. The top right and bottom left corners remain unlabelled because a weighted, individual distribution, multi-coder and a weighted, single distribution, two-coder coefficient was not found by Artstein and Poesio's (2008) survey.

Coefficient choice for segmentation Craggs and McGee Wood (2005, p. 290) point out that an appropriate statistic for determining reliability "must measure agreement as a function of the coding process and not of the coders, data, or categories." The statistic chosen should be able to demonstrate whether there exists a "shared understanding" between coders of how to describe the process of segmentation given the coding scheme used. Percentage agreement does not correct for agreement that could occur by chance, and does not then represent agreement necessarily as a function of the coding process. Instead, there are a variety of inter-coder agreement coefficients that do account for chance agreement from which to choose. Besides the choice of coefficients that can be

used with only two coders or more than two coders, the remaining choice is how chance agreement between coders should be quantified. Many coefficients make the assumption that if coders were operating solely by chance, that one would see in their codings a:

- Uniform distribution across coders and categories (S);
- Single distribution for each coder across categories (π, multi-π, and K); or
- Individual (i.e., different) distributions for each coder across categories (κ and multi-κ);

These coefficients measure the degree to which these distributions occur, and use this to adjust actual agreement to better be a function of the coding process itself by taking into account the idiosyncrasies produced by the coding scheme chosen. The question then is which assumption is correct for a specific linear text segmentation task.

K (i.e., multi-π) has been a popular coefficient in computational linguistics, and also in segmentation. It has been used in various publications to measure the reliability and/or replicability of:

- An action and discourse transaction segmentation study (Isard and Carletta, 1995);
- Coders and automatic segmenters[7] upon linear topical segmentations of magazine articles (Hearst, 1997);
- A dialogue coding scheme that modelled utterance function, game structure, and high-level transactions of a spontaneous task-oriented corpus of speech (Carletta et al., 1997);
- Topical segment annotations and TREC broadcast news segmentations (Reynar, 1998);
- Structural annotations of scientific articles (Teufel et al., 1999);
- Topical segmentations of a meeting dialogue corpus (Ries, 2001);
- Elementary discourse unit segments for Rhetorical Structure Theory (RST) annotations of 385 documents from Marcus et al.'s (1993) Penn Treebank (Carlson et al., 2003);
- Relational Discourse Analysis (Moore and Pollack, 1992; Moser and Moore, 1996) annotations of tutorial dialogues (Poesio and Patel, 2006); and
- Linear topical segmentations of a novel (Kazantseva and Szpakowicz, 2012).

[7] Various versions of TextTiling, an automatic topic segmenter (Hearst, 1994, 1997)

None of the works mentioned, however, offer any rationale for specifically using K over another coefficient besides arguing against non-chance-corrected percentage agreement.

Cohen's (1960) κ and Fleiss's multi-κ have also been used in segmentation, though with only small popularity since K has been so prolifically reported previously. Rosé (1995) proposes using Cohen's κ to evaluate the coding scheme used for her dialogue segmentation, conversational acts, and interactional structure, citing the recommendation for using the coefficient when analysing sequential behavior by Bakeman and Gottman (1986, 1997). Bakeman and Gottman (1986, 1997) and Rosé (1995) argue against non-chance-corrected percentage agreement, but do not offer a explanation of why one should specifically use Cohen's κ as opposed to another coefficient. Di Eugenio and Glass (2004) assert that Cohen's κ is an ideal coefficient for identifying inter-coder agreement for "the vast majority, if not all, of discourse—and dialogue tagging—efforts" because of the assumption that chance agreement is exemplified by equal coder distributions made by π, multi-π, and K does not hold (Di Eugenio and Glass, 2004, p. 96). Indeed, it is reasonable to expect that random chance could occur between two coders if one coder labelled a category more frequently than the other. For this reason, (Di Eugenio and Glass, 2004, p. 96) recommend reporting Cohen's κ or Fleiss's multi-κ, but because of the effects of prevalence on both κ and π, and the effects of bias upon κ, it should be accompanied by Scott's π or Fleiss' multi-π.

Prevalence is a characteristic of codings which affects the values of κ and π (and by extension K), potentially leading to difficulty in interpreting such statistics. Similarly, bias, which only affects κ, is another coding characteristic which affects agreement values. This is because the values of π and κ are affected by both A_a and A_e (actual and expected agreement, respectively; see Equations 3.7–3.8), but, troublingly, although changes in A_a produce obvious and expected changes in agreement, some coding distributions produce unexpected values of A_e (Di Eugenio and Glass, 2004, pp. 98–99). These coding distribution scenarios are referred to as bias and prevalence. When A_e increases, this decrease becomes more pronounced as A_a is lowered.

Prevalence occurs when the distribution of coders is concentrated in a very few (or one) categories. A_e's minimum (i.e., ideal) value occurs when codings are equally distributed amongst categories ($A_e = {}^{1}/_{|\kappa|}$), while the maximum value (i.e., worst case) occurs when all codings are concentrated in one category ($A_e = 1$) (Di Eugenio and Glass, 2004, p. 98). Prevalence is especially troubling because it affects both *kappa* and π, and because A_a and A_e are based upon proportions and scale proportionately with the number of codings, this means that increasing the number of codings does not counteract prevalence

		Coder 2		
		Accept	Ack	
Coder 1	Accept	90	5	95
	Ack	5	0	5
		95	5	100

$A_a = 0.90,\ A_e = 0.905$

$\kappa = \pi = -0.048$

$p = 1$

(a) Very prevalent category 'Accept'

		Coder 2		
		Accept	Ack	
Coder 1	Accept	45	5	50
	Ack	5	45	50
		50	50	100

$A_a = 0.90,\ A_e = 0.5$

$\kappa = \pi = 0.80$

$p = 0.5 \times 10^{-5}$

(b) Equally prevalent categories

Figure 3.9: Cohen's contingency tables illustrating the effect of prevalence upon κ and π (Di Eugenio and Glass, 2004, Figure 3, p. 99)

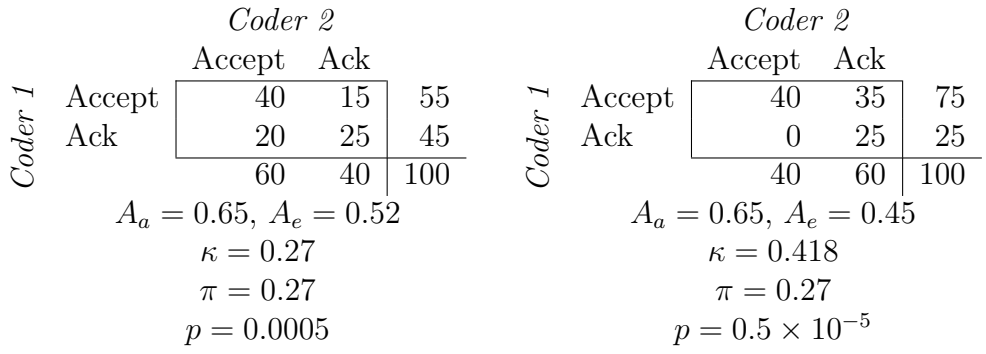
(Di Eugenio and Glass, 2004, p. 99).

In Figure 3.9, two examples from Di Eugenio and Glass (2004, Figure 3, p. 99) are shown—which demonstrate the effect of prevalence—where coders are attempting to distinguish the meaning of the word 'Okay' in dialogue as either an acceptance ('Accept') or an acknowledgement ('Ack') of something. In both examples, the two coders agree upon 90 out of 100 codings, but due to the prevalence of the 'Ack' category in Figure 3.9a, it has a far higher A_e which reduced the overall agreement values far below those shown in Figure 3.9b (-0.048 versus 0.80).[8]

Although Di Eugenio and Glass (2004) consider prevalence to be a negative aspect of inter-coder agreement coefficients such as κ and π, Craggs and McGee Wood (2005) do not. While re-examining Figure 3.9a, Craggs and McGee Wood (2005, p. 291–292) assert that despite the 90 out of 100 agreements, and because not a single coder agrees upon 'Ack' as an interpretation of 'Okay', that the only conclusion that can be drawn is that the coders cannot distinguish between the two categories–'Accept' and 'Ack'. In this way, prevalence's effect of producing very low agreement appropriately demonstrates that one should not rely upon this coding. Prevalence, then, is a desirable effect when the goal is to determine the reliability of a coding scheme.

Bias occurs in cases where A_e is calculated by assuming that individual coder distributions (and whether they differ) quantifies chance agreement, which is the case for Cohen's κ and Fleiss' Multi-κ. If two coders agree less in the behaviour of their codings, then they will have fewer expected chances to agree. This paradoxically increases κ when

[8]These examples also show tests for whether agreement is significantly different from zero (calculated according to Cohen (1960, pp. 43–44) for κ and Siegel and Castellan (1988) for K, or π). Only Figure 3.9b shows a significantly different value from zero.



(a) Very prevalent category 'Accept' (b) Equally prevalent categories

Figure 3.10: Cohen's contingency tables illustrating the effect of bias upon κ and not upon π (Di Eugenio and Glass, 2004, Figure 3, p. 99)

coders behave differently, because A_e decreases.

In Figure 3.10, two examples of bias from Di Eugenio and Glass (2004, Figure 3, p. 99) are shown. In both examples, the two coders each perform 100 codings, and in Figure 3.10a the category proportions are similar between coders (55 versus 60 for 'Accept' and 45 versus 40 for 'Ack'), indicating the they are behaving similarly. Figure 3.10b shows the two coders behaving very differently with coder 1 favouring 'Accept' more than coder 2 (75 versus 40), and inversely shunning 'Ack' with 25 versus 60 codings. π remains identical at 0.27, whereas κ fluctuates between 0.27 and 0.418, grossly overestimating agreement in Figure 3.10b. Craggs and McGee Wood (2005, p. 291–292) point to bias as one reason to avoid the usage of κ in computational linguistics.

Krippendorff α, though a versatile statistic, has seen very little usage in segmentation. Passonneau and Litman (1997) reported the agreement of their segmentation task using Krippendorff's α. Artstein and Poesio (2008, pp. 41–43) propose that Krippendorff's (1995) method of measuring agreement (using Krippendorff α) on unitizing may be appropriate for segmentation. Unitizing is the process of "identifying boundaries between units, and selecting the unit of interest" from overlapping units between coders (where coders each segment a unit but disagree upon exact boundary positions). Disagreement during unitizing is quantified as the "square of the lengths of the non-overlapping segments" (as shown in Figure 3.11). If there is no overlap, then disagreement is the square of the length of the whole segment (Artstein and Poesio, 2008, 42–43). This would help to overcome the issue of low agreement that occurs in segmentation tasks which often occurs when statistics do not account for near-misses in boundary placement (described later).

Overall, the choice of which agreement statistic to use, especially for segmentation,

Figure 3.11: Disagreement between overlapping units during unitizing, where disagreement $\mathrm{d}(A, B) = s_-^2 + s_+^2$ (Artstein and Poesio, 2008, p. 43)

depends upon one's purpose. The goal of calculating an inter-annotator agreement coefficient is often either to:

- "Infer the reliability of a large-scale annotation", and thus the "reliability of the annotation procedure, which is independent of the actual annotators used" (Artstein and Poesio, 2008, p. 24); or
- Infer the 'trustworthiness' (i.e., validity) of the data produced by a set of coders in agreement.

The question of validity is strongly linked to the purpose for which the data is to be used, and could often be better ascertained through other statistics (e.g., data validity is essential to the quality of solution used to train a statistical classifier). There is no doubt that validity is exemplified by a value near perfect agreement of any coefficient, but even in moderate agreement the individual coder bias of π can inform us of whether a particular annotation may not be reliable, but the decisions of individual coders are consistent, and easily trained upon.

There has been a large amount of discussion on which coefficient to use to infer the reliability of the annotation procedure (e.g., coding scheme, instructions, methodology, etc.) of a segmentation study. The assumption made by π, multi-π, and K that random chance is exemplified by equal coder distributions, as pointed out by Di Eugenio and Glass (2004, p. 96), does not likely hold for "the vast majority, if not all, of discourse—and dialogue tagging—efforts". This leads Di Eugenio and Glass (2004) to question whether π coefficients are suitable for such tasks. Cohen's and Fleiss' κ, though potentially more appropriate for determining reliability for discourse and dialogue tagging because of their lack of such an assumption, suffers from the bias, which leads to suspect values. Di Eugenio and Glass (2004) recommend reporting κ, but to mitigate its issues to also report π to indicate whether it is being affected by bias (as is done in Figure 3.10), and $2 \cdot A_a - 1$ to adjust κ for prevalence (as suggested by Byrt et al. (1993)). If one were to report $2 \cdot A_a - 1$ for Figure 3.9, they would notice that prevalence has an effect, as shown in

$A_a = 0.90,\ A_e = 0.905$	$A_a = 0.90,\ A_e = 0.5$
$\kappa = \pi = -0.048$	$\kappa = \pi = 0.80$
$2 \cdot A_a - 1 = 0.80$	$2 \cdot A_a - 1 = 0.80$
(a) Very prevalent category 'Accept'	(b) Equally prevalent categories

Figure 3.12: Coefficient values including $2 \cdot A_a - 1$ for the examples shown in Figure 3.9

Figure 3.12, where the prevalence of the category 'Accept' in Figure 3.12a is spotted by comparing the coefficient values to $2 \cdot A_a - 1$ and finding that they differ.

Craggs and McGee Wood (2005) take the opposite position of Di Eugenio and Glass (2004) and vehemently oppose the choice of κ-like coefficients—that model chance agreement as having individual coder distributions—in favour of π-like coefficients and Krippendorff's α (Craggs and McGee Wood, 2005, pp. 292–293)–which model chance agreement as having a single distribution. They argue that agreement must be representative of the coding process itself, "with coders being viewed as interchangeable".

> "The purpose of assessing the reliability of coding schemes is not to judge the performance of the small number of individuals participating in the trial, but rather to predict the performance of the schemes in general. The proposal that in most discourse and dialogue studies, the assumption of equal distribution between coders does not hold is, in fact, an argument *against* the use of Cohen's kappa"
>
> – Craggs and McGee Wood (2005, p. 292)

Their view is that by choosing an agreement coefficient that would model chance agreement as including when coders behave differently—as κ does—would not allow us to infer how a coding scheme would perform when applied by other coders, which is the main reason for attempting to quantify reliability in the first place.

Artstein and Poesio (2008, pp. 57–58) agree with Craggs and McGee Wood's (2005) recommendations in favour of preferring the usage of π (including multi-π or K) and Krippendorff's α because of their resiliency towards category and individual-coder bias. They note, however, that the issue of coder bias is often of little consequence, because such differences become negligible when the number of coders is increased (Artstein and Poesio, 2005), an effect which is also seen in the negligible bias values calculated by Fournier and Inkpen (2012, p. 160). To measure bias, Artstein and Poesio (2005, 2008)

define individual coder bias as the difference between expected agreement of π and κ, as shown in Equation 3.9 (Artstein and Poesio, 2005, p. 143).

$$Bias = A_e^\pi - A_e^\kappa \tag{3.9}$$

Artstein and Poesio's (2005) bias represents the degree to which individual coder bias affects κ. They demonstrated algebraically, and by example, that coder bias decreases as the number of coders is increased, so that "κ approaches π as the number of coders approaches infinity" (Artstein and Poesio, 2005, p. 146). Overall, Artstein and Poesio (2008, pp. 57–58) recommends reporting π (or multi-π or K) or Krippendorff's α, and emphasise that the usage of weighted versions of coefficients which treat errors of different types as having different severity should be used when possible, despite difficulty in interpreting them. Most importantly, however, they recommend that in addition to reporting coefficient values that studies should report

> "the methodology that was followed to collect the reliability data (number of coders, whether they coded independently, whether they relied exclusively on an annotation manual). The study should also indicate whether agreement was statistically significant, and provide the confusion matrix [e.g., Figures 3.9-3.10] or agreement table so that readers can find out whether overall figures of agreement hide disagreements on less common categories".
>
> – Artstein and Poesio (2008, p. 58)

Segmentation data reliability in practice Occasionally in literature introducing topical segmentation data, the calculation of inter-coder agreement coefficients is omitted. Instead, some works (e.g., Janin et al. 2003; Eisenstein and Barzilay 2008) assume that there is only one "true" reference segmentation[9] to compare against, so no measure of reliability is given, while some (Phillips, 1985; Richmond et al., 1994; Beeferman et al., 1997; Reynar, 1998; Choi, 2000) use completely artificial segmentation data[10] and codings

[9] An assumption which is incorrect, as demonstrated by the range of agreement values that differ from perfect agreement reported by similar studies, indicating that one coding cannot be truly representative of a whole population of even 'expert' coders.

[10] Phillips (1985) used the subtopic structure of chapters in science textbooks, Richmond et al. (1994, pp. 51–53) concatenated articles from an edition of The Times newspaper with each other and attempted to reconstruct the boundaries between articles, Beeferman et al. (1997) also used concatenated articles but from the Wall Street Journal (WSJ) corpus (Paul and Baker, 1992) and the Topic Detection and Tracking corpus (Allan et al., 1998), Reynar (1998) again used a subset of the WSJ corpus with concatenation, and

to bypass the issue of inter-coder agreement altogether (because there are no actual coders involved). A few works (Passonneau and Litman, 1993; Galley and Mckeown, 2003) report statistical significance from zero agreement using Cochran's test.

Of the works that do present chance-corrected inter-coder agreement coefficients (Table 3.3), many of the studies report low values for segmentation tasks—often even as trivial as denoting topic boundaries (Hearst, 1997; Carletta et al., 1997; Ries, 2001; Kazantseva and Szpakowicz, 2012). Many of these studies report low coefficient values because coders often approximately agree upon where segments lie, but disagree upon the exact placement of boundaries separating them (Artstein and Poesio, 2008, p. 40). This was noticed in the segmentation coding efforts of Hearst (1997); Passonneau and Litman (1997), and Carlson et al. (2003), and was also cited as the reason for the development of more lenient comparison methods than direct comparisons between boundary positions that take into account instances where coder boundaries had near-agreement in terms of their position (i.e., near-misses) (Pevzner and Hearst, 2002). For this reason, some works (Gruenstein et al., 2005; Malioutov and Barzilay, 2006; Kazantseva and Szpakowicz, 2012) report pairwise comparisons between coders using comparison methods[11] which award partial credit for near-misses, but do not simultaneously account for chance agreement.

Thus far, no sufficient agreement coefficient for either study reliability or data validity has been proposed that overcomes the issue of near-misses in segmentation. Hearst (1997, pp. 53–54) approached this goal by calculating Siegel and Castellan's K, but substituting for actual agreement the percentage agreement metric of Gale et al. (1992, p. 254). Unfortunately, this statistic suffers from the same gross over-estimation pitfalls as percentage agreement, despite the correction for chance agreement. Artstein and Poesio (2008, pp. 41–43) proposed that Krippendorff's (1995) method of measuring agreement (using Krippendorff α) on unitized segmentations may be appropriate, or the measurement of agreement upon windows produced by a segmentation comparison statistic such as P_k (Beeferman and Berger, 1999, pp. 198–200) or WindowDiff (Pevzner and Hearst, 2002, p. 10) may be appropriate. Both Krippendorff α and a window-comparison-method-based coefficient have the potential to allow for near-misses to be accounted for, but none of them have yet been demonstrated. This work proposes adapted inter-code agreement coefficients which account for near misses.

In Section 4.2, this work proposes and later experimentally evaluates adapted multi-π

Choi (2000) used 700 samples artificial text samples, where a sample is comprised of ten text segments selected from the first n sentences of a randomly selected document in the Brown corpus (Francis, 1964).

[11] Often P_k (Beeferman and Berger, 1999, pp. 198–200) and WindowDiff (Pevzner and Hearst, 2002, p. 10).

Study	*Type*	*Coders*	*Statistic*	*Value*
Kozima (1993)	Local coherence segmentation	16	None	None
Passonneau and Litman (1993)	Intention based, phrase level, majority comparison	7	%	0.70
Hearst (1993)	Topical, boundary/not	7	%	0.80–0.90
Rosé (1995)	Discourse acts and structure	4	$\bar{\kappa}$	.795, .895, .648
Isard and Carletta (1995)	Dialog transactions	1 expert 4 naïve	K	0.59
Passonneau and Litman (1997)	Intention based, phrase level, majority comparison	7	α	0.34
Hearst (1997)	Topical, boundary/not	7	$\bar{K}$	0.647
Carletta et al. (1997)	Transaction boundaries	4	K	0.57
Reynar (1998)	Boundary/not	2	K	0.764
Teufel et al. (1999)	Three zones types, and overall coding scheme	3	K	0.81, 0.71
Ries (2001)	Topic, boundary/not		K	0.36
Carlson et al. (2003)	Unit boundaries	6	K	0.87–0.97
Poesio and Patel (2006)	RDA annotation (Moore and Pollack, 1992; Moser and Moore, 1996)		K	0.9
Kazantseva and Szpakowicz (2012)	Topical, boundary/not	4–6	$\bar{\pi}^*/\bar{K}$	0.29± 0.15
Fournier and Inkpen (2012)	S-based agreement on data from Kazantseva and Szpakowicz (2012)	4–6	$\bar{\pi}^*_S/\bar{K}_S$	0.8904± 0.0392

Table 3.3: Reliability statistic values for segmentation corpora

and multi-κ coefficients that account for near misses by incorporating one of the new segmentation comparison methods proposed herein in Section 4.1.3. To evaluate automatic segmentations, an ideal segmentation comparison method and reliable manually coded data to use as a reference is a solid foundation on which to build, but a methodology is required to unify the process of evaluating segmentation. To that end, Section 3.3 discusses methodologies that can be used to evaluate automatic segmenters using an ideal segmentation comparison method and suitable multiply-coded reference segmentations.

3.3 Evaluating Segmenters

Experiment design Experiment design is the design of an effort to collect information in the presence of variation. In segmentation evaluation, the hypothesis often tested is

whether *there exists a significant difference in performance between a set of automatic segmenters*, and if this hypothesis is true, then *which is the best automatic segmenter?*. The testing of this segmentation-related hypothesis, and the answer to this question, can be obtained through an experiment that:

- Uses a suitable number of subjects so as to constitute a representative sampling;
- Randomly selects subjects from sources that contain enough diversity to support the breadth of the hypothesis;[12] and
- Controls for extraneous variables (e.g., removal of artificial segmentation breaks such as page breaks, illustrations, or other presentation features).

The definition of what constitutes a subject in a segmentation evaluation experiment is related to the definition of what constitutes a sample. Ideally, a subject would be defined as a boundary position contained within a manual sample. This granularity of analysis is not often feasible with the segmentation comparison methods available today, thus a subject in segmentation evaluation is often defined as the entire manual coding text/document (as opposed to the individual boundaries within). Each subject is then tested by applying an automatic segmenter to it (in medical literature this would be equivalent to a *treatment*), or more specifically, to the same document that was manually coded. This interpretation allows for the variable under test to be the automatic segmenter, and the documents and coders are controlled variables.

Not all subjects, however, are created equal. Subjects in segmentation can be divided by which coder produced them and which document (or even paragraph) that the manual sample describes. These divisions of subjects are called *factors*. Each factor has within it a variety of levels, e.g., the coder factor would contain the set of coders as levels. For segmentation, then, a design table that lists the relevant factors would resemble that shown in Table 3.4.

Table 3.4 shows 3 automatic segmenters over 5 documents each coded by 3 coders. Because the subjects under test are manual samples, if one applies automatic segmenters that each segment the same documents by the same coders (i.e., subjects), then we have produced *replicates*, or repetitions of an experiment. In the presence of this replication, when performing statistical hypothesis testing, one must then pick a test that appropriately models the sources of variance in this experiment design: a within-subjects (i.e., repeated measures) test (Cohen, 1995, pp. 287–307). If, instead, each automatic segmenter

[12] I.e., testing for the best segmenter for literature requires data from more types of texts than does the best segmenter for 19$^{\text{th}}$ century English sonnets

		Segmenter								
		A_1			A_2			A_2		
	Coder	C_1	C_2	C_3	C_1	C_2	C_3	C_1	C_2	C_3
Documents	Text 1	$x_{1,1,1}$	$x_{1,2,1}$	$x_{1,3,1}$	$x_{2,1,1}$	$x_{2,2,1}$	$x_{2,3,1}$	...		
	Text 2	$x_{1,1,2}$	$x_{1,2,2}$	$x_{1,3,2}$	$x_{2,1,2}$	$x_{2,2,2}$	$x_{2,3,2}$	...		
	Text 3	...	...	...	...	...	...			
	Text 4									
	Text 5									

Table 3.4: A two-factor within-subjects that is evaluating 3 automatic segmenters over 5 documents each coded by 3 coders where $x_{i,j,k}$ represents the output of a comparison method for an automatic segmenter i, coder j, and document k

		Experiment design	
		Between subjects	Within subjects
Data set properties	Normal and homoscedastic	ANOVA (Chambers et al., 1992) with Tukey's HSD (Miller, 1981; Yandell, 1997) post hoc	ANOVA (Chambers et al., 1992) with Tukey's HSD (Miller, 1981; Yandell, 1997) post hoc
	Non-normal or heteroscedastic	Kruskal-Wallis rank sum test (Hollander and Wolfe, 1973, pp. 115-120) with the Kruskal-Wallis rank sum multiple comparison test (Siegel and Castellan, 1988, pp. 213-214) post hoc	Friedman's rank sum test (Hollander and Wolfe, 1973, pp. 139–146) with the Wilcoxon-Nemenyi-McDonald-Thompson test (Hollander and Wolfe, 1999, p. 295) post hoc

Table 3.5: Statistical hypothesis tests used herein and their associated post-hoc tests for each design and whether their assumptions are met (normality and homoscedasticity)

was tested upon different subjects (i.e., did not segment the same documents and was compared against different coders), then we would select a statistical test that models the sources of variance as occurring between-subjects (Cohen, 1995, pp. 287–307). Specifically, the statistical hypothesis tests and their associated post-hoc tests utilized the most in this work are presented in Table 3.5.

Interpreting statistical hypothesis tests A significance level (α) is then specified, which is the threshold below which the probability of H_0 must be before one can reject in it favour of the alternate hypothesis.[13] A p-value is then calculated using the test statistic and procedure selected which represents the probability that given the distribution of

[13] "By convention, if $p < 0.05$ you say the result is statistically significant, and if $p < 0.01$ you say the result is highly significant and you can be more confident you have found a true effect." (Cumming et al., 2007)

H_0 is true that one would observe the distribution of H_1. If this p-value falls below the previously specified significance level (α), then there is cause to reject H_0 in favour of H_1. Depending upon α, a p-value may be reported as either statistically significant ($p < 0.05$) or highly statistically significant ($p < 0.01$), etc.

A visual example of this can be seen in Figure 2.4b, where confidence intervals for two estimated means (C and E) are compared for $n = 3$ and $n = 10$ showing p-values of 0.05 and 0.01 for differing amounts of confidence interval overlap. In this case, one is estimating the p-values by the degree of overlap of the confidence intervals shown, and illustrates the beneficial effect of increasing the sample size (n) so that one can better identify whether a difference is present or not (by decreasing the error of the estimated means).

Statistical hypothesis testing allows us to determine probabilistically whether one automatic segmenter is able to perform significantly differently from another given enough independent samples upon which to evaluate their performance, but it is not without error. By picking a significance level (α), or the threshold below which one rejects H_0, one accepts that the probability that H_0 was wrongly rejected (type I error) is α, and that the probability that H_0 was wrongly accepted (type II error) is β, where $1 - \beta$ is described as the power of a statistical test. The power of a statistical test ($1 - \beta$) is the probability that the selected statistical test and procedure will reject H_0 when it is false (Cohen, 1992). The power of a statistical test and procedure is a function of the "sample size (n), significance criterion (α), [and] population effect size (ES)" of a statistical test (Cohen, 1992, p. 156). The population effect size represents "the degree to which the H_0 is believed to be false", and can be "indexed by the discrepancy between H_0 and H_1" for a variety of statistical tests to allow us to determine the overall statistical power of a test (sometimes referred to as π)—for which a minimum of 0.80 is generally accepted (Cohen, 1992, p. 156). Analysis of the statistical power of a test can also be used to determine the minimum n needed to obtain statistical significance for a chosen α (Cohen, 1992, pp. 156–159).

Taking a text, one can collect manual segmentations of it. One can then select an appropriate segmentation comparison method which best quantifies the errors most relevant to our task, and using this method one can then compare an automatic segmentation to each of our sample manual segmentations. From the segmentation comparison method values, one can then perform hypothesis testing to determine whether the difference in their mean performances are significant or simply due to chance. Assuming that one has performed this evaluation over a suitably diverse set of documents, and that

one has demonstrated an appropriate degree of statistical significance, one can claim that one automatic segmenter consistently performs better than another for a particular application. If the segmenter also uses supervised machine learning, then cross-validation (Duda et al., 2000, pp. 483–485), bootstrap estimation (Duda et al., 2000, pp. 485–486), or maximum-likelihood model comparison (Duda et al., 2000, pp. 486–487), etc., can be used to assess how the segmenter may perform upon and generalize to unseen data.

Chapter 4

A Segmentation Evaluation Methodology Proposal

Current segmentation comparison methods such as WindowDiff and P_k have a variety of drawbacks which make their usage problematic: 1) They are not, by definition, metrics because they are not symmetric functions[1]; 2) They are sensitive to variations in internal segment sizes and thus lack consistency (Pevzner and Hearst, 2002); 3) Errors at the beginning and end of segmentations are not fully penalized (although this can be mitigated as shown by Lamprier et al. 2007); 4) Window size calculation rounding is not explicitly specified and not consistently applied; and 5) Quantifying error in terms of windows is not intuitively related to the units being segmented, making interpretation difficult.

This chapter proposes four new segmentation comparison methods which are: 1) Symmetric; 2) Insensitive to variations in internal segment sizes; 3) Unbiased (e.g., treats errors of the same type at any point in a segmentation equally); 4) Not based upon using windows to award partial credit for near-misses and instead use a new minimal edit distance to quantify the distance between two segmentations; and 5) More intuitive because their units are boundaries, boundary pairs, or potential boundaries as opposed to the indirect concept of analysing boundaries through windows. These comparison methods are referred to as *segmentation similarity* and *boundary similarity* (variants B_a, B_b, and B_c). Of these comparison methods, one is chosen as the recommended comparison method after evaluating them in a variety of experiments in Chapter 5.

This chapter proposes a new algorithm for unambiguously counting errors as the basis

[1]WindowDiff and P_k are not symmetric because they calculate their window size, k, from only one of the two segmentations compared.

of the four new comparison methods I proposed—named *boundary edit distance*. From this edit distance, a new method of constructing a multi-class confusion matrix that is able to partially award near misses is proposed from which metrics such as precision and recall can be computed. The resulting metrics are referred to as *B-precision*, *B-recall*, etc.

To describe these new comparison methods, this chapter first presents a conceptualization of segmentation and segmentation errors. Although this chapter restricts itself to discussing linear segmentation involving one or more mutually inclusive[2] types of boundaries, it can be applied to other segmentation types (e.g., mutually exclusive boundaries). This conceptualization leads to proposing *boundary edit distance* as an algorithm for quantifying these errors.

To normalize the error counts given by *boundary edit distance*, four normalizations are proposed in the form of four new segmentation comparison methods. These four comparison methods represent different forms of normalization which vary the units used and embody differing interpretations of what constitutes an error or the relative severity of an error.

Lastly, a discussion on an appropriate methodology for comparing segmentations using any comparison methods is presented.

4.1 Linear Segmentation Comparison

4.1.1 Conceptualizing Segmentation

Linear segmentation is the task of splitting an item, such as a document, into a linear sequence of segments using some criterion by placing boundaries between these segments. This boundary placement criterion is often the subject of one's study—especially when analysing the task of topic segmentation—and is defined by a description of the task. The specific task being studied often varies wildly—from sentence splitting to morphological segmentation, and for this section no specific task is assumed, but many examples are given of topic segmentation.

For the purposes of discussing how one can conceptualize and describe differences between segmentations, let us focus upon a series of hypothetical topical segmentations of a short excerpt. The excerpt used herein is from Coleridge's (1816) poem titled Kubla Khan (Coleridge, 1816, pp. 55–58), shown below in Figure 4.1, chosen for its brevity.

[2]I.e., boundaries of different types can occupy the same position in a segmentation.

1. In Xanadu did Kubla Khan
2. A stately pleasure-dome decree:
3. Where Alph, the sacred river, ran
4. Through caverns measureless to man
5. Down to a sunless sea.
6. So twice five miles of fertile ground
7. With walls and towers were girdled round:
8. And here were gardens bright with sinuous rills,
9. Where blossomed many an incense-bearing tree;
10. And here were forests ancient as the hills,
11. Enfolding sunny spots of greenery.

Figure 4.1: Excerpt from Kubla Khan (Coleridge, 1816, pp. 55–58)

Segmentations can occur on a variety of levels of granularity (e.g., at the word, phrase, sentence, paragraph level, etc.). A level of granularity is the smallest indivisible unit which comprises the segments in a segmentation. Figure 4.1 shows poem line numbers, and for the task of topical segmentation upon poetry, the level chosen is—arbitrarily—the line level. For the purposes of discussion, some notation is defined, where:

1. D is the sequence of atomic textual units (i.e., units) which comprise the document being segmented; and

2. A segmentation s_i of D is an ordered sequence of units in D that are described (in varying ways) as belonging to segments.

If one was to ask a human to topically segment, at the line level, the excerpt in Figure 4.1, they would produce a segmentation indicating the segments that they believe exist, separated by topic boundaries. To describe these segmentations, a human can indicate, for each segment, which lines (i.e., units) they span and also a short description of the segment, as in Figure 4.2. This procedure is similar to that used by Hearst (1997) and Kazantseva and Szpakowicz (2012) except that they asked for segmentations at the paragraph-level because of the length of their documents.

Lines	Description
1–2	Kubla Khan and his decree
3–5	Waterways
6–11	Fertile ground and greenery

Figure 4.2: A hypothetical manual line-level segmentation of the excerpt in Figure 4.1

Representing segmentation Mathematically, one could represent this single-boundary-type segmentation as a sequence of three segments. This allows one to see where the boundaries lie between internal segments. Each internal segment could also be represented as containing the sequence of line numbers (i.e., units of text) that they represent (as in Figure 4.3). Representing segmentations as a sequence allows the order of the internal segments and textual unit references within to be maintained. The size of each internal segment is then the cardinality of this sequence of line numbers. An advantage of this segmentation representation allows one to relate it back to the ordered sequence of textual units which compose the document under analysis and to represent multiple boundary types.

$$s_1 = \big((1,2),(3,4,5),(6,7,8,9,10,11)\big)$$

Figure 4.3: A sequence representation of the line-level segmentation in Figure 4.2

If there is no need to calculate any descriptive statistics upon the actual textual units under observation, then a more succinct representation can be used. Instead of representing each internal segment as a sequence of line numbers, internal segments can be represented by the cardinality of each internal segment calculated using Equation 4.1. This representation still captures the size (in units) of each of the internal segments and the relative position of the boundaries contained within a segmentation as being between the internal segments themselves.

$$ps_i \equiv \text{ps}(s_i) = \coprod_{j=1}^{|s_i|} (|s_{i,j}|) \tag{4.1}$$

Obtaining the cardinality sequence of the segmentation s_i gives us the much more compact sequence shown in Figure 4.4.

$$\text{ps}(s_1) = (2,3,6)$$

Figure 4.4: A sequence of the cardinality of each internal segment sequence in Figure 4.3

Descriptive statistics Using a cardinality-sequence representation of a segmentation, one can begin to calculate some descriptive segmentation statistics.

The most basic descriptive segmentation statistics relate to the size of a segmentation. Total number of textual units being analysed can be determined by either taking the

cardinality of the text being segmented (i.e., $|D|$), or taking the sum of the cardinalities of a segmentation (s_i) of that text (i.e., $|D| = \sum_{j=1}^{|\mathrm{ps}_i|} \mathrm{ps}_i[j]$). One can also determine the number of potential boundary positions that exist in a document D as the number of units minus one (Equation 4.2; adapted from Passonneau and Litman 1993, p. 150). To determine the number of boundaries placed in a linear segmentation s_i, one can take the cardinality of the segmentation's cardinality sequence representation (ps_i) minus one (Equation 4.3).

$$\mathrm{pb}(D) = |D| - 1 \qquad (4.2) \qquad\qquad \mathrm{b}(s_i) = |\mathrm{ps}(s_i)| - 1 \qquad (4.3)$$

Applying Equations 4.2 and 4.3 to s_1 and ps_1, the number of potential boundaries is $\mathrm{pb}(D) = |D| - 1 = 11 - 1 = 10$ and the number of boundaries placed by the manual segmenter is $\mathrm{b}(s_1) = |\mathrm{ps}_1| - 1 = 3 - 1 = 2$. Descriptive statistics related to size can begin to help describe the differences between two arbitrary segmentations, but they cannot fully describe how the boundaries placed within two segmentations differ.

Graphically representing segmentation Although the size differences between two segmentations cannot fully describe how the boundaries placed within two segmentations differ, they offer some intuition when represented graphically. The sequence of segment cardinalities (ps_1) can be used to visually represent a segmentation as having $\mathrm{pb}(D) = 11$ textual units separated into $|\mathrm{ps}_1| = 3$ internal segments divided by $\mathrm{b}(s_1) = 2$ boundaries, as in Figure 4.5. A label (s_1) denotes the segmentation's name for later reference, and the internal segment boundaries are denoted simply by the edge of each internal segment.

Figure 4.5: A graphical representation of the cardinality sequence in Figure 4.4

This visual representation can succinctly illustrate one segmentation, but it is more powerful when it is used to compare multiple segmentations. If a second segmentation of the excerpt in Figure 4.1 such as the one shown in Figure 4.6 was provided, it could be graphically compared to s_1, as shown in Figure 4.7 as s_2.

Figure 4.7 shows the left and right edges of the internal segments (i.e., boundaries) of s_1 not aligning horizontally with the edges of s_2's segments. In this graphical representation, one can begin to see the differences between the boundaries placed within the two segmentations.

Lines	Description
1–3	Kubla Khan's location
4–10	The land's features
11	The time of day

Figure 4.6: Another hypothetical manual line-level segmentation of Figure 4.1

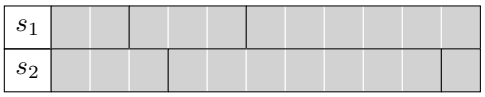


Figure 4.7: Two hypothetical segmentations of the text of Figure 4.1

Visually comparing segmentations When visually comparing these two adjacent segmentations, the differences in their boundary positions are apparent, but how can we describe them? One could adopt the information retrieval (IR) perspective where one segmentation is considered a reference (arbitrarily choosing s_1 herein), and the other a hypothesis (s_2). With these designations, each boundary can be labelled as either a false positive (FP) or false negative (FN). The pair of segmentations form Figure 4.7 then have two FNs and two FPs as shown in Figure 4.8.

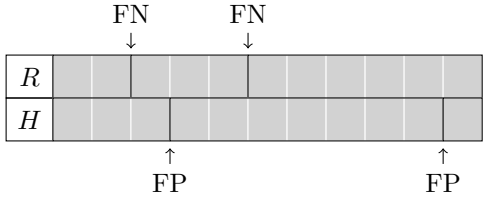


Figure 4.8: Two hypothetical segmentations labelled from an IR perspective

Adopting this IR perspective is not ideal, however, because IR metrics cannot award partial credit for what one could consider to be a nearly missed boundary in the first portion of the two segmentations compared. Another issue with IR comparison methods is that they are not symmetric because they require one segmentation to be designated as a hypothesis and the other a reference. An alternate to the IR perspective of segmentation comparison is needed.

In this work, it is proposed that the first two boundaries placed in the segmentations can be considered a near miss from the perspective of either s_1 or s_2 and that they are off by 1 PB (the FN is at position 3, the FP is at position 4, the offset is then $|3-4| = 1$ PB). The rest of the boundaries could then be considered additions/deletions—as if one was editing one segmentation to become the other through a series of edit operations. From

the perspective of s_1 a deletion followed by an addition occurred (shown in Figure 4.9).

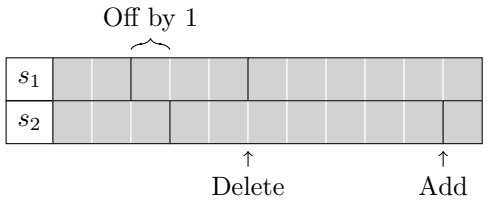


Figure 4.9: Two hypothetical segmentations with one pair of boundaries off by one unit, and one deleted, and added (from s_1's perspective)

Ideally, however, when one compares two segmentations, one's quantification of the differences (i.e., errors) should be independent of whether one segmentation is considered a reference or a hypothesis, (i.e., symmetric). To quantify differences symmetrically, this work defines the differences between two segmentations as whether a boundary is either off by n PBs (a near miss), or whether a boundary is added/deleted (a full miss). Additions and deletions are counted as being of the same conceptual edit operation, as shown in Figure 4.10, allowing for a perspective-independent comparison.

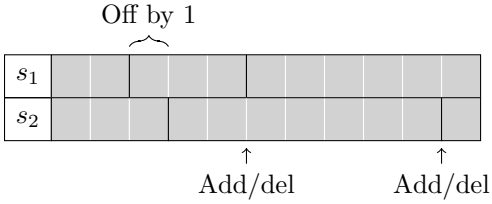


Figure 4.10: Two hypothetical segmentations labelled with symmetric edit operations

Using these two graphical operations (off-by-n and addition/deletion), this work is able to quantify all types of dissimilarities between two single-boundary-type segmentations independently of perspective. These edit operations are only being applied graphically, however. How can we more formally describe these graphical operations and algorithmically identify these edit operations?

Quantifying segmentation differences To quantify the differences between two segmentations, this work uses a sequence representation of the boundaries themselves to define the edit operations informally described previously (off-by-n and addition/deletion). First, potential boundary positions are represented as sets of boundary types so that multiple boundaries can be located at one place. These position sets are then placed within a sequence so that they lie between each textual unit in a document, forming a

boundary string, or bs_i. Each boundary string would be obtained from the annotations provided either by a human or automatic segmenter. Taking s_1 and s_2, for example, the boundary position set sequences would appear as in Figure 4.11, with each sequence containing $\text{pb}(D) = 10$ potential boundary sets.

$$bs_1 \equiv \text{bs}(s_1) = \big(\{\}, \{1\}, \{\}, \{\}, \{1\}, \{\}, \{\}, \{\}, \{\}, \{\}\big)$$
$$bs_2 \equiv \text{bs}(s_2) = \big(\{\}, \{\}, \{1\}, \{\}, \{\}, \{\}, \{\}, \{\}, \{1\}, \{\}\big)$$

Figure 4.11: Two sequences of boundary type sets for segmentations s_1 and s_2, referred to as boundary strings bs_1 and bs_2

When representing boundary strings, this work could have chosen to simply represent boundary positions as a sequence of digits, where 1 represents the presence of a boundary and 0 not. Using a boundary set at each potential boundary position instead of a binary representation allows this work to easily represent more than one type of boundary at the same position (shown later). In the single-boundary-type example shown in Figure 4.11, the relationship to the graphical representation in Figure 4.10 is easy to see: bright lines are now empty sets and solid black lines are now sets containing the digit 1. The locations in the boundary string where the edit operations can be applied to transform one boundary string into another are also visually apparent.

Using boundary strings, this work quantifies the differences between two segmentations using set operations such as union ($A \cup B$), intersection ($A \cap B$), difference ($A \setminus B$), symmetric difference ($A \Delta B$), etc. These operations are applied to pairs of boundary sets, where each pair is comprised of two boundary sets; e.g., a boundary set A comes from a specific position in bs_A, and B comes from the same position from bs_B. Using set operations upon pairs, this work creates functions to identify and count the off-by-1 (i.e., near miss) and addition/deletion edit operations graphically shown earlier. When multiple boundary types are present, a third operation can also be identified, referred to as a boundary *substitution* (shown later).

Counting addition/deletion or substitution edit operations between boundary strings can be accomplished by comparing the differences between boundary sets at the same position in either boundary string. If only a count of the number of differences is desired, then we can use the symmetric differences between the two sets divided by two (to count substitutions) and rounded up (to count remaining additions/deletions) as in Equation 4.4. Using this equation to count differences between two boundary type sets is ideal because it is zero when comparing identical sets (even if they are both empty) and a positive

integer otherwise.

$$\text{count}_\delta(A, B) = \left\lceil \frac{|A \Delta B|}{2} \right\rceil \tag{4.4}$$

When A and B have identical cardinality, but differing contents, then this difference is recorded as an edit (e.g., $\text{count}_\delta(\{1\}, \{2\}) = 1$). This edit represents a substitution operation where during segmentation a segmenter made the error of choosing the wrong type of boundary to place, but chose the correct location.

When A and B have differing cardinality, e.g., $\text{count}_\delta(\{1\}, \{1, 2\}) = 1$, this represents an addition/deletion edit. When both an addition/deletion and a substitution occur simultaneously, e.g. $\text{JN}_\delta(\{1\}, \{2, 3\}) = 2$, the numerator is able to quantify both errors simultaneously. This work uses Equation 4.4 to quantify the total number of addition/deletion and substitution edits that occur between two segmentations as its sum over all positions, as in Equation 4.5.

$$\text{set\_edits}(A, B) = \sum_{i=1}^{\text{pb}(|A|)} \text{count}_\delta(\text{bs}(A)_i, \text{bs}(B)_i) \tag{4.5}$$

Applying this to the two hypothetical segmentations s_1 and s_2, four set errors are identified. These set errors can be graphically represented as occurring at the locations indicated in Figure 4.12. Counting set edits is essentially the same as counting FPs and FNs as if they were the same conceptual error, and is unfortunately not much of an improvement because it cannot identify near misses.

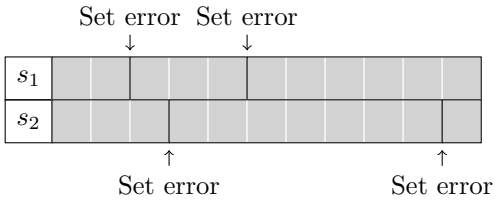


Figure 4.12: Two hypothetical segmentations labelled with set errors

When set errors are counted in segmentations containing multiple boundary types, the number of substitution errors that occur is also counted, which is an improvement over counting FPs and FNs. It would also be informative to know how many substitutions occur as opposed to additions/deletions. This differentiation can be made by modifying Equation 4.4 to count substitutions and additions/deletions separately, as in Equation 4.6 and 4.7, respectively. This presents a more detailed picture of the differences between

two segmentations in the form of descriptive statistics about the quantity and types of edits (i.e., errors) that occur.

$$\text{count\_ad}_\delta(A, B) = \text{abs}\big(|A| - |B|\big) \tag{4.6}$$
$$\text{count\_sub}_\delta(A, B) = \text{count}_\delta(A, B) - \text{count\_ad}_\delta(A, B) \tag{4.7}$$

Just as in Equation 4.5, the sum of addition/deletion and substitution edits can be computed as in Equation 4.8 and Equation 4.9.

$$\text{ad\_edits}(A, B) = \sum_{i=1}^{\text{pb}(|A|)} \text{count\_ad}_\delta(\text{bs}(A)_i, \text{bs}(B)_i) \tag{4.8}$$
$$\text{sub\_edits}(A, B) = \sum_{i=1}^{\text{pb}(|A|)} \text{count\_sub}_\delta(\text{bs}(A)_i, \text{bs}(B)_i) \tag{4.9}$$

To demonstrate the separate counting of both addition/deletion and substitution, an example comprised of two segmentations containing multiple boundary types is required. For this demonstration, it is assumed that there are three different types of boundaries for poetry segmentation, e.g., a simple topic shift (1), a location-related topic shift (2) and a temporal topic shift (3).[3] It is also assumed that these boundary types can occur simultaneously at the same position. Using these assumptions, one could hypothetically segment the excerpt in Figure 4.1 as s_3, shown as a boundary string in Figure 4.13 alongside s_1.

$$\text{bs}(s_1) = \big(\{\}, \{1\}, \{\}, \{\}, \{1\}, \{\}, \{\}, \{\}, \{\}, \{\}\big)$$
$$\text{bs}(s_3) = \big(\{\}, \{2, 3\}, \{\}, \{\}, \{\}, \{1\}, \{\}, \{\}, \{3\}, \{\}\big)$$

Figure 4.13: Hypothetical multiple-boundary-type segmentations as boundary strings

This work uses the notation that boundary types in a segmentation of a document D for a specific task are the set T_b, where $\text{T}_b = \{1, 2, 3\}$ for Figure 4.13. Because in this example there can be up to $|\text{T}_b|$ boundaries at each position, this work defines the total number of PBs in a segmentation as the number of textual units minus one multiplied by the number of boundary types as in Equation 4.10. The number of boundaries placed in a segmentation s_i is then determined using Equation 4.11.

[3] A more realistic example is a play which has both act and scene breaks which occur simultaneously.

$$\mathrm{pb}(D) = |\mathrm{T}_b| \times (|D| - 1) \qquad (4.10) \qquad \mathrm{b}(s_i) = \sum_{j=1}^{\mathrm{pb(D)}} |\mathrm{bs}(s_i)_j| \qquad (4.11)$$

Visualizing multiple boundary types Visually, we can represent both s_1 and s_3 as in Figure 4.14. This graphical representation is nearly identical to that shown before: it allows one to see where boundaries lie and the relative size of each internal segment, but now it also indicates above/below each boundary what types of boundaries are present[4] at each position. In this example, multiple boundaries at one position are also shown, meaning that boundary placement in this task is mutually inclusive.

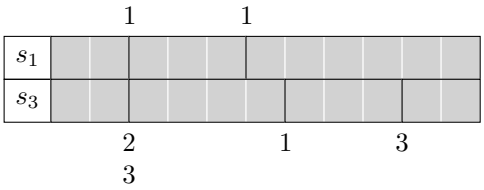


Figure 4.14: Two hypothetical multiple-boundary-type segmentations

Points where multiple boundaries have been placed at the same position present the possibility for additions/deletions to occur simultaneously with substitutions at the same position. Graphically, we can see that at least one substitution (Sub) is present, and four additions/deletions (AD), as shown in Figure 4.15.

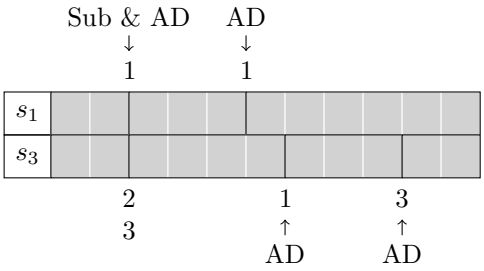


Figure 4.15: Two hypothetical multiple-boundary-type segmentations labelled with addition/deletion and substitution edits at locations where they occur

To quantify the differences between these two segmentations containing multiple boundary types, we can apply set_edits(bs_1, bs_3) (Equation 4.5), giving us 5 set edits. To determine how many substitutions are present, sub_edits(bs_1, bs_3) can be used

[4]The set of boundary types $\mathrm{T}_b = \{1, 2, 3\}$

(Equation 4.9). To instead determine how many additions or deletions are present, ad_edits(bs_1, bs_3) can be used (Equation 4.8). Applying both equations, this gives 1 substitution (Sub) and 4 addition/deletion (AD) edits, which matches what can be visually identified in Figure 4.15. The ability to discern between these two types of errors is useful, but how can near misses (i.e., off-by-n edits) be counted?

Off-by-n boundaries When analysing the differences between two segmentations, the focus of this work thus far has been upon identifying and counting additions/deletions, and substitutions of boundaries. Near misses between two segmentations, or off-by-n errors, are prevalent in both manual segmentations and in comparisons between automatic and manual segmentations. They are so prevalent that whole metrics have been devised to award partial credit for near misses (e.g., WindowDiff and P_k) as opposed to interpreting them as two full misses (e.g., FPs, FNs, additions, deletions, etc.). Visually, identifying off-by-n errors has been shown by the comparison between s_1 and s_2, where in this case $n = 1$ PB (as shown in Figure 4.16). A graphical method of finding off-by-n errors is insufficient because there is ambiguity in the interpretation of how close two boundaries must be to be considered a near miss. Instead, a mathematical method of counting off-by-n errors is desired by this work to create a new segmentation comparison method.

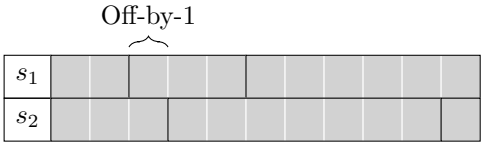


Figure 4.16: Two hypothetical segmentations labelled with set errors

Going back to the boundary string representation of these two segmentations (Figure 4.11), this work counts these errors again using set operations. From this representation, one can see where an off-by-n error occurs at the second and third pair of boundaries between the two boundary stings. Using this representation, which set operations can be used to count these off-by-n errors, and what edit operation can be used to model an off-by-n error?

When editing one segmentations to appear as another this work uses *transposition* edit operations to model off-by-n errors. A transposition is the act of swapping one item in a sequence with an adjacent item (e.g., the string 'AB' becomes 'BA'). Much like boundary set sequences, strings of characters are also sequences, and the addition/deletion and substitution operations described earlier mirror those used by string edit distances such as

Levenshtein edit distance (Levenshtein, 1966) and the boundary edit operation counting of Hearst (1993, p. 6) and Richmond et al. (1994, p. 53). Transpositions, however, have not previously been applied to segmentation.

Transpositions were introduced to the conceptually similar task of quantifying string edit distance by Damerau to handle the common error of swapped letter positions in misspellings. The resulting string edit distance that included transpositions is often referred to as Damerau-Levenshtein edit distance.

In Damerau-Levenshtein edit distance, only adjacent transposition are considered (e.g., 'ture' and 'true' for identifying misspellings), but in segmentation Hearst (1993, p. 6) and Richmond et al. (1994, p. 53) demonstrated that boundaries can be off by $n > 1$ PBs (i.e., not adjacent) and yet can still arguably considered to be near misses as opposed to full misses (additions/deletions). To quantify near misses in segmentation upon boundary strings, this work proposes identifying transpositions over n PBs, herein referred to as n-wise transpositions, or simply transpositions. Such an operation is preferable for modelling near misses because it unambiguously identifies a pair of boundaries from two segmentations that are off by n PBs, as opposed to using windows which indirectly quantify near misses as proportions of windows in error.

This work defines n-wise transpositions to be the swapping of the first and last items in a sequence A (where $|A| = n$), as in Equation 4.12. Applied to an example set of characters, $\text{transpose}(\{\text{A}, \text{B}, \text{C}\}) = \{\text{C}, \text{B}, \text{A}\}$ and $\text{transpose}(\{\text{A}, \text{B}, \text{C}, \text{D}\}) = \{\text{D}, \text{B}, \text{C}, \text{A}\}$; notice that the inner portion of the sequence remains unchanged.

$$\text{transpose}(A) = \begin{cases} |A| > 2 & A_{|A|} \parallel \left(\parallel_{i=2}^{|A|-1} A_i \right) \parallel A_1 \\ |A| = 2 & A_2 \parallel A_1 \\ |A| < 2 & A \end{cases} \tag{4.12}$$

To apply n-wise transpositions to boundary strings, this work proposes that transpositions be applied to pairs of sub-sequences taken from boundary strings. These sub-sequences would describe the same positions in the two segmentations compared, and each sub sequence would be comprised of n boundary sets.

To count n-wise transpositions alone, this work iterates over the boundary sets selecting a pair of boundaries from each segmentation at position i and then another pair at position $i + n - 1$ (as in Equations 4.13–4.14). Only two positions are selected because since only the start and end of a sub-sequence are swapped, only those two positions need to be compared.

$$\text{transpositions\_n}(A, B, n) = \sum_{i=1}^{\text{pb}(|A|)-n+1} |\text{pairs}(\text{bs}(A), \text{bs}(B), n, i)| \tag{4.13}$$

$$\text{pairs}(A, B, n, i) = \text{transpositions\_at\_position}(A_i, A_{i+n-1}, B_i, B_{i+n-1}) \tag{4.14}$$

To count the transpositions present in the two selected pairs of boundary sets, set operations are used. Specifically, the intersection of the symmetric difference between each pair (as in Equation 4.15) gives the set of boundaries that can participate in transpositions. Taking the cardinality of this set gives the maximum number of mutually possible transpositions within those pairs.

$$\begin{aligned}\text{transpositions\_at\_position}(A_i, A_{i+n}, B_i, B_{i+n}) = \\ (A_i \Delta B_i) \cap (A_{i+n} \Delta B_{i+n}) \cap (A_i \Delta A_{i+n}) \cap (B_i \Delta B_{i+n})\end{aligned} \tag{4.15}$$

Transpositions could occur over a variety of distances, however (e.g., 1, 2, 3 units apart). To determine the total number of n-wise transpositions in a segmentation, the maximum distance (in PBs) that a near miss is considered to span must be decided upon (a discussion of this complex issue is provided in Section 5.2.1.1). This is referred to as the maximum transposition spanning distance (n_t). Once this number has been selected, to count the number of transpositions within a segmentation this work takes the sum of transpositions found for each n_i from $2 \leq n_i \leq n_t$ PBs, as in Equation 4.16. A minimum of 2 PBs is used because a transposition must occur over at least two positions, and cannot occur over a single position (this would be a matching boundary). For simplicity, it is also assumed that transpositions can only occur between the same boundary type.

$$\text{transpositions}(A, B, n_t, t) = \sum_{n_i=1}^{n_t} \text{transpositions\_n}(A, B, n_i, t) \tag{4.16}$$

Evaluating the function $\text{transpositions\_n}(s_1, s_3, 2)$ gives us a count of one 2-wise transposition (2-wise T) as labelled on the top of Figure 4.17. This matches what is graphically expected and does not count the other boundaries as near misses. Evaluating the function $\text{transpositions\_n}(s_1, s_3, 3)$ instead finds a different set of transpositions: one 3-wise transposition (3-wise T), as labelled on the bottom in Figure 4.17. This means that evaluating the function $\text{transpositions}(s_1, s_3, 2) = 1$, which is expected, but that

transpositions$(s_1, s_3, 3) = 2$ because the function looks for transpositions spanning both $n = 2$ and $n = 3$ PBs.

Counting two transpositions when $n_t = 3$ is expected, but not ideal, because the 2-wise transposition overlaps a boundary of the same type in the 3-wise transposition. If both operations were applied to edit one segmentation, then the two segmentations would not be identical. Unfortunately, this means that a simple sum of the number of transpositions for differing spanning lengths is inadequate to produce a count of the errors in a segmentation as edits operations. A consistent method of choosing which transposition to apply and which to discard is then required for situations where transpositions of differing spanning sizes overlap. Without this consistent method, a consistent comparison method cannot be constructed using the segmentation edit operations defined herein.

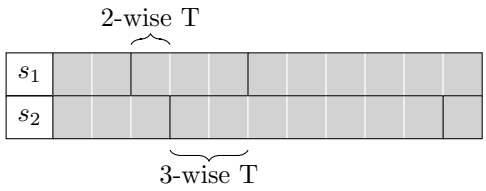


Figure 4.17: Two hypothetical segmentations labelled with two n-wise transpositions where $n = 2$ (2-wise T) and $n = 3$ (3-wise T)

4.1.2 Boundary Edit Distance

In the previous section, three types of edit operations were proposed for segmentation: additions/deletions, substitutions, and n-wise transpositions. These edit operations have the potential to overlap—a transposition can take the place of two additions/deletions, and two transpositions could attempt to transpose the same boundary. A method of unambiguously identifying the minimum number of edits, and their placement, is needed to create a comparison method based upon these edit operations. This section proposes a globally optimal minimum edit distance between the boundary strings of two segmentations that is suitable to be used to construct a comparison method.

A boundary edit distance is not an entirely foreign desire. Beeferman et al. (1997, p. 42) proposed that an edit distance incorporating addition and deletion operations would be beneficial, but stopped short of proposing an actual edit distance and instead proposed P_k. Both Hearst (1993, p. 6) and Richmond et al. (1994, p. 53) count off-by-n PB, addition, and deletions differences, but they did not attempt to create an edit distance to

consistently apply these operations. Instead, human interpretation was required to label where edit operations occurred.

Levenshtein string edit distance (Levenshtein, 1966) has the potential to serve as a segmentation edit distance that could identify additions/deletions in a boundary string when boundaries are mutually exclusive. Unfortunately, such an edit distance could not be applied to segmentations that allow multiple boundaries to occur at the same location and would not identify near misses. Damerau-Levenshtein string edit distance (Damerau, 1964; Levenshtein, 1966) adds transpositions between adjacent characters, which would enable the detection of off-by-1 errors. Unfortunately, such an edit distance could not cannot handle transpositions spanning more than 2 PBs (i.e., non-adjacent PBs). Damerau-Levenshtein edit distance also has the same limitation as Levenshtein distance of not being able to support multiple boundary types.

With no existing suitable edit distance that can be adapted for multiple and mutually-inclusive boundary type segmentations, a new boundary string edit distance is proposed herein. This globally optimal minimum edit distance operates upon boundary strings with one or more boundary types and mutually-inclusive/exclusive boundaries using three edit operations:

- *Addition/deletion* of individual boundaries of a particular type from a boundary set at a position i;
- *Substitution* of one boundary type for another at a position i; and
- *n-wise transposition* of the same boundary type from one boundary set at position i and an opposite set at position j, where $j - i + 1 = n$ PBs.

The edit distance proposed herein does not use dynamic programming as Levenshtein and Damerau-Levenshtein use. Instead, this new edit distance counts potential locations where additions/deletions and substitutions can occur and selectively adds transpositions of the smallest spanning size possible. When selectively adding transpositions, care is taken to ensure that they are only added when they:

- Replace two additions/deletions (reducing error from 2 to 1); or
- Replace one addition/deletion and one substitution (changing the interpretation of the edits to 1 transposition 1 one addition/deletion and not needlessly increasing the number of edits);

Replacing two substitutions can lead to an increase in edit distance (e.g., creating 1 transposition and 2 additions/deletions from 2 substitutions), so this is prohibited by design.

Transpositions in this new edit distance are added in increasing spanning size; if the maximum transposition spanning distance (n_t) is 4, then 2-wise transpositions are applied, then non overlapping 3-wise transpositions, etc. This order is used because it is assumed that lower-spanning-size transpositions are more probable interpretations of errors than greater spanning sizes, e.g., placing a boundary off by 1 unit is more likely to be a near miss error than being off by 5 units. This work assumes that the further apart two boundaries are, the greater the probability that two segmenters meant to place two different boundaries as opposed to placing the same boundary but were slightly off in position.

After all transpositions are identified, the rest of the edit operations are then applied to the remaining untransposed boundaries. First substitutions are applied, and then the remaining un-substituted boundaries are recorded as additions/deletions.

The new edit distance proposed herein is referred to as *boundary edit distance*, and its formulation can be summarized as the following high-level steps:

1. Record the locations and boundaries involved in each edit operation type:
   (a) Record the locations where potential addition, deletion, and substitution errors could occur as O_S;
   (b) For each n_i in N_t (where $N_t = 2 \leq n_i \leq n_t$ and n_t is the maximum transposition spanning distance), from smallest to largest n_i, iterate in order through boundary string positions and record where a potential n_i-wise transposition occurs in separate sets (i.e., a set T_{n_i} for each n_i), ensuring that they:
      - Only transpose two boundaries from two different positions between the two segmentations;
      - Only transpose boundaries of the same type;
      - Do not overlap those that have been already recorded;
      - Do not displace two potential substitutions; and
      - Remove from O_S those boundaries involved as transpositions that are recorded (so that future transpositions added will be able to determine whether they are replacing two substitutions or not);
2. Add recorded transpositions to the edit set (E) and transposition edit set (T_e);

3. From the entries in remaining O_S, add all substitution operations that can be identified to the substitution edit set (S_e) and E and record which segmentation each boundary type was transposed in;

4. Add all remaining entries in O_S as addition/deletion operations to A_e and E and record from which segmentation the boundary was located in;

5. Edit distance is |E|, with a breakdown of the error types as the:

    - Number of additions/deletions $|A_e|$;
    - Number of substitutions $|S_e|$;
    - Number of transpositions $|T_e|$ (with counts per n_i available as $|T_{n_i}|$).

Using these steps, counts of additions/deletions, substitutions, and n-wise transpositions are recorded. Although these edit operations are symmetric, information related to which segmentation contained a boundary in an addition/deletion or substitution operation are still recorded. Recording this boundary information allows for more detailed error analysis such as decomposing additions/deletions into FPs and FNs (shown later).

To more formally define this algorithm, a pseudocode version of it is provided which closely parallels the software implementation of the algorithm used to generate results shown later. In this pseudocode, a data structure referred to as an associative array is used (created by calling a function titled dict() where an associative array Υ returns a stored value ϱ by supplying a key y as $\varrho \leftarrow \Upsilon[y]$), and the time complexity ($\mathcal{O}(\cdot)$ notation) of the various algorithms is analysed. An associative array is essentially a hash-table, and an amortised analysis of the time complexity of operations upon them is assumed to cost $\mathcal{O}(1)$ for each operation involving such a data structure. Associative arrays were chosen because the positions in a boundary set string upon which edit operations occur will often be very sparse. This sparsity is caused by the inherent sparsity of boundary placement in segmentations themselves. This representation lends itself to an efficient use of memory while not compromising overall amortised time complexity.

The outline of boundary edit distance given previously is represented in pseudocode by Algorithm 4.1, where a number of functions are called that correspond to sub-points in the previous outline. The algorithm goes through the steps of finding potential set errors (O_S) such as additions/deletions and substitutions and then identifies transpositions (T_e). Once transpositions are identified, and overlapping entries in O_S removed, addition/deletion and substitution edits are determined per position and recorded in A_e and S_e, respectively. These sets, along with the total set of edits applied (E), are then returned. The time

complexity of this function is approximately $\mathcal{O}(n)$, where n is the number of positions in O_S, but subsequent function calls are more time intensive. Individual function complexities are analysed and an overall time complexity given later.

To calculate potential set edits such as additions/deletions and substitutions, Algorithm 4.2 is applied. This function records a sequence of tuples of the form (d, a, b) that represent the differences between the boundaries found in one boundary set versus the corresponding boundary set at the same position in the other boundary string. These are then stored in the associative array O_S, which associates the position i with the tuple recorded. The time complexity of this function is approximately $\mathcal{O}(n)$, where n is the number of boundary sets in bs_a.

To identify transpositions, the associative array of potential/optional set edits (O_S) and the set of transposition spanning lengths (n_t) is used by Algorithm 4.3. This function determines where to place transpositions and where to remove any potential/optional set edits that transpositions would overlap. An associative array of transpositions which stores, for each entry, a list of those transpositions which occur at a particular position is recorded as O_T. The overall set of transpositions is initialized as an empty set (T_e). Transposition operations are identified first for the lowest n_i values, and then higher, by iterating over a sorted sequence of the set N_t.

Algorithm 4.1 boundary_edit_distance

```
1:  function BOUNDARY_EDIT_DISTANCE(bs_a, bs_b, N_t)
2:      O_S ← OPTIONAL_SET_EDITS(bs_a, bs_b)
3:      T_e ← TRANSPOSITIONS(bs_a, bs_b, N_t, O_S)
4:      A_e ← {}
5:      S_e ← {}
6:      for i, o ∈ O_S do
7:          d, a, b ← o                          ▷ Decompose tuple into multiple variables
8:          A_o, S_o ← ADDITIONS_SUBSTITUTIONS_SETS(d, a, b)
9:          for a_o ∈ A_o do
10:             A_e ← A_e ∪ {(a_o, i)} ▷ Addition/deletion of boundary type a_o at position i
11:         end for
12:         for s_o ∈ S_o do
13:             S_e ← S_e ∪ {(s_o, i)}      ▷ Substitution of boundary type s_o with position i
14:         end for
15:     end for
16:     E ← A_e ∪ S_e ∪ T_e                      ▷ Create a full set of edits for convenience
17:     return E, A_e, S_e, T_e
18: end function
```

Algorithm 4.2 optional_set_edits

```
1: function OPTIONAL_SET_EDITS(bs_a, bs_b)
2:     O_S ← dict()                                  ▷ Create associative array
3:     for i = 1 → |bs_a| do
4:         a_i, b_i ← bs_a[i], bs_b[i]
5:         a , b, d ← a_i \ b_i, b_i \ a_i, a_i Δ b_i   ▷ Record additions/deletions
6:         if |d| > 0 then
7:             O_S[i] ← (d, a, b)
8:         end if
9:     end for
10:    return O_S
11: end function
```

Each position i in the boundary string is then iterated over with the latter point to be transposed represented as position j, until j reaches the end of the sequence. For each potential transposition found ($d \in t_p$), the associative array of recorded transpositions (O_T) is checked to see if it would overlap an already recorded transposition. A check is also performed to see whether a transposition would overlap two substitution operations in O_S. If either two substitutions or one or more transpositions would overlap, then the current transposition begin considered during iteration, then it is not recorded. If a transposition is added, its positions i and j for the boundary type stored in d (where t_p is a set of types) are recorded in O_T, and any additions/deletions or substitutions that overlap this transposition are removed from O_S. The time complexity of this function is approximately $\mathcal{O}(n_t \cdot p \cdot d)$, where n_t is the number of transposition spanning lengths considered, p is the number of positions in bs_a, and d is the total number of boundary types.

To check whether one transposition overlaps another, as in Algorithm 4.4, a search of the set of transpositions at the positions i and j in the associative array O_T is performed. If either position contains the same boundary type d as the transposition being considered, then we do not want to place a transposition at this location. An overlapping transposition is not placed so that one that is smaller in spanning distance or previous in boundary set string position will remain. The time complexity of this function is approximately $\mathcal{O}(d)$, where d is the total number of boundary types.

To determine whether a potential transposition at positions i and j overlaps two substitutions of the same boundary type d, Algorithm 4.5 is used. This function determines first if an untransposed boundary of type d exists at both positions and then whether both positions have the potential for substitution operations which could involve a boundary

Algorithm 4.3 transpositions

```
1: function TRANSPOSITIONS(bs_a, bs_b, N_t, O_S)
2:     O_T ← dict()                                   ▷ Create associative array
3:     T_e ← {}
4:     for n_i ∈ sorted(N_t) do
5:         n_i ← n_i − 1          ▷ Convert n_i from PBs spanning into PBs from a position
6:         for i = 1 → |bs_a| − n_i do
7:             j ← i + n_i
8:             a_i, a_j, b_i, b_j ← bs_a[i], bs_a[j], bs_b[i], bs_b[j]   ▷ Select edge sets
9:             d_i, d_j, d_a, d_b ← a_iΔb_i, a_jΔb_j a_iΔa_j b_iΔb_j     ▷ Symmetric differences
10:            t_p = d_i ∩ d_j ∩ d_a ∩ d_b                ▷ Detect potential transposition
11:            for d ∈ t_p do          ▷ Attempt to apply each by boundary type d
12:                t ← (i, j, d)                  ▷ Create transposition representation
13:                if ¬ OVERLAPS_EXISTING(i, j, d, O_T) ∧
14:                    ¬ HAS_SUBSTITUTIONS(i, j, d, O_S) then
15:                    T_e ← T_e ∪ {t}                          ▷ Record transposition
16:                    O_T[i] ← O_T[i] ∪ {t}                    ▷ Record positions covered
17:                    O_T[j] ← O_T[j] ∪ {t}
18:                    for k = 1 → 3 do
19:                        O_S[i][k] ← O_S[i][k] \ d     ▷ Remove set errors that overlap
20:                        O_S[j][k] ← O_S[j][k] \ d
21:                    end for
22:                end if
23:            end for
24:        end for
25:    end for
26:    return T_e
27: end function
```

of type d. The time complexity of this function is approximately $\mathcal{O}(d)$, where d is the total number of boundary types.

To determine the number of addition/deletion and substitution operations which can occur at a given position i for Algorithm 4.5, Algorithm 4.6 is used. This function quickly determines these values by taking the cardinality of the differences of one boundary set versus another at position i represented as a and b, and symmetrically as d. The time complexity of this function is approximately $\mathcal{O}(1)$. Later, Algorithm 4.7 performs this same task, but returns the actual sets themselves instead of their cardinalities. Algorithm 4.7 also makes an effort to produce a consistent pairing of specific boundaries from each segmentation to produce substitutions (detailed later).

Algorithm 4.4 overlaps_existing

```
1: function OVERLAPS_EXISTING(i, j, d, O_T)
2:     function OVERLAPS_EXISTING(p)
3:         for t ∈ O_T[p] do
4:             if t[3] = d then ▷ If the transposition (t) is of the same boundary type (d)
5:                 return True
6:             end if
7:         end for
8:         return False
9:     end function
10:    return OVERLAPS_EXISTING(i) ∨ OVERLAPS_EXISTING(j)
11: end function
```

Algorithm 4.5 has_substitutions

```
1: function HAS_SUBSTITUTIONS(i, j, d, O_S)
2:     if d ∈ O_S[i][1] ∧ d ∈ O_S[j][1] then     ▷ If boundary type d is at both positions
3:         d_i, a_i, b_i ← O_S[i]
4:         d_j, a_j, b_j ← O_S[j]
5:         if ADDITIONS_SUBSTITUTIONS(d_i, a_i, b_i)[2] > 0 ∧
6:             ADDITIONS_SUBSTITUTIONS(d_j, a_j, b_j)[2] > 0 then
7:             return True          ▷ If both positions contain potential substitutions
8:         end if
9:     end if
10:    return False
11: end function
```

Algorithm 4.7 is used to identify which of the remaining non-overlapping potential/optional set errors (O_S) are to be considered additions/deletions versus substitutions. This function iterates over all permutations of the sets of boundary difference tuples (d, a, b) at a position i to first identify substitutions. Those substitutions which pair two boundaries together from each segmentation with the smallest difference between them (assuming that the boundary types are ordinal—an unnecessary but useful property which is later used when creating comparison methods). The remaining boundaries not substituted are considered additions/deletions. The time complexity of this function is approximately $\mathcal{O}(\mathrm{P}(d, 2)^2)$, where d is the total number of boundary types and $\mathrm{P}(m, n)$ is the number of permutations of the set m choosing 2.

Boundary edit distance, as described, identifies the minimum number of additions/deletions, substitutions, and n-wise transpositions that can be used to edit a multiple-boundary-type boundary string into another. The algorithm presented is symmetric in

Algorithm 4.6 additions_substitutions

```
1: function ADDITIONS_SUBSTITUTIONS(d, a, b)
2:     add ← abs(|a| − |b|)
3:     sub ← (|d| − add)/2
4:     return add, sub
5: end function
```

Algorithm 4.7 additions_substitutions_sets

```
1: function ADDITIONS_SUBSTITUTIONS_SETS(d, a, b)
2:     O_sub, D_e ← (), ()
3:     δ ← −1
4:     for p_a ∈ permutations(a) do
5:         for p_b ∈ permutations(b) do
6:             δ_p ← 0
7:             for a_i, b_i ∈ p_a, p_b do
8:                 δ_p ← δ_p + abs(a_i − b_i)
9:                 if (δ_p < δ) ∨ (δ = −1) then
10:                    δ ← δ_p
11:                    O_sub ← (p_a, p_b)
12:                end if
13:            end for
14:        end for
15:    end for
16:    for p_a, p_b ∈ O_sub do
17:        D_e ← D_e ∪ (p_a) ∪ (p_b)
18:    end for
19:    S_e ← set(D_e)                ▷ Convert D_e from a sequence to a set
20:    A_e ← d \ S_e
21:    return A_e, S_e
22: end function
```

that it will produce the same cardinality of operations regardless of which boundary string is labelled as bs_a versus bs_b. The overall time complexity of the edit distance is approximately $\mathcal{O}(n_t \cdot p \cdot d)$, where n_t is the maximum transposition spanning length considered, p is the number of positions in bs_a, and d is the total number of boundary types. Most important, boundary edit distance distinguishes between additions/deletions, substitutions, and n-wise transpositions and interprets them in a consistent manner.

A formal proof of the minimality of this edit distance is not given. Instead, it is assumed that the strict order of how the edit operations are applied will guarantee

global minimality. First, the smallest transpositions are applied. Then, iteratively larger non-overlapping transpositions are applied. Transpositions are applied first because they can inherently convert two addition/deletion edits into one transposition. Transpositions are not added when they would overlap two substitutions because this could convert two potential substitutions into one transposition and two additions/deletions (two edits versus three).

Substitutions are added after transpositions as long as they do not overlap transpositions of the same boundary type. Substitutions are added before additions/deletions because one substitution replaces two potential additions/deletions. Finally, additions/deletions are added that do not overlap any other edit of the same boundary type. Additions/deletions are applied last because they represent the operation that is being minimized by converting them into transpositions and substitutions.

This order of operations maximizes the number of pairs of addition/deletion edits that are converted into single edit operations such as transpositions and substitutions, thereby producing a minimal number of edits. It accomplishes this using a greedy method of selection that bounds its search by the maximum distance that a transposition is allowed to span (n).

How does boundary edit distance act in various situations? How does boundary edit distance produce a minimal number of edits in situations where edits may overlap? What follows is a demonstration of how boundary edit distance handles various scenarios.

To illustrate boundary edit distance, the same segmentation task as described earlier is used where an excerpt from the poem in Figure 4.1 is being hypothetically segmented into a multiple-boundary-type boundary string representation. Returning to Figure 4.14 (reproduced below in Figure 4.18), boundary edit distance can be applied to identify the minimum number of types of edits that can be applied to edit one segmentation to appear as the other. In this example, boundary edit distance finds one substitution ($\{\mathrm{S}(1,2)\}$), two additions/deletions ($\{\mathrm{AD}(3), \mathrm{AD}(3)\}$), and one n-wise transposition ($\{$2-wise T(1)$\}$) for a total edit distance of 4. The boundary types involved in each edit are listed between the brackets accompanying the edit labels.

The previous example shows a combination of addition/deletion, substitution, and n-wise transposition edits. It shows that, when there are more boundaries at a position in one segmentation, a combination of both addition/deletion edits and substitution edits can result. It also illustrates a single transposition, and one additional misplaced boundary as an addition/deletion.

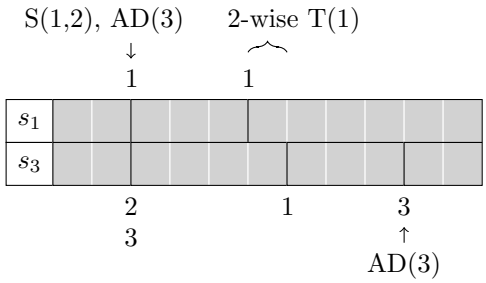


Figure 4.18: Two hypothetical multiple-boundary-type segmentations labelled with edits using boundary edit distance ($n_t = 2$)

Overlapping potential transpositions What if one encounters two potential transpositions that overlap upon the same position? This is illustrated in Figure 4.19, where the first potential transposition is chosen over the second potential transposition, and an addition/deletion results from the last boundary. This occurs because, in boundary edit distance, transpositions closest to the beginning of a segmentation are added first.

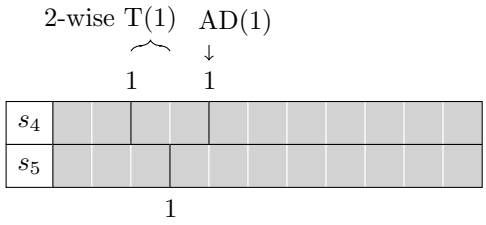


Figure 4.19: Two potential overlapping transpositions interpreted by boundary edit distance ($n_t = 2$) as one transposition and one addition/deletion

What if these two transpositions were of differing spanning distance (e.g., 2-wise versus 3-wise) and the largest one occurred first? In this example, n_t was increased to $n_t = 3$ to be able to consider both 2-wise and 3-wise transpositions. This situation is illustrated in Figure 4.20, where the first potential transposition spans further than the second, and despite the first occurring earlier, the second potential transposition is chosen, and an addition/deletion results from the first boundary. This occurs because, in boundary edit distance, transpositions of the smallest spanning size are added first.

What if the maximum transposition spanning distance (n_t) was too small to consider what is a visually-recognizable potential transposition? This is illustrated in Figure 4.21, where a potential transposition could occur if $n_t = 3$, but it is instead considered to be two addition/deletion edits because $n_t = 2$.

What if one was to analyse two potential transpositions which overlapped in position, but not in boundary type? This is illustrated in Figure 4.22, where two transpositions

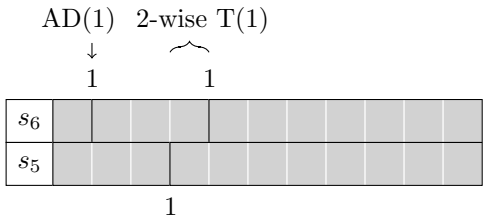


Figure 4.20: Two potential overlapping transpositions interpreted by boundary edit distance ($n_t = 3$) as one transposition and one addition/deletion

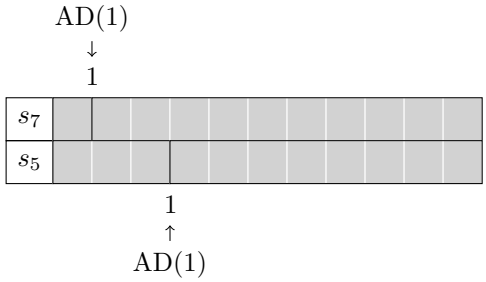


Figure 4.21: A potential transposition is interpreted by boundary edit distance ($n_t = 2$) as two addition/deletion edits

occur involving the same positions, but one is over boundaries of type 1 whereas the other is over boundaries of type 2. This occurs because, in boundary edit distance, transpositions are allowed to overlap if they do not transpose the same types of boundaries.

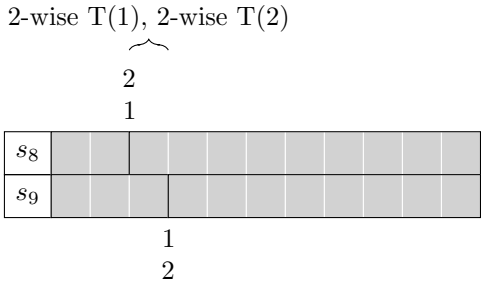


Figure 4.22: Two overlapping transpositions involving different boundary types as interpreted by boundary edit distance ($n_t = 2$)

Potential transpositions overlapping substitutions Taking the same basic scenario as before, what if one was to change the locations of the type 2 boundaries to instead be arranged across from the type 1 boundaries? This is illustrated in Figure 4.23, where there are two potential transpositions exactly as before, but to place either transposition would result in replacing two substitutions, which is prohibited in boundary edit

distance to maintain minimality. In this case, the choice is between two transpositions or two substitutions, and although the choice seems arbitrary, the next example will illustrate why substitutions are chosen over transpositions.

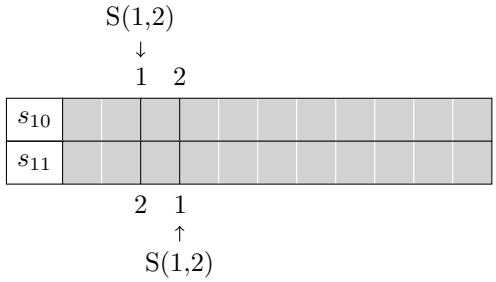


Figure 4.23: Two overlapping potential transpositions involving different boundary types interpreted by boundary edit distance ($n_t = 2$) as two substitutions

What if one was to analyse a potential transpositions which overlapped two substitutions involving three different types of boundaries? This is illustrated in Figure 4.24, where if one was to consider placing a transposition involving boundary type 1, two addition/deletion edits would result. Choosing to place a transposition instead of two substitutions would then result in 3 edits (one transposition and two additions/deletions), whereas it could instead be interpreted as two substitution operations. To preserve minimality, boundary edit distance prioritizes situations involving two substitutions over allowing a transposition to be placed.

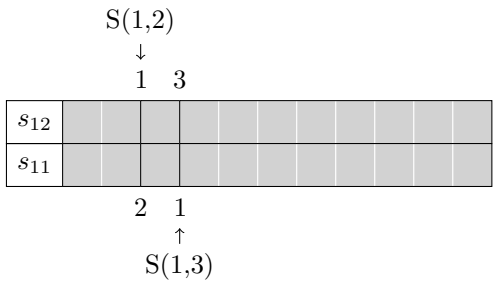


Figure 4.24: One potential transpositions and two addition/deletion edits interpreted by boundary edit distance ($n_t = 2$) as two substitutions

Boundary edit distance gives the ability to quantify the cardinality and types of the minimal number of edits between two boundary strings. This gives a consistent count of the edits (i.e., differences, or errors) between two segmentation. These counts, however, would be more intuitive if they were normalized so that they represented the proportion of segmentations that contain differences. A method of normalizing these edit counts

can be used to replace measures that lack consistency, such as WindowDiff and P_k. In the next section, a series of normalizations are proposed from which the best one will be chosen as a recommended segmentation comparison method.

4.1.3 Comparison Methods

When comparing two segmentations, one would ideally use a metric. A metric is a distance function which defines the distance between arbitrary elements from a set. In segmentation, these two elements are any two arbitrary segmentations of the same document D taken from the set of all possible segmentations of D. The set of all possible segmentations of D is the metric space in which our desired metric operates.

In Section 3.1, it was established that there exists a large number of segmentation comparison methods. Few of these methods, however, are able to differentiate between a fully missed boundary (a full miss) and a nearly missed boundary (a near miss), making them unable to award partial credit for near misses. Of those that can award partial credit for near misses, P_k and WindowDiff, neither is strictly considered metrics because they are not symmetric functions. For this reason, they are also not consistent, as demonstrated by Pevzner and Hearst (2002, pp. 11–18) when they analysed WindowDiff and P_k for their consistency (i.e., ability to produce the same result when presented with two different pairs of segmentations containing equivalent errors). Besides symmetricity, what are the requirements of a metric, and how can a better metric be created for segmentation comparison?

A segmentation metric is a function $g(x, y)$ that returns the distance between two arbitrary segmentations of a document D as a non-negative real number ($\mathbb{R}^+$). The following axioms must also be satisfied by $g(x, y)$ to be define a metric space (Rudin, 1976, pp. 30–36):

1. $g(x, y) = 0 \iff x = y$ (identity of indiscernibles)
2. $g(x, y) = g(y, x)$ (symmetry)
3. $g(x, y) + g(y, z) \geq g(x, z)$ (triangle inequality)

Defining a segmentation comparison method How can the axiomatic requirements of a metric be satisfied to create distance functions for segmentation? This work addresses this question by basing such functions upon the minimal *boundary edit distance* proposed

$$
\begin{aligned}
bs_x &= [\{\}, \{1\}, \{1\}] \\
bs_y &= [\{\}, \{2\}, \{1\}] \\
bs_z &= [\{1\}, \{2\}, \{\}] \\
g(a, b) &\equiv \text{boundary\_edit\_distance}(a, b) \\
g(x, y) + g(y, z) &\geq g(x, z) \qquad \textbf{Triangle inequality} \\
g(bs_x, bs_y) + g(bs_y, bs_z) &\geq g(bs_x, bs_z) \\
|\{\text{S(1,2)}\}| + |\{\text{3-wise T(1)}\}| &\geq |\{\text{2-wise T(1), AD(2), AD(1)}\}| \\
1 + 1 &\geq 3 \qquad \textbf{False}
\end{aligned}
$$

Figure 4.25: An example where the triangle inequality does not hold true for boundary edit distance using $n_t = 3$

earlier. Counts derived from this edit distance satisfy the first two axioms by design, because the edit distance:

- Only identifies edits if there are differences between two segmentations, and if two segmentations are identical then it does not (satisfying the identity of indiscernibles axiom); and
- Operations are defined so that they can be applied regardless of which segmentation is considered the first/second argument of the distance metric (satisfying the symmetry axiom).

The triangle inequality, unfortunately, does not always hold true for boundary edit distance, as shown in Figure 4.25. As a result, boundary edit distance cannot be used to create a true metric, it can still be used to create an informative comparison method for segmentation.

Boundary edit distance gives us an integer representation of the distance between two segmentations, but in NLP it is more common to compare proportions (i.e., values between zero and one). For this reason, it would be desirable to normalize boundary edit distance's values to produce a comparison method. As a percentage which represents how close one segmentation is to another, such a comparison method should return 0 when two segmentations are completely different and 1 when they are identical. How can boundary edit distance be used to create such a comparison method? Answers to this question are given later in this section, but first a variety of descriptive statistics are proposed which are later used to normalize boundary edit distance.

Value	*Codomain*	*Range*	*Description*
PBs	$\mathbb{N}_0$		Potential boundaries
$\text{fnc}(\text{T}_b)$	$\mathbb{N}_1$	$[1, \infty]$	Boundary types, where $\text{fnc}(x)$ could be $\min(x)$, $\max(x)$, $\lvert x\rvert$, etc.
$\text{fnc}(n)$	$\mathbb{N}_1$	$[1, \infty]$	Transposition spanning distances, where $\text{fnc}(x)$ could be $\min(x)$, $\max(x)$, $\lvert x\rvert$, etc.
$\lvert\text{E}\rvert$	$\mathbb{N}_0$	$[0, \text{pb}(D)]$	Edit operations (of all types)
$\lvert\text{A}_e\rvert$	$\mathbb{N}_0$	$[0, \text{pb}(D)]$	Addition/deletion edit operations
$\lvert\text{S}_e\rvert$	$\mathbb{N}_0$	$[0, \lfloor\frac{1}{2}\text{pb}(D)\rfloor]$	Substitution edit operations
$\lvert\text{T}_e\rvert$	$\mathbb{N}_0$	$[0, \lfloor\frac{1}{2}\text{pb}(D)\rfloor]$	Transposition edit operations
$\text{w}_\text{a}(\text{A}_e)$	$\mathbb{Q}_+$	$[0, \text{pb}(D)]$	Weighted addition/deletion edit operations
$\text{w}_\text{s}(\text{S}_e, \text{T}_b)$	$\mathbb{Q}_+$	$[0, \lfloor\frac{1}{2}\text{pb}(D)\rfloor]$	Weighted substitution edit operations
$\text{w}_\text{t}(\text{T}_e, n_t)$	$\mathbb{Q}_+$	$[0, \lfloor\frac{1}{2}\text{pb}(D)\rfloor]$	Weighted transposition edit operations
$\lvert\text{B}_M\rvert$	$\mathbb{N}_0$	$[0, \text{pb}(D)]$	Matching boundaries
$\lvert\text{B}_U\rvert$	$\mathbb{N}_0$	$[0, \text{pb}(D)]$	Un-matching boundaries (i.e., position i has a boundary in s_1 but not in s_2)
$\lvert\text{B}_1\rvert$	$\mathbb{N}_0$	$[0, \text{pb}(D)]$	Boundaries placed in s_1
$\lvert\text{B}_2\rvert$	$\mathbb{N}_0$	$[0, \text{pb}(D)]$	Boundaries placed in s_2
$\lvert\text{B}\rvert$	$\mathbb{N}_0$	$[0, 2 \cdot \text{pb}(D)]$	Boundaries placed in s_1 and s_2 (i.e., $\lvert\text{B}\rvert = \lvert\text{B}_1\rvert + \lvert\text{B}_2\rvert$)

Table 4.1: Segmentation statistics, codomains, and their codomain ranges that can be determined using either boundary edit distance or another simple counting function upon two segmentations (e.g., s_1 and s_2) of a document D

Which descriptive statistics can be used to define a comparison method? Using boundary edit distance, the number of near misses and full misses can be unambiguously quantified as addition/deletion or transposition edit operations, respectively.[5] Besides these counts, what other descriptive segmentation statistics can be of use? What numerator and denominator should be used to create a ratio that represents how close one segmentation is to another? To answer these question, first, a set of the available descriptive statistics for segmentations that could be of use is shown in Table 4.1 as proposed and organized herein.

The majority of the values in Table 4.1 are either familiar, or their calculation is trivial, except for $\text{w}_\text{a}(\text{A}_e)$, $\text{w}_\text{s}(\text{S}_e, n_t)$, and $\text{w}_\text{t}(\text{T}_e, \text{T}_b)$. These values are weighted counts of edit operation types which allow for one to discount error represented by edits in specific scenarios (described later).

[5]Substitution edit operations represent another form of near miss that occurs not in position but in boundary type, referred to as conversion error.

Scaling edit operation severity through weighting Boundary edit distance returns the total set of edits E, but it also returns these edits as three separate sets of edits by type: A_e, S_e, T_e, or additions/deletions, substitutions, and transpositions. The severity of the interpretation of these edits as errors can be adjusted by scaling them by some function to reduce their severity, and a few such functions are provided herein. Each function has, as a parameter, (at least) the set of operations that they weight, and is expected to return a real number v between $0 \leq v < \mathrm{pb}(D)$, while satisfying the condition that $\mathrm{sum}(\mathrm{w_a}(\mathrm{A}_e), \mathrm{w_s}(\mathrm{S}_e, \mathrm{T}_b), \mathrm{w_t}(\mathrm{T}_e, n_t)) \leq |E| \leq \mathrm{pb}(D)$. If one were to not weight the counts at all, then the functions shown in Equations 4.17–4.19 could be used.

$$\mathrm{w_a\_unweighted}(\mathrm{A}_e) = |\mathrm{A}_e| \tag{4.17}$$

$$\mathrm{w_s\_unweighted}(\mathrm{S}_e, \mathrm{T}_b) = |\mathrm{S}_e| \tag{4.18}$$

$$\mathrm{w_t\_unweighted}(\mathrm{T}_e, n_t) = |\mathrm{T}_e| \tag{4.19}$$

If one desired to scale some errors such that they are not penalized as harshly as others, the cardinality of the sets could be scaled by a value, say (arbitrarily) 0.5 for transpositions so that they are penalized even less harshly, as in Equations 4.20–4.22.

$$\mathrm{w_a\_scalar}(\mathrm{A}_e) = |\mathrm{A}_e| \times 1.0 \tag{4.20}$$

$$\mathrm{w_s\_scalar}(\mathrm{S}_e, \mathrm{T}_b) = |\mathrm{S}_e| \times 1.0 \tag{4.21}$$

$$\mathrm{w_t\_scalar}(\mathrm{T}_e, n_t) = |\mathrm{T}_e| \times 0.5 \tag{4.22}$$

Edit operation distance scaling One can create more complicated weight functions which utilize the information stored within each set. In each set, and for each edit operation, both the position information and the boundary types involved in each edit are recorded. This information allows one to create distance functions between individual edit operations of the same type which could be used to intuitively scale errors to better reflect their severities for a segmentation task.

Substitutions are stored in the set S_e in the form (a, b, i), where a and b are both boundary types and i the position. If the numerical values used to represent each boundary type in T_b are ordinal (by the definition of the segmentation task), then one can create a distance function between boundary types which can be used to scale the value of the penalty represented by a substitution edit operation. Arguably, for ordinal boundaries, the greater the numerical distance between two boundaries, then the greater severity of the error. Using this assumption, an example substitution edit operation weighting function

is shown in Equation 4.23 which penalizes each substitution operation with $w_s < v \leq 2$ edits of error (where $0 \leq w_s \leq 1$). Substitutions could be penalized up to 2 times each and still be normalized properly because each substitution could be decomposed into two additions/deletions. This function uses the difference between the numerical values of ordered boundary types that are substituted over the maximum difference possible for the set of boundary types T_b as a distance function.

$$\mathrm{w_s\_ordinal}(\mathrm{S}_e, \mathrm{T}_b) = \sum_{j=1}^{|\mathrm{S}_e|} \left(w_s + \frac{\mathrm{abs}(\mathrm{S}_e[j][1] - \mathrm{S}_e[j][2])}{\max(\mathrm{T}_b) - \min(\mathrm{T}_b)} \right) \tag{4.23}$$

Transposition weight functions can also be created which utilize the information contained within the set of transpositions T_e. Transpositions are stored in the set T_e in the form (i, j, d), where i and j are the positions of the start and end of the transposition, and d is the boundary type transposed. One can use the spanning distance of a transposition as a distance function with which to scale transposition errors. Arguably, for segmentation, a transposition spanning a shorter distance could be awarded more credit than a transposition spanning a greater distance, because shorter spanning transpositions are assumed to be more probable error interpretations than greater-spanning transpositions (i.e., it is assumed that a qualitative judgement would interpret a boundary being off by one unit to be more preferable than a boundary being off by two units). Using this assumption, an example transposition edit operation weighting function is shown in Equation 4.24. Transpositions could be penalized up to 2 times each and still be normalized properly because each transposition could be decomposed into two additions/deletions. This function uses the difference between transposition spanning distances over the maximum difference possible given the maximum specified spanning distance in n_t as a distance function.

$$\mathrm{w_t\_span}(\mathrm{T}_e, n_t) = \sum_{j=1}^{|\mathrm{T}_e|} \left(w_t + \frac{\mathrm{abs}(\mathrm{T}_e[j][1] - \mathrm{T}_e[j][2])}{\max(n_t) - 1} \right) \tag{4.24}$$

Modelling error as a ratio Using edits to model the sources of error that are apparent when comparing two segmentations, a penalty function can be defined. This work defines a penalty function that can then be used as the numerator in a ratio subtracted from 1 as in Equation 4.25 to become a reward function (for stylistic reasons). This percentage—where a value of 1 represents complete similarity and 0 complete dissimilarity—is the general

form of each comparison method proposed herein, where each comparison method defines a specific penalty(bs_1, bs_2, n_t) and max_penalties(bs_1, bs_2, D) function.

$$\text{reward}(bs_1, bs_2) = 1 - \frac{\text{penalty}(bs_1, bs_2, n_t)}{\text{max\_penalties}(bs_1, bs_2, D)} \tag{4.25}$$

From this form, the choice of how to formulate the numerator and denominator (i.e., penalty and maximum penalties functions) is left to the individual comparison methods proposed herein. To choose a numerator and denominator, some questions must be asked: how should error be modelled (as a mismatch between boundaries or internal segments), and what units should be used? Each comparison method proposed herein answers these questions differently, leaving the selection of the most appropriate strategy to experimentation (see Chapter 5).

What should error be modelled by: boundaries or segments? Given the descriptive statistics in Table 4.1, there exists a lot of choice in how a similarity comparison method could be defined. There are, however, two prominent conceptualizations of what could constitute error for a task that can dictate our designs; specifically, is error a product of:

- *Misplaced boundaries*; i.e., out of the boundaries placed, which boundaries do not match up, and to what degree?
- *Mismatched segments*; i.e., each PB is a member of a segment within an overall segmentation; out of all PBs, which PBs do not fall within the same segment, and to what degree?

There may exist a comparison method which is an ideal indicator for both error conceptualizations, but if there is not, then two comparison methods are required. For both conceptualizations, comparison methods are proposed and later evaluated against each other and WD and P_k to determine which is most suitable for comparing two arbitrary segmentations.

Edits as penalties To identify the most intuitive normalization strategies available for the development of comparison methods, it is helpful to identify what the effective penalty would be for each strategy. There are two obvious strategies: normalization by boundaries or potential boundaries. Assuming that for each edit a penalty of 1 (B or PB)

		Involved		*Penalties/x*	
Edit	Penalty	Bs	PBs	$x = \text{Bs}$	$x = \text{PBs}$
A_e	1	1	2	$^1/_1$	$^1/_2$
S_e	1	2	2	$^1/_2$	$^1/_2$
T_e	1	2	2	$^1/_2$	$^1/_2$
B_M	0	2	2	$^0/_2$	$^0/_2$
$\text{w}_\text{s}\_\text{ordinal}(\text{S}_e, \text{T}_b)$	$[0, 1]$	2	2	$^{[0,1]}/_2$	$^{[0,1]}/_2$
$\text{w}_\text{t}\_\text{span}(\text{T}_e, n_t)$	$[0, 1]$	2	2	$^{[0,1]}/_2$	$^{[0,1]}/_2$

Table 4.2: Penalties awarded per edit and per boundaries or potential boundaries involved when comparing segmentations (e.g., s_1 and s_2) of a document D

is awarded (except for matches, B_M, which are not penalized), the penalties over the unit involved can be seen in Table 4.2.

Table 4.2 shows us that when normalizing using Bs, a full miss (A_e) is penalized more severely than a conversion error or a near miss (S_e and T_e, respectively) by a factor of 2; this is an ideal scenario because it discounts errors that are not full misses. When normalizing by PBs, however, each is penalized at an equal rate; this is not ideal, and a comparison method devised that uses PBs as units should accommodate this issue in some manner. For all comparison methods proposed, however, each uses the penalty function shown in Equation 4.26 unless otherwise specified, and each comparison method then proposes its own denominators in the form of $\text{max\_penalties}(bs_1, bs_2, D)$ functions.

$$\text{penalty}(bs_1, bs_2, n_t) = \text{w}_\text{a}(\text{A}_e) + \text{w}_\text{s}\_\text{ordinal}(\text{S}_e, \text{T}_b) + \text{w}_\text{t}\_\text{span}(\text{T}_e, n_t) \tag{4.26}$$

Mismatched boundaries If the source of error in our segmentation task is best conceptualized as a mismatch between boundary placement in one segmentations versus another, then the choice of a denominator should reflect the number of boundary choices made. Looking back to Table 4.1, we have a variety of values that represent the Bs in one or both segmentations (e.g., $|B_M|$, $|B_U|$, $|B_1|$, $|B_2|$, $|B|$), and how they related to each other. These can be used to construct appropriate penalty and maximum penalty functions which once used in the general form shown in Equation 4.25 can express similarity between the Bs placed in two segmentations as a percentage. Three new comparison methods that model error as mismatched boundaries are proposed herein:

- Boundary Similarity variant A (B_a);
- Boundary Similarity variant B (B_b); and
- Boundary Similarity variant C (B_c).

These comparison methods are also accompanied by a proposal for a new confusion matrix for segmentation multi-class boundary classification based upon boundary edit distance.

Boundary Similarity—A In this comparison method, named *boundary similarity* (variant A), the units used are boundaries (Bs). As a denominator, it essentially uses the total number of Bs placed, $|B|$, as a maximum penalty function (Equation 4.27). Keeping the penalty function described in Equation 4.26, this comparison method would report a similarity of 1 if and only if there are no edits made, and would report a similarity of 0 if and only if $|B| = \text{penalty}(bs_1, bs_2, n_t)$.

$$\text{max\_penalties}_{\text{BS}_a}(bs_1, bs_2, D) = \begin{cases} |\text{B}| & \text{when } |\text{B}| > 0 \\ 1 & \text{when } |\text{B}| = 0 \end{cases} \tag{4.27}$$

Using the functions described, *boundary similarity* variant A is defined as Equation 4.28 when $|B| > 0$. This comparison method has the advantage of being simple in that it only penalizes edits that occur out of the set of all Bs placed.

$$\text{B}_a(bs_1, bs_2, n_t) = 1 - \frac{\text{w}_\text{a}(\text{A}_e) + \text{w}_\text{s}\text{\_ordinal}(\text{S}_e, \text{T}_b) + \text{w}_\text{t}\text{\_span}(\text{T}_e, n_t)}{|\text{B}|} \tag{4.28}$$

Boundary Similarity—B In this comparison method, named *boundary similarity* variant B, the units used are boundaries (Bs) mapped as boundary pairs. As a denominator, it essentially uses the total number of Bs that have been edited (and unweighted) plus those that are matching as a maximum penalty function (Equation 4.29). This allows for penalties to be reduced by the presence of matching Bs, and adds greater emphasis upon the weighting used in the penalty function to reduce the severity of errors in some situations (scaling by substitution ordinal distance or transposition spanning distance), which should provide a very nuanced reflection of segmentation error. Keeping the penalty function described in Equation 4.26, this comparison method would report a similarity of 1 if and only if there are no edits made and would report a similarity of 0 if and only if $|E| = \text{penalty}(bs_1, bs_2, n_t)$ and $|\text{B}_M| = 0$.

$$\text{max\_penalties}_{\text{BS}_b}(bs_1, bs_2, D) = \begin{cases} |\text{A}_e| + |\text{S}_e| + |\text{T}_e| + |\text{B}_M| & \text{when } |\text{B}| > 0 \\ 1 & \text{when } |\text{B}| = 0 \end{cases} \tag{4.29}$$

Pair	Correctness
Match	1
Addition/deletion	0
Transposition	$1 - \mathrm{w_t\_span}(\mathrm{T}_e, n_t)$
Substitution	$1 - \mathrm{w_s\_ordinal}(\mathrm{S}_e, \mathrm{T}_b)$

Table 4.3: Correctness of each type of pair (i.e., match or edit) for $\mathrm{B_b}$

Using these functions, *boundary similarity* (variant B) is defined as Equation 4.30 when segmentations contain boundaries. This comparison method has the advantage of increasing the credit given to matches and emphasizing the weighting used to scale the severity of each error type.

$$\mathrm{B}_b(bs_1, bs_2, n_t) = 1 - \frac{\mathrm{w_a}(\mathrm{A}_e) + \mathrm{w_s\_ordinal}(\mathrm{S}_e, \mathrm{T}_b) + \mathrm{w_t\_span}(\mathrm{T}_e, n_t)}{|\mathrm{A}_e| + |\mathrm{S}_e| + |\mathrm{T}_e| + |\mathrm{B}_M|} \tag{4.30}$$

$\mathrm{B_b}$ has the interesting property that it is essentially the sum of one minus each edit plus the number of matches over the total number of edits and matches (Equation 4.31). Because each edit and each match represents a pair of boundaries—where a boundary is accompanied either by another boundary (match), a missing boundary (AD), a closely-off boundary (T), or a boundary of the wrong type (S)–$\mathrm{B_b}$ is essentially the sum of the correctness of each pair over the total number of pairs. This correctness defined as one minus the scaled penalty for each edit, is shown in Table 4.3, This is a powerful intuition which is later utilized to enable per-boundary-pair statistical hypothesis testing (See Section 4.3).

$$\mathrm{B}_b(bs_1, bs_2, n_t) = \frac{|\mathrm{S}_e| - \mathrm{w_s\_ordinal}(\mathrm{S}_e, \mathrm{T}_b) + |\mathrm{T}_e| - \mathrm{w_t\_span}(\mathrm{T}_e, n_t)}{|\mathrm{A}_e| + |\mathrm{S}_e| + |\mathrm{T}_e| + |\mathrm{B}_M|} \tag{4.31}$$

Boundary Similarity—C In this comparison method, named *boundary similarity* variant C, the units used are potential boundaries (PBs). As a denominator, it essentially uses the total number of potential boundaries involved in an edit or match, which is calculated as the sum of all boundaries plus any boundaries that do not have a matching boundary in the opposite segmentation (Equation 4.33) to represent the PBs which, if they contained Bs, would become matches. This allows for penalties to be measured over all potential boundaries involved, but not over every single potential boundary. Using PBs requires a modification to the penalty function so that addition/deletion operations

are penalized at a rate of $^1/_1$ units per PB, as opposed to $^1/_2$ (detailed in the discussion of Table 4.2). For this reason, Equation 4.26 is modified to count addition/deletion edits as a penalty twice, as in Equation 4.32. Using the functions described, this comparison method would report a similarity of 1 if and only if there are no edits made, and would report a similarity of 0 if and only if $|\mathrm{B}| + |\mathrm{B}_U| = \mathrm{penalty}_{\mathrm{B}_c}(bs_1, bs_2, n_t)$ (where B_U is un-matching boundaries; see Table 4.1).

$$\begin{aligned}\mathrm{penalty}_{\mathrm{B}_c}(bs_1, bs_2, n_t) &= \mathrm{penalty}(bs_1, bs_2, n_t) + \mathrm{w_a}(\mathrm{A}_e) \\ &= 2 \cdot \mathrm{w_a}(\mathrm{A}_e) + \mathrm{w_s\_ordinal}(\mathrm{S}_e, \mathrm{T}_b) + \mathrm{w_t\_span}(\mathrm{T}_e, n_t)\end{aligned} \tag{4.32}$$

$$\mathrm{max\_penalties}_{\mathrm{BS}_c}(bs_1, bs_2, D) = \begin{cases} |\mathrm{B}| + |\mathrm{B}_M| & \text{when } |\mathrm{B}| > 0 \\ 1 & \text{when } |\mathrm{B}| = 0 \end{cases} \tag{4.33}$$

Using these functions, *boundary similarity* (variant C) is defined as Equation 4.34 when $|B| > 0$. This comparison method has the advantage of measuring error over all potential boundaries involved, but not over every single potential boundary.

$$\mathrm{BS}_c(bs_1, bs_2, n_t) = 1 - \frac{2 \cdot \mathrm{w_a}(\mathrm{A}_e) + \mathrm{w_s\_ordinal}(\mathrm{S}_e, \mathrm{T}_b) + \mathrm{w_t\_span}(\mathrm{T}_e, n_t)}{|\mathrm{B}| + |\mathrm{B}_U|} \tag{4.34}$$

Mismatched segments If the source of error in our segmentation task is best conceptualized as a mismatch between segments in one segmentations versus another, then the units used to measure error would ideally be expressed in potential boundaries. Ideally, one would be able to map segments from one segmentation into another, and identify the number of PBs that are not contained within the appropriate segment. This mapping, however, is not often possible because the number of internal segments between two segmentations often differs, and the definition of what is a sub segment (in a hierarchical sense) versus a truncated and adjacent segment is not clear. The best that can be done then is to analyse boundaries that have been placed out of all potential boundary positions, and to determine how each boundary position differs between the two segmentations compared. *One new comparison method that models error as a mismatch between segments is proposed herein: Segmentation Similarity.*

Segmentation Similarity For the purpose of conceptualizing error in terms of mismatched segments, a comparison method named *segmentation similarity* (S), first defined in Fournier and Inkpen (2012), has been reformulated herein to fit the general form

expressed in Equation 4.25 and to utilize the refined version of boundary edit distance proposed herein. The units used are potential boundaries (PBs). As a denominator, it uses the total number of PBs as a maximum penalty function (Equation 4.35). This simple normalization allows for it to react to linear increases in error in the same fashion; it is a predicable and simple comparison method. Keeping the penalty function described in Equation 4.26, this comparison method would report a similarity of 1 if and only if there are no edits made, and would report a similarity of 0 if and only if $\text{pb}(D) = \text{penalty}(bs_1, bs_2, n_t)$.

$$\text{max\_penalties}_{\text{S}}(bs_1, bs_2, D) = \text{pb}(D) \tag{4.35}$$

Using these functions, *segmentation similarity* (S) is defined as Equation 4.36. This comparison method has the advantage of reacting to increases in error linearly, as will be shown in Chapter 5.

$$\text{S}(bs_a, bs_b, n_t) = 1 - \frac{\text{w}_\text{a}(\text{A}_e) + \text{w}_\text{s}\text{\_ordinal}(\text{S}_e, \text{T}_b) + \text{w}_\text{t}\text{\_span}(\text{T}_e, n_t)}{\text{pb}(D)} \tag{4.36}$$

Segmentation task errors vary Not all segmentation errors have the same severity from task to task. In some segmentation tasks, a near miss might need to always be considered as two full misses, or a near miss might be permissible that spans up to 5 units. Perhaps a near miss should not be penalized at all, or neither should substitutions, or perhaps they should be penalized less. Segmentation similarity can be customized to suit these scenarios, and more, to be able to adapt to the specific segmentation task being analysed and to appropriately model the relative severity of various segmentation errors for a task.

The comparison methods presented all have one customizable parameter: n_t, or the maximum spanning distance allowed by a transposition. If, for a segmentation task, no near misses are to be allowed, then n_t be can set to 1, which ensures that no transposition edits are considered. Alternatively, if near misses should be considered, and for a task a segmenter should be afforded partial credit for boundaries of by up to 3 units, then one can set $n_t = 4$, allowing us to consider transpositions spanning up to 4 PBs (or encompassing 3 units). By default, $n_t = 2$, so that only adjacent near misses (i.e., 2-wise transpositions) are awarded partial credit, but this number can be set on a task by task basis if sufficient rationale is given (an example of which is shown later in Section 5.2).

A boundary-edit-distance based confusion matrix for segmentation Scaiano and Inkpen (2012) proposed formulating segmentation as a classification problem, and proposed using WindowDiff to populate a confusion matrix. From this two-class confusion matrix, any number of IR metrics could be calculated, e.g., precision, recall, F_β-measure, etc. The issue with this approach is that it utilized WindowDiff, thus inheriting its lack of intuition and some of its flaws. It was also limited to producing a two-class confusion matrix, meaning that it is only suitable for single-boundary-type segmentation tasks.

This work proposes using boundary edit distance to create a multi-class confusion matrix for multi-boundary-type segmentations. In such a matrix, the set of classes is defined as the set of boundary types plus a non-boundary class (i.e., $\varnothing$). This non-boundary class models when a boundary is contained within a manual segmentation but not an automatic segmentation. In the definition contained herein, the confusion matrix uses scaled counts of transposition and substitution edits to represent partial matches (shown later).

To populate such a matrix, $\mathrm{CM}(a, p)$ can be used (see Equation 4.37) where a and p are boundary types within the set T_b. When a and p are the same type, $\mathrm{CM}(a, p)$ is the set of matching boundaries of that type ($|\mathrm{B}_{M,a}|$) summed with the scaled number of edits in the set of substitutions and transpositions where the manual segmentation contains type a and the automatic segmentation contains type p. When a and p are not the same type, $\mathrm{CM}(a, p)$ is the scaled number of edits in the set of substitutions and transpositions where the manual segmentation contains type a and the automatic segmentation contains type p. When a is a boundary but p is not then this is represented by $|\mathrm{A}_{e,a}|$ (the set of A/D edits from boundary edit distance), which is the number of full misses where the sole boundary in the pair is found in the manual segmentation (e.g., a FN). When p is a boundary but a is not then this is represented by $|\mathrm{A}_{e,p}|$, which is the number of full misses where the sole boundary in the pair is found in the automatic segmentation (e.g., a FP).

$$\mathrm{CM}(a,p) = \begin{cases} |\mathrm{B}_{M,a}| + \mathrm{w_s\_ordinal}(\mathrm{S}_e^{a,p}, \mathrm{T}_b) + \mathrm{w_t\_span}(\mathrm{T}_e^{a,p}, n_t) & \text{where } a = p \\ \mathrm{w_s\_ordinal}(\mathrm{S}_e^{a,p}, \mathrm{T}_b) + \mathrm{w_t\_span}(\mathrm{T}_e^{a,p}, n_t) & \text{where } a \neq p \\ |\mathrm{A}_{e,a}| & \text{where } p = \varnothing \\ |\mathrm{A}_{e,p}| & \text{where } a = \varnothing \end{cases} \tag{4.37}$$

Using Equation 4.37 in a situation where there is only one boundary type, a confusion matrix would be populated as shown in Tables 4.4a–4.4b.

Predicted \ *Actual*	B	Non-B
B	$\text{CM}(1,1)$	$\text{CM}(\varnothing,1)$
Non-B	$\text{CM}(1,\varnothing)$	$\text{CM}(\varnothing,\varnothing)$

(a) One boundary type ($\text{T}_b = \{1\}$)

Predicted \ *Actual*	1	2	Non-B
1	$\text{CM}(1,1)$	$\text{CM}(2,1)$	$\text{CM}(\varnothing,1)$
2	$\text{CM}(1,2)$	$\text{CM}(2,2)$	$\text{CM}(\varnothing,2)$
Non-B	$\text{CM}(1,\varnothing)$	$\text{CM}(2,\varnothing)$	$\text{CM}(\varnothing,\varnothing)$

(b) Two boundary types ($\text{T}_b = \{1,2\}$)

Table 4.4: Boundary edit distance based confusion matrices for differing numbers of boundary types segmentation

This work refers to IR metrics that use this new confusion matrix proposed herein as B-precision, B-recall, BF_β-measures, etc. Using this confusion matrix, these IR metrics are able to award partial credit to near misses through the scaling of boundary matches. For multiple boundary type segmentations, this confusion matrix also allows IR metrics to award partial credit for substituted boundaries in the same manner as transpositions.

B-precision, B-recall, and B-F_β-measure are not evaluated directly against WindowDiff, S, B_a, B_b, and B_c because they are all directly related to B_b. B_b measures the correctness of each pair, whereas this confusion matrix also breaks down the correctness of each pair by class. This work proposes that one of the four segmentation comparison methods proposed should be used to compare segmentations to each other in terms of the similarity of their boundary placements. Then, after such an evaluation, B-precision or B-recall should be used to answer how a segmenter did better or worse. Alternatively, if a segmentation task favours boundary precision over recall, then B-IR metrics should be used, however, this is not often the case because the end product often desired is a segmentation, not a set of boundaries. For this reason, most segmentation tasks require an evaluation that emphasizes similarity as opposed to precision or recall.

Summary of the comparison methods proposed The definitions of these new comparison methods use boundary edit distance to model the differences between two segmentations (i.e., their error) out of how each defines the maximum amount of observable error. Each is able to award partial credit to near misses by considering them as individual transposition edits as opposed to two addition/deletion edits each, or one addition/deletion and a substitution edit each. Unlike their counterparts (e.g., WindowDiff and P_k), they also allow for the analysis of segmentations containing multiple, mutually inclusive, and potentially ordinal boundary types.

To summarize, the comparison methods described and defined herein quantify error using boundary edit distance to handle near misses. They also have the advantage over

previous comparison methods (e.g. WindowDiff, P_k) that they:

- Are consistent[6];
- Handle multiple boundary types and mutually inclusive boundary placement, unlike all other existing comparison methods;
- Award partial credit for near-misses, and are customizable to be able to support a wide range of segmentation tasks for which different errors may have different degrees of severity; and
- Intuitively conceptualize error using edit operations that directly manipulate boundaries, as opposed to using an interpretation that must first utilize an intermediate unit such as windows which contain and operate upon overlapping regions.

Overall, the comparison methods proposed herein offer consistency, customizability, and the intuition one expects of comparison methods for usage in comparing two segmentations and in evaluating how close an automatic segmenter is to a human-produced (i.e., manual) segmentation.

4.2 Inter-Coder Agreement

To evaluate the performance of an automatic segmenter, the output of such a segmenter is often compared to segmentations produced by humans. These human (i.e., manual) segmentations are collected by creating and applying:

- Coding schemes to label hypothesized types of segments and boundaries; and
- A set of instructions to guide manual segmenters (i.e., coders) during their task.

To determine whether some manual segmentations collected can be relied upon, we must analyse the reliability of the coding scheme and instructions used to collect them. To accomplish this task, multiple coders segment the same documents, and then an inter-coder agreement coefficient is calculated to identify the reliability of the coding scheme that has been applied.

[6]Importantly, they does not succumb to the failings of window-based methods such as WindowDiff, P_k which vary due to internal segment sizes and improper counting of errors at the beginning and ending of segmentations and issues with calculating their window size k.

There are a variety of existing inter-coder agreement coefficients, as detailed in Section 3.2. All of those coefficients can be applied to segmentation to evaluate, per potential boundary, whether a set of coders reliably applied a coding scheme. These coefficients, however, judge agreement upon a strict boundary presence or absence interpretation. Many studies report low inter-coder agreement coefficients because coders often approximately agree upon where segments lie, but disagree upon the exact placement of boundaries separating them (Artstein and Poesio, 2008, p. 40).

To compute inter-coder agreement coefficients, first the actual agreement amongst coders is calculated. Actual agreement is a pairwise mean that indicates how well coders agreed with each other. Not all sets of labels are as easy to apply as others, however. To judge how well the coders applied a set of labels in the presence of chance agreement, actual agreement must be adjusted by expected agreement. How expected agreement is calculated depends upon how random agreement should be modelled for a task, but its function is to correct actual agreement to reflect whether agreement is primarily due to chance or not.

Current inter-coder agreement coefficients use a formulation of actual agreement that produces low values because it does not account for near misses. For this reason, it is desirable to have a method of quantifying inter-coder agreement that is able to award partial credit to near misses between two coders. This work proposes using one of the four normalizations of boundary edit distance proposed earlier to make actual agreement for inter-coder agreement coefficients award partial credit for near misses.

In this section, the boundary edit distance based comparison methods defined earlier (Chapter 4) are used to adapt a variety of inter-coder agreement coefficients to be able to award partial credit for near misses between coders. Upon the recommendation of Artstein and Poesio (2008, pp. 57–58) and Craggs and McGee Wood (2005), the coefficient reported herein will be Scott's π and Fleiss' multi-π (Scott, 1955; Fleiss, 1971)–referred to as π and π^* respectively. This coefficient was chosen for its ease of adaptation and its resiliency towards category and individual-coder bias so that a generalized view of how well a coding scheme was applied can be obtained. Cohen's κ and multi-κ (Cohen, 1960; Davies and Fleiss, 1982)–referred to as κ and κ^* respectively—are also adapted. κ is not recommended for usage in evaluating a coding scheme due to its sensitivity to individual coder bias, but it is adapted herein so that Artstein and Poesio's Bias (Artstein and Poesio, 2008, p. 146) can be calculated (discussed later).

To discuss inter-coder agreement upon segmentation, a modified form of the notation used by Artstein and Poesio (2008) is used. This work also bases its proposals on the

inter-coder agreement coefficient formulations provided by Artstein and Poesio (2008) and reproduces them herein to illustrate the changes made to them. In the notation used for this chapter, the set of:

- *Items* (i.e., boundary positions) coded is $\{i|i \in I\}$ with cardinality $\mathbf{i}$;
- *Documents* of which segmentations are produced is $\{d|d \in D\}$ with cardinality $\mathbf{d}$;
- *Categories* is $\{k|k \in K\}$ with cardinality $\mathbf{k}$;
- *Coders* is $\{c|c \in C\}$ with cardinality $\mathbf{c}$;
- *Segmentations* of an item i by a coder c is $\{s|s \in S\}$, where when $s_{i,c}$ is specified with only one subscript, it denotes s_c, for all relevant items (i); and
- *Types* of segmentation boundaries is $\{t|t \in \mathrm{T}\}$ with cardinality $\mathbf{t}$.

The four coefficients (π and κ in both their two-coder and multi-coder forms) adapted herein range from $[{}^{A_e}/_{1-A_e}, 1]$, where 0 indicates that their agreement is primarily due to chance, and 1 indicates that their agreement is not. All four coefficients have the general form shown in Equation 4.38, where they are the calculation of actual agreement (A_a) minus expected (i.e., chance) agreement (A_e) over one minus expected agreement. For each coefficient, it is the formulation and calculation of expected agreement that differs (A_e), whereas actual agreement (A_a) remains the same.

$$\kappa, \pi, \kappa^*, \text{ and } \pi^* = \frac{\mathrm{A}_a - \mathrm{A}_e}{1 - \mathrm{A}_e} \tag{4.38}$$

Actual agreement for segmentation Actual agreement (or percentage of agreement, observed agreement) is "the percentage of judgements on which the two analysts agree when coding the same data independently" (Scott, 1955, p. 323). For segmentation, this is the proportion of boundary positions (i.e., items in the set I) upon which two or more coders agree. Using formulations provided in Artstein and Poesio (2008, p. 13), actual agreement can be calculated for two coders as in Equation 4.39, and for more than two coders as the mean agreement of all coder pairs as in Equation 4.40. In this formulation,

$\mathbf{n}_{ik}$ is the number of times that an item i has been assigned category k by a coder.

$$\mathrm{A}_a = \frac{1}{\mathbf{i}} \sum_{i \in I} \mathrm{agr}_i \tag{4.39}$$

$$\mathrm{A}_a = \frac{1}{\mathbf{ic}(\mathbf{c}-1)} \sum_{i \in I} \sum_{k \in K} \mathbf{n}_{ik}(\mathbf{n}_{ik}-1) \tag{4.40}$$

$$\mathrm{agr}_i = \begin{cases} 1 & \text{if the two coders assign } i \text{ to the same category} \\ 0 & \text{if the two coders assign } i \text{ to different categories} \end{cases} \tag{4.41}$$

Equations 4.39–4.40 represent how agreement is often presently calculated for segmentation. Each item is a boundary position and the set of categories K is the presence or absence of a boundary (an example of which can be seen in Kazantseva and Szpakowicz (2012, p. 214)). Actual agreement in segmentation has also been calculated using other methods in an attempt to mitigate the effects of prevalent near-misses. Hearst (1997, pp. 53) formulated actual agreement as the "proportion of times that coders agree" using the percentage agreement metric of Gale et al. (1992, p. 254). Unfortunately, Gale et al.'s (1992) percentage agreement grossly overestimates agreement by comparing segmenters against the majority opinion of the segmenters chosen (detailed further in Section 3.2).

Agreement based upon Boundary Edit Distance To deal with the prevalence of near misses, alternative methods of calculating actual agreement are proposed herein which utilize boundary-edit-distance-based comparison methods. The comparison methods proposed for these alternative methods of actual agreement calculation are *boundary similarity* (B; variants $\mathrm{B_a}$, $\mathrm{B_b}$, and $\mathrm{B_c}$) and *segmentation similarity* (S). Using *boundary edit distance* these comparison methods can award partial credit for near misses and can measure the proportion of PB positions that have been edited (using varying forms of normalization). In this section, the inter-coder agreement coefficient adaptations are shown that can use any of these four comparison methods. To determine which comparison method is best suited for calculating actual agreement, Section 5.2 experimentally tests the most promising solutions.

When calculating agreement, the goal is to identify positions where there exists agreement and disagreement. This work assumes that the number of edits between two segmentations is equivalent to the number of disagreements. Not all disagreements are of equal value, however, e.g., near misses are not as severe as full misses. Taking advantage of

the transposition and substitution edit scaling that occurs in boundary-edit-distance-based comparison methods can allow actual agreement to less severely penalize near misses in segmentation. All that is left to decide is how to best normalize these edit distances to model these errors. Because agreement in segmentation is to be measured upon the agreement of the choices made by manual coders, and because these choices are to place a boundary or to not, it makes sense to model disagreement in terms of mismatching boundary placements (as in B_a–B_c), as opposed to the mismatched segments (S). Despite this, both S and B_a–B_c are demonstrated later in Chapter 5 to confirm that disagreement should be modelled in terms of mismatching boundary placements.

Actual agreement using boundary-edit-distance-based comparison methods (referred to in formulas as the function φ), is then calculated as in Equation 4.42. The function φ takes three arguments including two boundary strings bs of a document d and coders m, n, and the maximum transposition spanning distance n_t. Equation 4.42 calculates mean agreement using a specific comparison method φ (one of S, B_a, B_b, or B_c) over all coder pairs (i.e., pairwise mean) for every document $d \in D$. Taking the summation of agreement between each boundary string (bs) produced by each pair of coders for all documents is equivalent to performing a summation of agreement over all i in I and dividing by the number of items $\mathbf{i}$ as in Equation 4.40. By performing a summation over all documents $d \in D$ of agreement, one is essentially performing a micro-average over the segmentations of each document. This formulation requires that, for calculating agreement, all coders have coded all documents.[7]

$$\mathrm{A}_{a(\varphi)} = \frac{1}{\binom{\mathbf{c}}{2}} \sum_{m=1}^{\mathbf{c}-1} \sum_{n=m+1}^{\mathbf{c}} \sum_{d \in D} \varphi(bs_d^m, bs_d^n, n_t) \tag{4.42}$$

This formulation also requires, when paired with the expected agreement formulations that follow, that each comparison method model errors as mismatched boundaries, which S does not. Neither, strictly, does P_k or WD (which use windows). Because of this, inter-coder agreement coefficients are not adapted herein to use P_k and WD, but S is used so that a comparison can be made to the formulation using S in Fournier and Inkpen (2012). The units that the ratio produced by Equation 4.42 represents depend upon the comparison method used, and are boundaries for B_a–B_c and potential boundaries for S.

[7] If this is not the case, then multiple agreement values can be computed for each group of coders who have coded the same sets of documents, otherwise a formulation is required that does not have this requirement.

Expected agreement for segmentation Traditionally, for linear segmentation with one boundary type, the set of categories K is defined as the presence of a boundary, or the absence of it, for each item (i.e., at each boundary position; Passonneau and Litman 1993, pp. 150–151, Hearst 1997, pp. 53–54, Kazantseva and Szpakowicz 2012, p. 214). This category interpretation is unsuitable, because

> "...coders have only two choices when segmenting: to place a boundary, or not. Coders do not place non-boundaries. If they do not make a choice, then the default choice is used: no boundary. This default option makes it impossible to determine whether a segmenter is making a choice by not placing a boundary, or whether they are not sure whether a boundary is to be placed."
>
> – Fournier and Inkpen (2012, p. 156)

For this reason, when characterizing chance agreement between coders, the categories used herein are one boundary presence category per type, as in Equation 4.43.

$$K = \{k_t | t \in T\} \tag{4.43}$$

This representation reflects a coder's decision to place a boundary at a position in a segmentation, and reflects the types of boundary that they chose to place. This allows the formulation of agreement for segmentation proposed herein to apply to segmentation studies which utilize multiple boundary types (placed mutually exclusively or not).

Scott's π Proposed by Scott (1955), π is a coefficient that measures agreement between two coders. It makes the assumption that chance agreement between coders would be indicated by all coders placing boundaries according to a single distribution per category k. Because of this assumption, π distinguishes chance agreement between categories, but not coders, and thus offers a generalization of the performance of all coders. Its expected agreement is normally calculated as formulated in Artstein and Poesio (2008, p. 9) (shown in Equation 4.44) with actual agreement calculated as detailed previously in Equation 4.42.

$$\mathrm{A}_e^{\pi} = \sum_{k \in K} \left(\mathrm{P}_e^{\pi}(k)\right)^2 \tag{4.44}$$

Chance agreement per category ($\mathrm{P}_e^{\pi}(k)$) is the number of codings of a category k by both coders ($\mathbf{n}_k$) over the total number of items $\mathbf{i}$ from both coders (Artstein and Poesio,

2008, p. 9).

$$\mathrm{P}_e^{\pi}(k) = \frac{\mathbf{n}_k}{2\mathbf{i}} \tag{4.45}$$

For segmentation, the same formulation of expected agreement is proposed, but with a set of categories tailored to segmentation. This set of tailored categories is comprised of one category per boundary type (which indicates the presence of that boundary type, i.e., Equation 4.43). For convenience, appearance of the formulation is adjusted as shown in Equation 4.46. In this formulation, $\mathrm{b}(bs_d^i, k)$ is the count of boundaries of type k placed in a boundary string bs by coder i (i.e., for coders 1 and 2, $\mathbf{n}_k$) and $\mathrm{b}(bs_d^i)$ the total number of items coded by that coder i for a document d (i.e., for coders 1 and 2, $\mathbf{i}$).

$$\mathrm{P}_{e(\varphi)}^{\pi}(k) = \frac{\sum_{d \in D} \mathrm{b}(bs_d^1, k) + \sum_{d \in D} \mathrm{b}(bs_d^2, k)}{\sum_{d \in D} \mathrm{b}(bs_d^1) + \sum_{d \in D} \mathrm{b}(bs_d^2)} \tag{4.46}$$

Using this formulations of $\mathrm{P}_e^{\pi}(k)$ allows us to compute A_e^{π} for segmentation—referred to as $\mathrm{A}_{e(\varphi)}^{\pi}$. With both $\mathrm{A}_{a(\varphi)}$ (Equation 4.42) and $\mathrm{A}_{e(\varphi)}^{\pi}$, a new form of Scott's π (referred to as π_φ) can be formulated using the general form of Equation 4.38. This formulation (see Equation 4.47) discounts near miss errors using $\mathrm{A}_{a(\varphi)}$ and better reflects chance agreement due to coder decisions using $\mathrm{A}_{e(\varphi)}^{\pi}$.

$$\pi_\varphi = \frac{\mathrm{A}_{a(\varphi)} - \mathrm{A}_{e(\varphi)}^{\pi}}{1 - \mathrm{A}_{e(\varphi)}^{\pi}} \tag{4.47}$$

Cohen's κ Proposed by Cohen (1960), κ is a coefficient that measures agreement between two coders. It makes the assumption that chance agreement between coders would be indicated by all coders placing boundaries according to individual distributions per coder and per category. Because of this assumption, κ distinguishes chance agreement between categories and coders while not generalizing to a group of coders and is thus susceptible to the issues of both bias and prevalence (see Section 3.2). Its expected agreement is normally calculated as formulated in Artstein and Poesio (2008, pp. 9–10) (shown in Equation 4.48) with actual agreement calculated as detailed previously in Equation 4.42.

$$\mathrm{A}_e^{\kappa} = \sum_{k \in K} \mathrm{P}_e^{\kappa}(k|c_1) \cdot \mathrm{P}_e^{\kappa}(k|c_2) \tag{4.48}$$

Chance agreement per category per coder ($\mathrm{P}_e^{\kappa}(k|c_i)$) is the number of codings of a category k by a coder, $\mathbf{n}_{c_i k}$, over the total number of items $\mathbf{i}$ coded (Artstein and Poesio,

2008, pp. 10).

$$\mathrm{P}_e^{\kappa}(k|c_i) = \frac{\mathbf{n}_{c_i k}}{\mathbf{i}} \tag{4.49}$$

For segmentation, we can re-use this formulation of expected agreement for κ but with our specific set of categories of one per boundary type (which indicates the presence of that boundary type). For convenience, we can adjust the appearance of the formulation as in Equation 4.50. In this formulation, $\mathrm{b}(bs_d^{c_i}, k)$ is the count of boundaries of type k placed in a boundary string bs by coder c_i (i.e., $\mathbf{n}_{c_i k}$) and $\mathrm{b}(bs_d^{c_i})$ the total number of items coded by that coder c_i for a document d (i.e., $\mathbf{i}$).

$$\mathrm{P}_e^{\kappa}(k|c_i) = \frac{\sum_{d \in D} \mathrm{b}(bs_d^{c_i}, k)}{\sum_{d \in D} \mathrm{b}(bs_d^{c_i})} \tag{4.50}$$

Using this formulation of $\mathrm{P}_e^{\kappa}(k|c_i)$ allows us to compute A_e^{κ} for segmentation—referred to as $\mathrm{A}_{e(\varphi)}^{\kappa}$. With both $\mathrm{A}_{a(\varphi)}$ (Equation 4.42) and $\mathrm{A}_{e(\varphi)}^{\kappa}$, a new form of Cohen's κ (referred to as κ_{φ}) can be formulated using the general form of Equation 4.38. This formulation (see Equation 4.51) discounts near miss errors using $\mathrm{A}_{a(\varphi)}$ and better reflects chance agreement due to coder decisions using $\mathrm{A}_{e(\varphi)}^{\kappa}$.

$$\kappa_{(\varphi)} = \frac{\mathrm{A}_{a(\varphi)} - \mathrm{A}_{e(\varphi)}^{\kappa}}{1 - \mathrm{A}_{e(\varphi)}^{\kappa}} \tag{4.51}$$

Fleiss's multi-π Proposed by Fleiss (1971), multi-π (π^*) is an adaptation of Scott's π for measuring inter-coder agreement between multiple annotators. Artstein and Poesio (2008, p. 14) formulate the calculation of expected agreement for π^* in the same manner as π (Equation 4.52), with only a change in how $\mathrm{P}_e^{\pi}(k)$ is calculated.

$$\mathrm{A}_e^{\pi^*} = \sum_{k \in K} \left(\mathrm{P}_e^{\pi^*}(k)\right)^2 \tag{4.52}$$

$\mathrm{P}_e^{\pi}(k)$ is now calculated for multiple coders as being the number of codings of a category k by all coders, $\mathbf{n}_k$, over the total number of items $\mathbf{i}$ and coders $\mathbf{c}$, as in Equation 4.53 Artstein and Poesio (2008, pp. 13).

$$\mathrm{P}_e^{\pi^*}(k) = \frac{\mathbf{n}_k}{\mathbf{ic}} \tag{4.53}$$

For segmentation, we can use this same formulation of expected agreement for π using the new set of segmentation categories (one per boundary type which indicates the

presence of that type). For convenience, we can adjust the appearance of the formulation as in Equation 4.54. In this formulation, $\mathrm{b}(bs, k)$ is the count of boundaries of type k placed in a boundary string bs and $\text{max\_penalties}_{\varphi}(bs_d^1, bs_d^2, d, k)$ is the normalization used by the comparison method φ summed over all coders and all items (i.e., documents).

$$\mathrm{P}_e^{\pi^*}(k) = \frac{\sum_{c \in C} \sum_{d \in D} \mathrm{b}(bs_d^c, k)}{\sum_{c \in C} \sum_{d \in D} \mathrm{b}(bs_d^c)} \tag{4.54}$$

Using this formulations of $\mathrm{P}_e^{\pi^*}(k)$ allows us to compute $\mathrm{A}_e^{\pi^*}$ for segmentation—referred to as $\mathrm{A}_{e(\varphi)}^{\pi^*}$. With both $\mathrm{A}_{a(\varphi)}$ (Equation 4.42) and $\mathrm{A}_{e(\varphi)}^{\pi^*}$, a new form of Fleiss's multi-π (referred to as π_{φ}^*) can be formulated using the general form of Equation 4.38. This formulation (see Equation 4.55) discounts near miss errors using $\mathrm{A}_{a(\varphi)}$) and better reflects chance agreement due to coder decisions using $\mathrm{A}_{e(\varphi)}^{\pi^*}$.

$$\pi_{\varphi}^* = \frac{\mathrm{A}_{a(\varphi)} - \mathrm{A}_{e(\varphi)}^{\pi^*}}{1 - \mathrm{A}_{e(\varphi)}^{\pi^*}} \tag{4.55}$$

Fleiss's multi-κ Proposed by Davies and Fleiss (1982), multi-κ (κ^*) is an adaptation of Cohen's κ for measuring inter-coder agreement between multiple annotators. Artstein and Poesio (2008, p. 14) formulate the calculation of expected agreement for κ^* as the pairwise mean of the joint probability of a pair of coders c_m and c_n assigning category k.

$$\mathrm{A}_e^{\kappa^*} = \sum_{k \in K} \left(\frac{1}{\binom{\mathbf{c}}{2}} \sum_{m=1}^{\mathbf{c}-1} \sum_{n=m+1}^{\mathbf{c}} \mathrm{P}_e^{\kappa}(k|c_m) \cdot \mathrm{P}_e^{\kappa}(k|c_n) \right) \tag{4.56}$$

In this case, the formulation of $\mathrm{P}_e^{\kappa}(k|c_i)$ remains identical to the Artstein and Poesio's (2008) in Equation 4.49 and the segmentation adaptation defined here in Equation 4.50. With both $\mathrm{A}_{a(\varphi)}$ (Equation 4.42) and the adapted version of $\mathrm{A}_e^{\kappa^*}$ (referred to as $\mathrm{A}_{e(\varphi)}^{\kappa^*}$), a new form of Davies and Fleiss's (1982) multi-κ (referred to as κ_{φ}^*) can be formulated using the general form of Equation 4.38. This formulation (see Equation 4.57) discounts near miss errors using $\mathrm{A}_{a(\varphi)}$) and better reflects chance agreement due to coder decisions using $\mathrm{A}_{e(\varphi)}^{\kappa^*}$.

$$\kappa_{\varphi}^* = \frac{\mathrm{A}_{a(\varphi)} - \mathrm{A}_{e(\varphi)}^{\kappa^*}}{1 - \mathrm{A}_{e(\varphi)}^{\kappa^*}} \tag{4.57}$$

Annotator bias Having defined segmentation versions of both π^* and κ^*, the degree of individual coder bias present in a study can be determined using a measure of variance

proposed by Artstein and Poesio (2008, pp. 23–24) and shown in Equation 4.58. This variance is quantified in terms of the difference between the expected agreement of π and κ. This is the difference between chance agreement exemplified by a single distribution over categories (regardless of individual coder distributions) and chance agreement exemplified by a per coder and per category distribution. It is formulated in this way because $A_e^{\pi^*}$ is always greater than $A_e^{\kappa^*}$ (Artstein and Poesio, 2008, pp. 59–60), giving us a positive real number representing the "variances of $\mathrm{P}(k|c)$ for all categories $k \in K$ divided by the number of coders $\mathbf{c}$ less one" (Artstein and Poesio, 2008, p. 24).

$$Bias = A_e^{\pi^*} - A_e^{\kappa^*} = \frac{1}{\mathbf{c}-1} \sum_{k \in K} \sigma^2_{\mathrm{P}(k|c)} \tag{4.58}$$

This bias measurement can be calculated using the segmentation versions of the inter-coder agreement statistics defined herein. Bias is a useful method of determining the degree to which an individual coder bias is present in a segmentation study. It can determine whether manual codings were produced by coders who coded in a manner quite differently from each other, whereas π simply informs us of whether our coding scheme and instructions are adequate.

Statistical significance of inter-coder agreement coefficients Although one cannot test whether a statistically significant degree of inter-coder agreement exists, it is possible to test whether there is a statistically significant chance that agreement is greater than zero. This can be accomplished using hypothesis testing where the null hypothesis is that the coefficient is zero and the alternate hypothesis is that it not.

For π^*, Siegel and Castellan (1988, Section 9.8.2) state that the sampling distribution of K (i.e., π^*), given large sample sizes[8], is approximately normal and centred around zero. This distribution "allows testing the obtained value of K against the null hypothesis of chance agreement by using the z statistic." (Artstein and Poesio, 2008, p. 28). We can obtain the variance of K using Equation 4.59 (Siegel and Castellan, 1988, pp. 284–290)

$$\mathrm{Var}(\pi^*) = \frac{2 \cdot \Big(\big(\sum_{k \in K} p_k(1-p_k) \big)^2 - \sum_{k \in K} p_k(1-p_k)(1-2 \cdot p_k) \Big)}{\mathbf{ic}(\mathbf{c}-1) \big(\sum_{k \in K} p_k(1-p_k) \big)^2} \tag{4.59}$$

$$p_k = \frac{1}{\mathbf{ic}} \sum_{i \in I} \mathbf{n}_{ik} \tag{4.60}$$

[8] A large sample size could be reasonably assumed to be $\mathbf{i} > 100$ as it is by Cohen (1960, pp. 44).

Using this variance, we can perform a z-test by taking our π^* value and dividing it by its standard deviation (i.e., square root of the variance; shown in Equation 4.61). This critical ratio z can then be compared to the normal curve to determine (for a predefined significance level α) whether the null hypothesis ($\pi^* = 0$) can be rejected.

$$z = \frac{\pi^*}{\sqrt{\mathrm{Var}(\pi^*)}} \tag{4.61}$$

For κ, Cohen (1960, pp. 44–45) states that its sampling distribution is normal for large sample sizes ($\mathbf{i} \geq 100$). This distribution allows for confidence intervals to be determined and a z-test to be performed (as with π^*). The standard deviation of the coefficient can be calculated as in Equation 4.62, and the critical ratio z as in Equation 4.63. This critical ration can again be compared to the normal curve to determine (for a predefined significance level α) whether the null hypothesis ($\kappa = 0$) can be rejected.

$$\sigma_{\kappa_o} = \sqrt{\frac{A_e^\kappa}{\mathbf{i}(1 - A_e^\kappa)}} \tag{4.62}$$

$$z = \frac{\kappa}{\sigma_{\kappa_o}} \tag{4.63}$$

Reporting agreement Using the new formulation of the inter-coder agreement coefficient π^* as adapted herein (referred to as π^*_φ), reliability of a segmentation coding scheme and instructions can be more clearly evaluated. This is because π^*_φ can account for near-miss errors that previous formulations could not by using boundary-edit-distance-based comparison methods to calculate actual agreement. One can also identify whether individual coder bias has affected a particular segmentation study by using Artstein and Poesio's (2008) bias, and attempt to infer the validity of data using κ^*_φ when agreement is not perfect.

Given these coefficients, how should they be reported? Artstein and Poesio (2008, pp. 58) indicate that studies should report

> "the methodology that was followed to collect the reliability data (number of coders, whether they coded independently, whether they relied exclusively on an annotation manual). The study should also indicate whether agreement was statistically significant, and provide the confusion matrix [e.g., Figures 3.9-3.10] or agreement table so that readers can find out whether overall figures of agreement hide disagreements on less common categories."

– Fournier and Inkpen (2012, p. 156)

When choosing a specific coefficient to report, the context must be taken into account. When coding scheme reliability is being evaluated, π^*_φ is the appropriate statistic to report because it generalizes the agreement of individual coders and is unaffected by individual coder bias. If manual segmentations are being evaluated for their suitability as training data, then one wishes to determine whether individual coder bias is present, and thus one should report κ^*_φ. If one is attempting to use inter-coder agreement to evaluate automatic segmenters (detailed in Section 4.3), then individual coder bias is desired and κ^*_φ should be reported.

4.3 Evaluating Segmenters

For a segmentation task, one often has a choice between two or more automatic segmenters to use. Barring an ecological evaluation, one can perform a statistical comparison of automatic segmenters over a variety of data sets to identify which is the best for the task at hand. To determine which automatic segmenter to choose, there are a number of desirable qualities which one may wish to obtain, such as the:

- Ability to approximate a reference solution (potentially comprised of one or more manual segmentations);
- Domain independence (in terms of the document topic, type, language, etc.);
- Speed and efficiency in usage/training; and
- Fewer high-cost errors.

This work focuses upon evaluating whether an automatic segmenter is able to approximate a reference solution comprised of one or more manual segmentations and determining the types of errors that occur. This work assumes that:

- Domain independence is accounted for by the choice of a suitably diverse set of documents;
- Speed and efficiency are discussed on a case by case basis in other works; and
- Error for the segmentation task being evaluated is best modelled by similarity (if not, then another metric such as a boundary edit distance based IR metric should be used).

Specifically, this section proposes the usage of existing methodologies from other fields to this natural language processing task and the use of new measures and methods proposed herein, including:

- Use of an upper and lower bound during segmenter comparisons;
- Use of macro-averages and statistical hypothesis testing methods with satisfactory statistical power and associated post-hoc tests (e.g., ANOVA and a post-hoc test as opposed to paired T-tests);
- Comparison of automatic segmenter performance through drops in inter-coder agreement; and
- A new per-boundary-pair based micro-average using B_b.

Mean statistics given multiple codings In segmentation, there is often no one "true" segmentation. Instead, there are multiple interpretations of how one document should be segmented. For this reason, to perform a meaningful evaluation, an automatic segmenter must often be compared against multiple manually produced segmentations (i.e., codings) of the same text. This presents a number of challenges when it comes to calculating the mean performance of an automatic segmenter.

Because the performance of an automatic segmenter must be evaluated over multiple documents and over multiple codings of each document, the mean performance is then dependent upon two variables: coders and documents. To measure mean performance while taking into account these two variables requires us to compute either a:

- Weighted macro-average to account for the varying sizes of documents; or
- Micro-average that can use individual boundary pairs as samples as opposed to whole documents.

This is because documents can vary in length, meaning that they also vary in the number of potential boundaries within them. Varying numbers of potential boundaries means that documents have the potential for varying amounts of error which could obscure their effects upon a mean computed as an unweighted macro-average.

Weighting macro-averages to compensate for varying document size and numbers of coders can only probabilistically and inexactly compensate for such variations. Differing document size may affect the number of boundaries placed within such a document, but

the relationship is probabilistic and not deterministic. This probabilistic relationship makes weighting by document size imprecise and—as a result—undesirable. Differing numbers of coders per document should increase the total boundaries placed within a document by coders while keeping the mean and variance relatively unchanged—in a well designed and administered study, but this is again probabilistic and inexact. In light of these issues, micro-averages give the clearest indicator of relative performance.

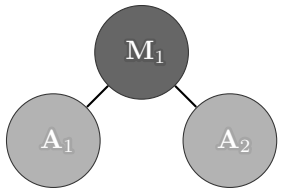
To compute micro-averages in the presence of varying coders and documents, pairwise comparisons must be made where each pair must contain a member of the set of coders and a member of the set of automatic segmenters. From each such comparison, those comparison methods which are ratios (e.g., S and B_a–B_c) can sum their numerators and denominators before performing their final division to compute a micro-average. To discuss these comparisons further, a notation is defined herein where:

- *Coders* is the set $\{c|c \in C\}$ with cardinality $\mathbf{c}$;
- *Documents* is the set $\{d|d \in D\}$ with cardinality $\mathbf{d}$;
- $A_{a,i}$ represents the segmentation by an automatic segmenter a of a document i;
- $M_{c,i}$ represents the segmentation by a manual coder c of a document i;
- $c(a, b)$ represents an arbitrary segmentation comparison function whose arguments are segmentations and returns a real number; and
- *Comparisons* is the set $\{x_{a,c,i}|x_{a,c,i} \in X\}$ with cardinality $\mathbf{x}$ where each comparison $x_{a,c,i} = c(a, b)$ with $a = A_{a,i}$, $b = M_{c,i}$, and i represents a particular document.

For each comparison $x_{a,c,i}$ the numerator and denominator of a ratio-based comparison method can be summed to produce a micro-average as in Equation 4.64.

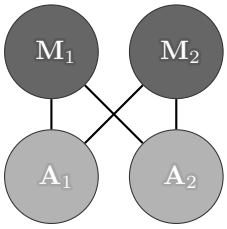
$$\varphi_{\text{micro}} = \frac{\sum \varphi_{\text{numerator}}(x_{a,c,i})}{\sum \varphi_{\text{denominator}}(x_{a,c,i})} \qquad \text{where } \varphi \text{ is one of S, } B_a, B_b, \text{ or } B_c \tag{4.64}$$

If one only has a single segmentation by a single coder to compare two automatic segmenters against, determining the performance of the automatic segmenters is as simple as applying a comparison method, e.g., $c(A_a, M_c)$. Applying a comparison method twice as $c(A_1, M_1)$ and $c(A_2, M_1)$ can determine the performance of two automatic segmenters as in Figure 4.26a. From the comparisons made, the performance of segmenters can



X
$c(A_1, M_1)$
$c(A_2, M_1)$

(a) Comparison of a single segmentation by a single manual coder (M_1) to two automatic segmenters



X
$c(A_1, M_1)$
$c(A_2, M_1)$
$c(A_1, M_2)$
$c(A_2, M_2)$

(b) Comparison of two segmentations by two manual coders (M_1, M_2) to two automatic segmenters

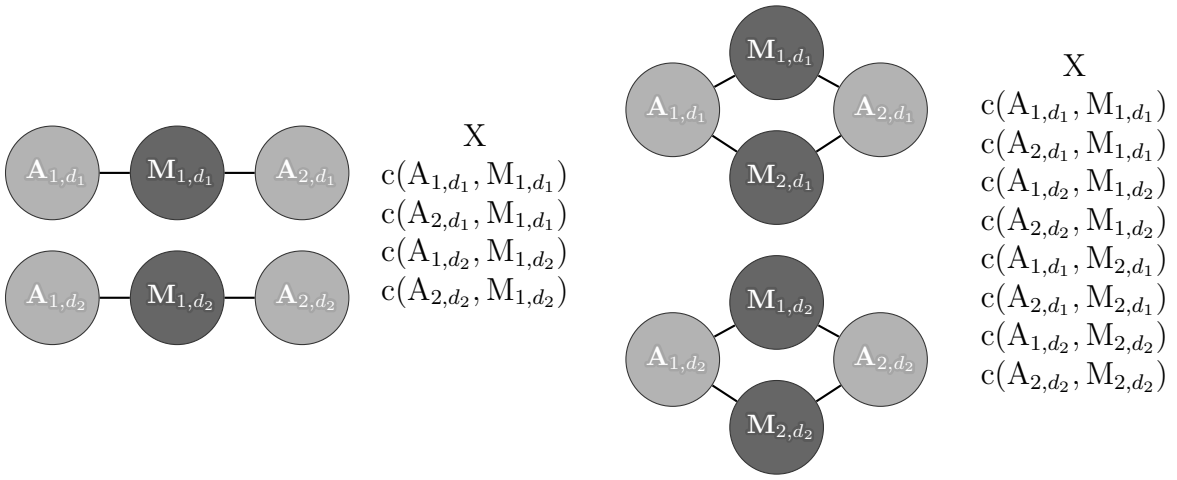
Figure 4.26: Comparisons of segmentations using the method $c(A_a, M_c)$ between automatic (A_a) and manual segmenters (M_c)

be computed as $x_{A_1} = c(A_1, M_1)$ and $x_{A_2} = c(A_2, M_1)$. These two values can then be compared directly. Unfortunately, no general inferences can be made about how these two segmenters perform when they are compared using the population of all coders and all documents as a solution. This is because only one coder and one document is used.

When more than one coding is used, the number of comparisons that must be made increases, as in Figure 4.26b. This increase is because there is now more than one solution to compare against. The number of comparisons made is then $\mathbf{x} = \mathbf{a} \cdot \mathbf{c}$, where $\mathbf{a}$ and $\mathbf{c}$ are the cardinalities of the sets of all analysed automatic and manual segmenters, respectively. To determine whether one segmenter performs better than another, one can compute the micro-average of one automatic segmenter's performance over all manual codings.

In the presence of multiple documents which have been annotated by a single coder, one can determine the performance of an automatic segmenter by taking the micro-average of the comparisons made as in Figure 4.27a. If multiple coders have annotated multiple documents, the same calculation can be performed upon comparisons as in Figure 4.27b (which are more numerous). In this way, each automatic segmenter is evaluated against all coders and all documents to produce a single pairwise micro-average.

Statistical hypothesis testing given multiple codings The mean values calculated previously are simply estimates of what the real population mean (μ) would be if one were to calculate the mean using every possible document segmented by every possible coder. Unfortunately, these estimates are not accurate and contain error (as outlined in Section 3.3). In the presence of this error, one must utilize additional methods to judge whether one automatic segmenter is better than another in the presence of this error. One such method is statistical hypothesis testing.



(a) Comparison of segmentations of two documents $(\mathrm{d}_1, \mathrm{d}_2)$ by one manual coder (M_1) versus two automatic segmenters

(b) Comparison of segmentations of two documents $(\mathrm{d}_1, \mathrm{d}_2)$ by two manual coders $(\mathrm{M}_1, \mathrm{M}_2)$ versus two automatic segmenters

Figure 4.27: Comparisons of segmentations from multiple documents using the method $\mathrm{c}(A_a, M_c)$ between automatic (A_a) and manual segmenters (M_c)

Using statistical hypothesis testing, and given an appropriate segmentation comparison method, one can determine—with a degree of accuracy—whether one automatic segmenter performs better than another in the presence of our estimate's error. First, an appropriate test statistic and comparison procedure are chosen given various assumptions about the data as detailed by Vázquez et al. (2001) and using algorithms such as those described in Figure 2.5 (see Section 3.3). Then, having multiple automatic segmenters, one's goals is to obtain performance comparisons over multiple documents and coders which will comprise one's mean performance estimates. To determine whether the differences between them are significantly different or not, one formulates two hypothesis where for each it states that the number automatic segmenter performances means compared are probably:

- H_0: from the same populations (i.e., not statistically significantly different); or
- H_1: not from the same populations (i.e., statistically significantly different).

If the two means are not from the same population, then they are statistically different (either less so or more so than the other). To determine how significantly they differ, a significance level α must be selected, which is the threshold below which the probability of the null hypothesis must be before one can reject in it favour of the alternate hypothesis. "By convention, if $p < 0.05$ [one says] the result is statistically significant, and if $p < 0.01$

[one says] the result is highly significant and [one] can be more confident [that they] have found a true effect" (Cumming et al., 2007), or in this case, a true difference between segmenters.

When performing this testing, the definition of our subject is important. When determining which automatic segmenter has the best general performance over all possible documents, our subjects are the manual document codings. Automatic segmenters are a factor in our experiment (where these various automatic segmenters are each a level of this factor). Because the levels in this factor—i.e., applying one automatic segmenter to a singly coded document—are independent of each other, a within-subjects (i.e., repeated measure) experiment design can be used. A within-subjects experiment design is the usage of the same subjects within each treatment. This allows us to maintain a high n, or number of subjects, which helps one reduce the error inherent in mean performances estimations.

There often does not exist, however, one, true, manual segmentation of a document to compare against. Instead, there exist multiple interpretations of the segmentation of a document, i.e., codings. To model this, one can retain their interpretation of their subject being document codings but use a second factor to model that each document is multiply coded. Assuming that all coders code all documents, evaluating $\mathbf{a} = 3$ automatic segmenters over $\mathbf{d} = 5$ documents each coded by $\mathbf{c} = 3$ coders produces the comparison method values shown in Table 3.4.

Using the 2-factor design in Table 3.4, statistical hypothesis testing can determine whether there is a statistically significant difference between automatic segmenters. This hypothesis testing can also determine whether an automatic segmenter's performance is influenced by either coders or documents (or both). This can be accomplished using a 2-factor analysis of variance (ANOVA; Chambers et al. 1992) if its assumptions are satisfied (normality and homoscedasticity). If ANOVA's assumptions are not met, then, to maintain a reasonable degree of power, a non-parametric statistical test such as the 2-factor Friedman's rank sum test (Hollander and Wolfe, 1973, pp. 139–146) should be used. ANOVA and Friedman's rank sum test, however, should only be used to compare the mean performance of three or more automatic segmenters because for pairs there are tests with greater power such as Student's T-test. Following each test, if there are statistically significant differences present, then they can be followed by a multiple comparison post-hoc. A post-hoc test determines between which automatic segmenters a significant difference in performance exists, and is chosen as specified in Table 3.5.

Analysis may be simpler in a 1-factor within-subjects design, however. Because each

coding of a document is independent of all the others, it could be assumed that each coding of each document is a subject. This assumption which would allow the use of a 1-factor design where the only factor is the set of automatic segmenters being evaluated against each other using some segmentation comparison method.

Statistical hypothesis testing and micro-averages using B_b B_b is different from the other proposed comparison methods in that it can distinguish expected and actual boundary pairs from each other. When calculating B_b, a full miss is a boundary pair where one side is missing a boundary and has a reward of 0, a match is a pair with a reward of 1, and a near miss/substitution is a pair which has a reward equal to 1 minus the scaled penalty (see Table 4.3). B_b is the sum of these reward values for each pair divided by the number of pairs, which are all computed from the edits obtained from boundary edit distance. The mean of the set of these boundary pair reward values is equivalent to a micro-average of B_b where its numerators and denominators are summed before their division. This is especially useful when performing statistical hypothesis testing or analysing standard error of the mean. Using B_b, the subject can be defined as independent boundary pairs themselves as opposed to a more abstract entity such as a coder and document pair.

Using B_b, performance can be determined for each pair of boundaries, with the subject being defined then as the boundary pair itself. Many of these pairs will repeat because many pairs will result from the manual segmentation that an automatic segmenter is compared against. Many pairs will not be repetitions because they do not map to pairs in manual segmentations but are caused by a boundary in the automatic segmentation. Because of this uniqueness, a between-subjects experiment design can be used with 1 or 2-factor ANOVA if the test's assumptions are met. If ANOVA's assumptions are not met, then, to maintain a reasonable degree of power in the test chosen, a non-parametric statistical test such as the Kruskal-Wallis rank sum test (Hollander and Wolfe, 1973, pp. 115-120) can be selected along with a post-hoc test as described in Table 3.5. ANOVA and the Kruskal-Wallis rank sum test, however, should only be used to compare the mean performance of three or more automatic segmenters because for pairs there are tests with greater power such as Student's T-test.

Reasons for choosing ANOVA, Friedman's, or the Kuskal-Wallis test The for choosing these tests that require three or more levels in factors are used is because the number of automatic segmenters is, in most cases, going to be greater than or equal to

three. Otherwise, Student's T-test or other tests between pairs of automatic segments could be used. This number of factors is the result of a well designed evaluation which:

- Compares two automatic segmenters; and
- Also compares either an upper or lower performance bound (e.g., a human segmenter or random segmenter, respectively) or both.

This means that there will be at least 3–4 segmenters compared to identify any floor or ceiling effects (Cohen, 1995, p. 80–82). A complicated and resource-intensive automatic segmenter which improves upon a baseline by a few percentage points is, for most tasks, not a reasonable improvement in performance. Likewise, a series of seemingly poorly performing automatic segmenters may in fact have reasonable performance in comparison to humans—indicating that the task itself, coding scheme, or instructions may be ill-defined.

Comparisons of agreement for segmentation Due to the lack of a single, true, manual segmentation, statistical hypothesis testing becomes more difficult to apply and to interpret the interactions between coders and automatic segmenters. It would be desirable to have a single number for each automatic segmenter that could be compared to easily determine whether one segmenter better approximates manual performance than another. One type of statistic that can produce such a number is an inter-annotator agreement coefficient.

In addition to gauging the reliability of a study, inter-coder agreement coefficients can also reflect the validity of a set of codings in terms of their proximity to complete agreement. If one study is replicated using the same coding system and documents using different manual coders and it obtains agreement that differs from the original study, then one set of coders produced a more valid solution. Such a phenomenon should be further investigated to determine its cause.

Taking a set of manual coders and determining their agreement can also give one an idea of the upper bound to performance for a task. If one adds another manual coding to this set, one would gain more information about human agreement, and the agreement of the group would either increase, decrease, or stay constant. If agreement decreases, then the added coder was probably a poor coder placed amongst existing skilled coders because a group of existing poor coders would not likely have high agreement to start

with.[9] If agreement increases, then one has probably added a coding which agrees with the majority of coders.

Adding one coder to a group of existing coders may have the effect of increasing an inter-coder agreement coefficient when the coding is similar to the majority opinion. Because of this, it is proposed herein that this can be used to evaluate the performance of automatic segmenters if no one manual solution represents a "true" segmentation. Specifically, this method is most useful when:

- All coders (including automatic segmenters) coded all items;
- Manual codings exhibits high agreement;
- Sufficient numbers of manual coders are used; and
- Qualitative analyses of the manual codings verify that any bias present is desired.[10]

If all of these are qualities are present, then one can compare drops in inter-annotator agreement between groups with and without adding automatic segmenters to a group of manual coders, one at a time. This is much like how drops in inter-indexer consistency have been used by Medelyan (2009) and Medelyan et al. (2009) to relate the performance of an automatic topic indexer to human performance.

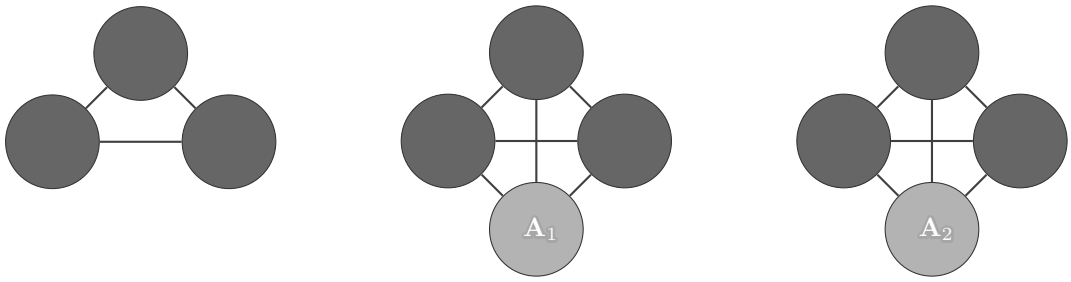
For this method of automatic segmenter comparison, one should prefer to use a coefficient that is sensitive to both bias and prevalence to ensure that a drop in agreement due to differing coding behaviour is recorded. κ and κ^* should then be used. When calculating κ for segmentation comparison, assuming that one has 3 manual coders and two automatic segmenters, one would calculate three values as represented by Figures 4.28a–4.28c, referred to as κ^*_M, $\kappa^*_{A_1}$ and $\kappa^*_{A_2}$, respectively.

Comparing these three agreement values can help answer two questions:

- Does an automatic segmenter perform better or worse than average human performance?
- In terms of average human performance, does one segmenter perform better than another?

[9] Poor random coders agree by chance which is corrected for by agreement coefficients, although poor degenerate coders who simply place all/no boundaries will be an issue if they constitute a large portion of the codings—this can be identified qualitatively.

[10] I.e., the majority of coders appear to be performing the desired task and are not acting in a degenerate manner, e.g., placing boundaries at all locations.



(a) κ^*_M: agreement between manual coders

(b) $\kappa^*_{A_1}$: agreement between manual coders and one automatic segmenter (A_1)

(c) $\kappa^*_{A_2}$: agreement between manual coders and one automatic segmenter (A_2)

Figure 4.28: Calculating inter-coder agreement using three different sets of coders to compare two automatic segmenters against each other in terms of drops in agreement and to compare both against human agreement.

The first question is answered by comparing to see whether adding an automatic segmenter increases or decreases agreement by a significant degree. If not, then it is on par with human performance (a desirable property, nonetheless). Comparing the coefficients obtained when adding each automatic segmenter can give us an indication of whether one automatic segmenter better approximates human performance than another.

This method of comparison has the advantage that it can put automatic segmenter performance in terms of average human performance in a way that accounts for the probability of chance agreement. It is not without its weaknesses, though. The definition of what constitutes high manual agreement, a sufficient number of coders, and a satisfactory qualitative analysis must be reasonably defined on a case by case basis and cannot easily be prescribed. This makes the method difficult to apply and interpret. The definition of a significant increase/decrease requires interpretation, and no acceptable method of performing statistical hypothesis testing can be performed to interpret such increases/decreases. This method of comparison is then best used as a supplement to statistical hypothesis testing to help identify whether hypothesis testing arrived at a correct result and could be reported along with such tests.

How did an automatic segmenter fail? Although it is usually impossible to answer why a segmenter performed poorly in comparison to another, using boundary edit distance one can gain more insight into how failure occurred. Boundary edit distance gives the number of edits between two arbitrary segmentations, and can separate out these edits into three main types (with subgroups and other distinctions able to be made within

each):

- Additions/deletions, i.e., *full misses* of a boundary type;
- Substitutions, i.e., *confusion* between boundary types at positions; and
- Transpositions, i.e., *near misses*.

Further, for additions/deletions, the segmentation containing the boundary can be recorded or the edit (e.g., was the boundary found in the manual segmentation, i.e. a FN, or an automatic segmentation, i.e. a FP). Substitutions can also be classified by the boundary types that they confuse, and whether a particular incorrect boundary type is being confused for anther by an automatic segmenter.

In the presence of multiple codings, one can perform comparisons using boundary edit distance as in Figure 4.27b. For each comparison one can sum the number of edits that boundary edit distance identifies together into the three edit types (and associated boundary subtypes) For comparison, the number of boundary matches made should also be given to put the number of errors into perspective. These counts, if performed for multiple automatic segmenters, can then be compared against each other to provide a comparison of how each performed in terms of the pairwise errors (i.e., edits) that they produced. An example of such error counts is shown in Figure 4.29, where two hypothetical automatic single-boundary-type segmenters have been compared against multiple manual codings. The total sum of the error types encountered during these pairwise comparisons is shown visually, which allows us to understand what proportion of errors were near misses versus full misses out of all errors. Further, it demonstrates what proportion of the errors were FPs versus FNs, and how the errors compare to the number of successful matches.

This comparison informs us of the types of errors that have been committed by automatic segmenters. These counts can also tell us whether perhaps our evaluation is too harsh or not (e.g., if very few near misses are found then perhaps the maximum transpositions spanning distance used by boundary edit distance should be increased). It can also tell us whether a segmenter is guessing too often and has low precision (many FPs, e.g., A_2) or precise with low recall (many FNs, e.g., A_1). In fact, these edit counts can be use to craft a multi-class confusion matrix allowing us to obtain IR metrics such as precision and recall (as shown in Section 4.1.3). However, these metrics should not be the sole method of evaluating automatic segmenter errors because they conflate transpositions and substitutions into the true positive (TP) category. This conflation also obscures the

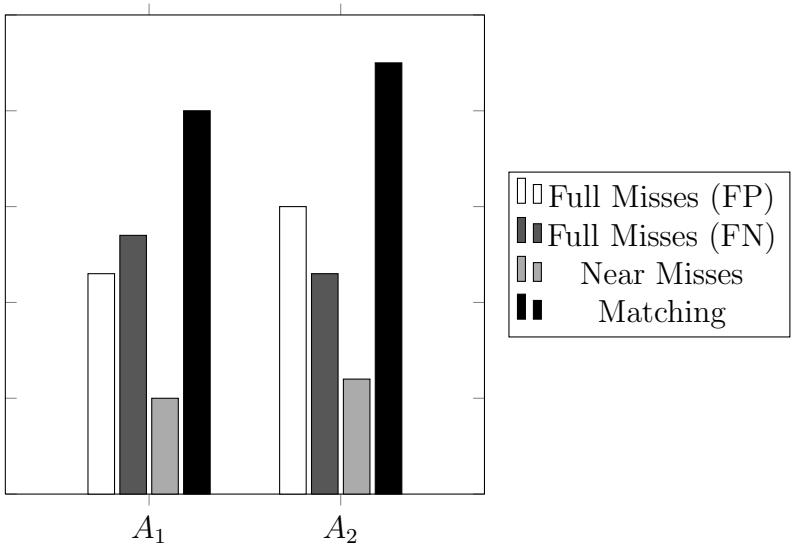


Figure 4.29: Single-boundary type segmentation performed by two automatic segmenters and compared against multiple manual coders to produce counts of the types of errors committed out of the total number of possible errors

near-miss and substitution errors which are concepts of precision and rely upon only the presence or lack of a boundary (FPs and FNs), and should be observed separately to diagnose such undesired behaviour by an automatic segmenter. B-IR metrics should be used to look for boundary type confusion and boundary type frequency issues (e.g, B-precision, B-recall, etc.). The comparison method selected in this work should be used to quantify how well boundaries were positioned (i.e., the similarity of segmentations).

Upper and lower bounds upon performance No error counts or mean comparison method values are truly useful when comparing automatic segmenters without context such as the upper and lower bounds to performance that can be obtained. A lower bound can easily be established by comparing a baseline (e.g., a random segmenter) in the same manner as the automatic segmenters, but an upper bound is more difficult.

An upper bound can be established by choosing the manual coder who has the greatest pairwise similarity with every other manual coder. This coder can then be added as if it was an automatic coder in a comparison as described previously. In this scenario, the manual coder fills the place of the automatic coder, and is compared to each other coder and against itself. Alternatively, a leave-one-out approach can be taken where one manual coder is randomly selected as the upper bound and is evaluated against only the remaining coders. These methods both present reasonable approximations of the highest

possible performance that can be achieved upon a specific data set (though it is not the absolute highest performance). This, however, work arbitrarily prefers the former type of upper bound (i.e., not leave-one-out).

Chapter 5

Experiments, Results, and Discussions

In the previous chapters, a variety of concepts have been presented that all relate to the evaluation of text segmentation. First, a review of the state-of-the-art evaluation methods and methodologies was presented (Chapter 3). Then a set of new methodologies was proposed (Chapter 4), which can be separated into three broad categories:

- Comparison methods for linear segmentation (S, B_a, B_b, B_c, and $1-WD$);
- Inter-coder agreement coefficients for segmentation (π^*_φ, κ^*_φ); and
- Methodologies for comparing automatic segmenters (pairwise means and statistical hypothesis tests handling multiple codings and multiple documents).

In this chapter, each of these three concepts are used to evaluate text segmentation in comparison with their counterparts (if they exist). Specifically, the segmentation evaluation methods proposed herein, *boundary similarity* (variants B_a, B_b, and B_c) and *segmentation similarity*, are evaluated in comparison to WindowDiff.[1] Proposals for calculating adapted inter-coder agreement coefficients are demonstrated upon natural data and tested upon artificial data. The methodologies for comparing automatic segmenters are then applied, demonstrating their ability to discern between random baselines and automatic segmenters of varying performance.

The experiments in this chapter are broken up into three main sections which revolve around attempting to answer each one of these three questions:

[1]Demonstrations of P_k are omitted because WindowDiff is an improvement over P_k (Pevzner and Hearst, 2002).

1. How can we best compare two arbitrary segmentations?
2. How can we determine the reliability of manual segmentations and replicability of how they are collected?
3. How can we select the best automatic segmenter for a task?

To answer each broad research question, a series of smaller questions are asked, and for each a series of small experiments have been designed and performed to elicit answers. For each smaller question, the methodology, experiment, results, analysis, and discussion are presented all within this chapter with an overall discussion left to the end of each sub section. This structure is used because this thesis aims to identify the best methodology to use when evaluating text segmentation. A meta-evaluation of an evaluation methodology requires not just one major experiment, but many small experiments to demonstrate the viability and reliability of each step of the proposed evaluation methodology.

The meta-evaluation performed in this chapter is, in many cases, subjective. This is a severe weakness, but a variety of measures have been taken to quantify and limit this subjectivity. For each experiment, a hypothesis is defined. This hypothesis is formed by the author of this work prior to performing the experiments. Each hypothesis is designed to reflect how an ideal segmentation comparison method should react to the experiment performed. It is these hypotheses wherein most subjectivity is introduced into this meta-evaluation. The validity of this meta evaluation depends, then, upon how reasonable each hypothesis is to the reader. The comparison method for which the most hypotheses are true is then, arguably, the best of those tested. This meta-evaluation aims to ascertain which comparison the burden of evidence favours, and weighs this in discussions that follow each series of experiments.

This work asks and answers its questions in a specific order. To trust in the methodology proposed for selecting an automatic segmenter, one must trust both the evaluation metric (or measure/comparison method) used and the data upon which it is evaluated. Trust in the data used for evaluation requires trust in the validity of the data (do coders agree?) and the manner in which they were collected (is the coding methodology reliable?). Data and coding scheme reliability can be evaluated using inter-coder agreement coefficients. To trust the inter-coder agreement coefficients used herein, there must also be trust in the underlying segmentation comparison method used (if one is used). The experiments in this chapter are then performed in the order of precedence required to trust in the entire chain of tools used:

1. Comparison methods;
2. Inter-coder agreement coefficients; and then
3. Statistical hypothesis testing to evaluate a set of automatic segmenters.

Because of the dependencies of each question upon the others, at the end of each section, the answer to the research question with which the section is concerned is discussed. This discussion enumerates the evidence gleaned from the sub-questions (i.e., experiments), and an overall answer is given. A summary of the entire picture is then left for Chapter 7 to expound upon.

5.1 Linear Segmentation Comparison Methods

The most popular linear segmentation comparison method in use today is WindowDiff (WD). WD replaced P_k as the dominant method of comparing two arbitrary segmentations, and compelling arguments were made by Pevzner and Hearst (2002) for its usage. It replaced P_k because of its ability to account for near misses yet equally penalize both FPs and FNs (although as Georgescul et al. 2006, p. 148 points out, not entirely equally). It is also a more consistent method (i.e., its not as sensitive to variation in internal segment size as P_k and returns more consistent values when presented with the same types of errors in varying scenarios).

WD is not, however, without fault. WD is not consistent when presented with the same number and types of errors at varying positions. It also fails to properly account for errors at the beginning and end of a segmentation (a problem for which Lamprier et al. 2007 offered solutions). It is also inconsistently applied due to a lack of a concrete definition of what form of rounding is to be used when calculating window size, and it has also been criticized as not being an intuitive value.

In this section, a variety of experiments are performed which illustrate the deficiencies of WD, and compare the performance of the comparison method proposed herein, *boundary similarity* (variants B_a, B_b, and B_c) and *segmentation similarity* (S), against WD. Any deficiencies in WD are assumed to be present in P_k. To answer the overarching question of how can we best compare two arbitrary segmentations, a number of questions are asked for which experiments are conducted to answer:

1. How does each comparison method react to extreme comparisons (e.g., completely similar or fully dissimilar segmentations);

2. How does each comparison method react to various errors (full and near misses);
3. How does each comparison method rank, in terms of similarity, a set of hypothetical artificial segmentations in comparison to one reference segmentation?
4. How does each comparison method react to the linear increase of error types (full and near misses)?
5. How does each comparison method react to the linear increase in the size of a document with a constant amount of error?
6. How is the consistency of each comparison method affected by changes in internal-segment-size variance?

For each of these questions, one or more experiments are conducted to elicit an answer, with a discussion and conclusion following each result. Because all of the comparison methods proposed herein are reward-based, WD is reported as $1-WD$ to be more easily compared.

5.1.1 Experiments

5.1.1.1 Extrema

Question How does each comparison method react to extreme comparisons?

Introduction This experiment tests the responses of segmentation comparison methods when comparing:

1. Identical segmentations containing few boundaries;
2. One fully segmented versus one completely unsegmented segmentation; and
3. Two segmentations lacking boundaries.

Hypothesis If properly implemented, each comparison method should return a value for the three described scenarios of: 1) 1.0; 2) 0.0; and 3) 1.0.

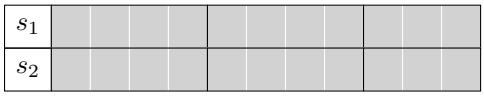
Procedure

1. Each pair of segmentations described is arbitrarily created using a single-boundary type and is 11 units long, giving 10 PBs (arbitrarily).

2. Arbitrarily, the segmentation labelled s_1 in each pair is considered the reference segmentation for $1-WD$.
3. Each of the segmentation comparison methods (S, B_a, B_b, B_c, $1-WD$) is used to compare each of the pairs of segmentations.

Results

1. When comparing identical segmentations as shown in Figure 5.1a each comparison method responds as reported in Figure 5.1b.



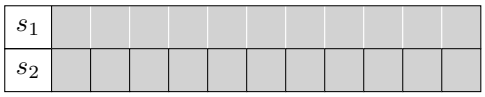
(a) Identical Segmentations

$c(s_1,s_2)$	S	B_a	B_b	B_c	$1-WD$
	1.0	1.0	1.0	1.0	1.0
					$k=2$

(b) Comparison Method Values

Figure 5.1: Comparison Method Values for Identical Segmentations

2. When comparing a segmentation with no boundaries to one that is fully segmented as shown in Figure 5.2a each comparison method responds as reported in Figure 5.2b.



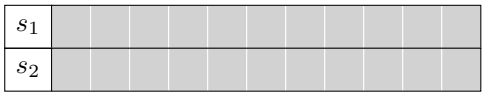
(a) All or No Boundaries Segmentations

$c(s_1,s_2)$	S	B_a	B_b	B_c	$1-WD$
	0.0	0.0	0.0	0.0	0.0
					$k=6$

(b) Comparison Method Values

Figure 5.2: Comparison Method Values for Fully and Fully Un-Segmented Segmentations

3. When comparing two segmentations with no boundaries as shown in Figure 5.3a each comparison method responds as reported in Figure 5.3b.



(a) No Boundaries Segmentations

$c(s_1,s_2)$	S	B_a	B_b	B_c	$1-WD$
	1.0	1.0	1.0	1.0	1.0
					$k=6$

(b) Comparison Method Values

Figure 5.3: Comparison Method Values for Fully and Fully Unsegmented Segmentations

Discussion & Conclusion In the case where each comparison method compared:

1. Identical segmentations, each indicated complete similarity (1), indicating that each comparison method properly recognized that no errors were present;

2. A segmentation with no boundaries to one that was fully segmented, each reported complete dissimilarity (0), indicating that each comparison method properly recognized a maximum number of errors was present; and

3. Two segmentations with no boundaries, each indicated complete similarity (1), indicating that each comparison method properly recognized that no errors were present (despite the lack of boundary placement). Failure of this test would have indicated that the denominators used by B_a, B_b and B_c may have been improperly implemented (causing a division by zero).

The hypothesis is then correct; each method appears to have had its formulation implemented correctly and each reacts as one would expect, and is thus an ideal comparison method in each of these scenarios.

5.1.1.2 Full Miss

Question How does each comparison method react to full miss errors?

Introduction This experiment tests the responses of segmentation comparison methods when comparing two segmentations containing one match (so that a value of 0 is not reported) and one full miss (so that a value of 1 is not reported, and to test how each reacts to one unambiguous full miss). The comparison methods tested should reward the match, penalize the full miss, and report error reflecting the presence of these two events.

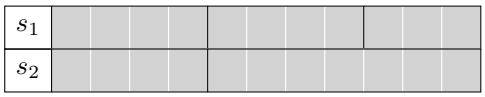
Hypothesis Each comparison method will report a value around 0.5, given that there exists one match and one full miss, representing 1 error out of 2 B/PB placements.

Procedure

1. A pair of segmentations is arbitrarily created using a single-boundary type and is 11 units long, giving 10 PBs, with s_1 having two boundaries placed within, and s_2 has only one that matches the first boundary in s_1.

2. Arbitrarily, the segmentation labelled s_1 in each pair is considered the reference segmentation for $1-WD$.
3. Each of the segmentation comparison methods (S, B_a, B_b, B_c, $1-WD$) is used to compare each of the pairs of segmentations.

Results When comparing segmentations as shown in Figure 5.4a each comparison method responds as reported in Figure 5.4b.



(a) Segmentations with a Full Miss

$c(s_1,s_2)$	S	B_a	B_b	B_c	$1-WD$
	0.9	0.7	0.5	0.5	0.6
					$k = 2$

(b) Comparison Method Values

Figure 5.4: Comparison Method Values for Segmentations with a Full Miss

Discussion The hypothesis was:

1. *Incorrect* for S; it appears to show that the segmentations are largely similar because of the large sections without boundaries and because only one boundary was incorrect out of all 10 potential boundary pairs—this produces a value far above 0.5;

2. *Partially correct* for B_a because it was closer to 0.5 than S; it appears to show that of the boundaries placed in both s_1 and s_2 that $\sim {}^1/_4$ boundaries should have been placed (from s_1's perspective), but were not.

3. *Correct* for B_b; it appears to show that from the perspective of either segmentation, one pair of boundaries match and another boundary is left alone that could have formed a pair, leaving only ${}^1/_2$ pairs in error;

4. *Correct* for B_c; it performed identically to B_b; and

5. *Partially correct* for $1-WD$ because it is close to 0.5; it appears to show that 60% of the windows while comparing these segmentations were not in error, and appears to be between both B_a and B_b/B_c in the interpretation of the similarity between these segmentations.

Conclusion If error should be modelled as mismatched segments, then S appears to provide a representative value of the similarity between the segmentations in Figure 5.4a. If error should instead be modelled as misplaced boundaries, then B_b or B_c appears to provide a representative value—with $1-WD$ following as a close second—of the similarity between the segmentations in Figure 5.4a.

B_b, B_c, and $1-WD$ are the ideal comparison methods in this scenario for evaluating error modelled by mismatched boundaries.

5.1.1.3 Near Miss

Question How does each comparison method react to near miss errors?

Introduction This experiment tests the responses of segmentation comparison methods when comparing two segmentations containing one match (so that a value of 0 is not reported) and one near miss (so that a value of 1 is not reported, and to test how each comparison method reacts to one unambiguous near miss). The comparison methods tested should reward the match and near miss, but still discern between the two by reporting error reflecting the presence of these two events.

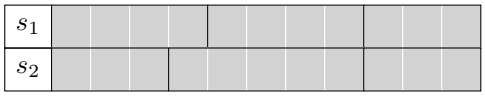
Hypothesis Each comparison method will report a value far above 0.5 but below 1.0 because there exists one match and one near miss. This would be considered as 1 error out of 2 B/PB placements, but not as severe an error as a full miss (which could be considered in this scenario as a similarity of approximately 0.5 because at most half of the boundary pairs are in error). Additionally, the value reported by each comparison method should be higher than their values reported in the full miss experiment (§ 5.1.1.2).

Procedure

1. A pair of segmentations is arbitrarily created using a single-boundary type and is 11 units long, giving 10 PBs, with both segmentations having two boundaries placed within:
    - One pair in s_1 that is off by one potential boundary position in comparison to a boundary in s_2; and
    - One matching pair of boundaries placed sufficiently far away from the nearly missed pair of boundaries to not be considered a near miss.

2. Arbitrarily, the segmentation labelled s_1 in each pair is considered the reference segmentation for $1-WD$.
3. Each of the segmentation comparison methods (S, B_a, B_b, B_c, $1-WD$) is used to compare each of the pairs of segmentations.

Results When comparing segmentations as shown in Figure 5.5a each comparison method responds as reported in Figure 5.5b.



(a) Segmentations with a Near Miss

$c(s_1,s_2)$	S	B_a	B_b	B_c	$1-WD$
	0.9	0.9	0.9	1.0	0.8
					$k = 2$

(b) Comparison Method Values

Figure 5.5: Comparison Method Values for Segmentations with a Near Miss

Discussion The hypothesis was:

1. *Incorrect* for S; it shows that the segmentations are nearly identical except for the near miss, but it failed to produce a value that differed from the full miss experiment;
2. *Correct* for B_a; it shows that the segmentations are nearly identical except for the near miss;
3. *Correct* for B_b; it performed identically to B_a;
4. *Incorrect* for B_c; it indicated complete similarity, despite there being a minor error present; and
5. *Correct* for $1-WD$ because it is far above 0.5; it appears to show that 80% of the windows while comparing these segmentations were not in error and penalized the error $k = 2$ times, which is perhaps undesirably harsh in this scenario.

Conclusion S, B_a and B_b appear to all indicate that a minor error has occurred, which is desirable from both a mismatched segments and mismatched boundaries error modelling perspective. S, unfortunately, failed to report a value that differed from the full miss experiment, meaning that it cannot discern between the two scenarios. $1-WD$ appears to indicate that a minor error has occurred as well, but is more harsh in its assessment than

the other comparison methods, which is a less than desirable behaviour in this scenario. B_c indicates that no errors have occurred, which is an undesired behaviour, indicating that this method may lack the ability to identify near misses in some situations (or is to generous in its assessment).

B_a and B_b are the ideal comparison methods in this scenario for evaluating error modelled by mismatched boundaries, while $1-WD$ is acceptable.

5.1.1.4 Simultaneous Full and Near Misses

Question How does each comparison method react to simultaneous full and near miss errors?

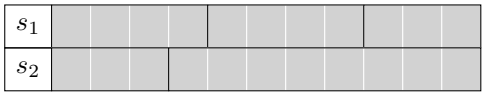
Introduction This experiment tests the responses of segmentation comparison methods when comparing two segmentations containing one full miss and one near miss (so that a value of 0 is not reported). The comparison methods tested should reward the near miss, penalize the full miss, and report error reflecting the presence of these two events.

Hypothesis Each comparison method will report a value below 0.5 but above 0, given that there exists one near miss (affording some reward) and one full miss (offering no reward). This could be interpreted as 1 major error and one minor error out of 2 B/PB placements.

Procedure

1. A pair of segmentations is arbitrarily created using a single-boundary type and is 11 units long, giving 10 PBs, with one segmentation having two boundaries placed within and the other only one boundary, resulting in:
   - One boundary in s_1 that is off by one potential boundary position in comparison to a boundary in s_2; and
   - One additional boundary placed in s_1 sufficiently far away from the nearly missed pair of boundaries to be considered a full miss;
2. Arbitrarily, the segmentation labelled s_1 in each pair is considered the reference segmentation for $1-WD$.
3. Each of the segmentation comparison methods (S, B_a, B_b, B_c, $1-WD$) is used to compare each of the pairs of segmentations.

Results When comparing segmentations as shown in Figure 5.6a each comparison method responds as reported in Figure 5.6b.



(a) Segmentations with a Full & Near Miss

	S	B_a	B_b	B_c	$1-WD$
$c(s_1,s_2)$	0.8	0.6	0.4	0.6	0.4
					$k=2$

(b) Comparison Method Values

Figure 5.6: Comparison Method Values for Segmentations with both a Full & Near Miss

Discussion The hypothesis was:

1. *Incorrect* for S; it appears to show that set of all boundary positions is largely similar between the two segmentations are similar because of the large sections without boundaries;

2. *Incorrect* for B_a; it indicated that there was a similarity greater than 0.5 and either overly rewarded the near miss or did not fully penalize the full miss;

3. *Correct* for B_b; it penalized the full miss and did not fully reward the near miss;

4. *Incorrect* for B_c; it performed identically to B_a; and

5. *Correct* for $1-WD$ because it is below 0.5; it appears to show that 40% of the windows while comparing these segmentations were not in error, thus penalizing the full miss and not fully rewarding the near miss.

Conclusion S appears to indicate that little error occurred, whereas B_a and B_c appear to not fully penalize the full miss, or over-generously reward the near miss. If error should be modelled as mismatched segments, then S appears to provide a representative value of the similarity between the segmentations in Figure 5.6a. If error should instead be modelled as misplaced boundaries, then B_b and $1-WD$ appear to be more representative value of the similarity between the segmentations in Figure 5.6a because both report less than half of the pairs as being in full agreement.

B_b and $1-WD$ are the ideal comparison methods in this scenario for evaluating error modelled by mismatched boundaries.

5.1.1.5 5 Hypothetical Automatic Segmenters

Question How does each comparison method rank, in terms of similarity, a set of hypothetical artificial segmentations in comparison to one reference segmentation?

Introduction This experiment tests the responses of segmentation comparison methods when comparing 5 hypothetical automatic segmentations to one designated reference as defined in Pevzner and Hearst (2002, p. 9) while proposing WD (reproduced in Figure 5.7). The data is artificial, and the rationale for the ranking solution that the comparison methods are tested against is arbitrary. This means that the hypothesis in this experiment has a weak rationale, and this experiment alone is not able to decide whether a comparison method can produce reasonable rankings of automatic segmenters. The result of this experiment should then be weighted less heavily in one's mind. The ability for comparison methods to properly order automatic segmenters during an evaluation is better evaluated later in Section 5.3.

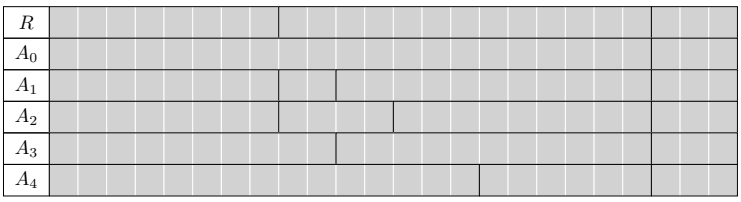
As described by Pevzner and Hearst (2002, pp. 8–9), "algorithm A_4 is arguably the worst of the examples," since it contains two full misses. "Algorithms A_0 and A_2 follow", both containing at least one full miss.[2] "Comparing algorithms A_1 and A_3, algorithm A_3 is arguably better because it recognizes that there is only one boundary present rather than two." Following this logic, the order of the automatic segmenters, from best to worst, is: A_3, A_1, A_2, A_0, A_4.

Hypothesis Each comparison method will report values that order automatic segmenters, from best to worst, following the other described in Pevzner and Hearst (2002, pp. 8–9): A_3, A_1, A_2, A_0, A_4.

Results When comparing the five automatic segmentations and the reference from Pevzner and Hearst (2002, p. 9), each comparison method responds as reported in Figure 5.7. The row in the table shown corresponds to the artificial segmentation in the same row of the figure to its left.

The resulting order for each of the automatic segmenters, from best to worst, provided by each comparison method is shown in Table 5.1.

[2]Pevzner and Hearst (2002, p. 9) indicate that since A_0 contains a false negative, it is penalized more than A_2 is for its false positive when being compared using P_k, which suggests that they assume A_2 is better than A_0, but they do not explicitly state a preference.



	S	B_a	B_b	B_c	$1-WD$
A_0	0.96	0.67	0.50	0.50	0.67
A_1	0.96	0.80	0.67	0.67	0.86
A_2	0.96	0.80	0.67	0.67	0.86
A_3	0.96	0.90	0.80	0.93	0.80
A_4	0.91	0.50	0.33	0.33	0.60

Figure 5.7: One reference (R) and five artificially generated hypothetical segmentations (A_0—A_4; adapted from Pevzner and Hearst (2002, p. 9)) and compared using the comparison methods proposed herein

P & H 2002	A_3, A_1, A_2, A_0, A_4
S	(A_0, A_1, A_2, A_3), A_4
B_a	A_3, (A_1, A_2), A_0, A_4
B_b	A_3, (A_1, A_2), A_0, A_4
B_c	A_3, (A_1, A_2), A_0, A_4
$1-WD$	(A_1, A_2), A_3, A_0, A_4

Table 5.1: Automatic segmenters from Figure 5.7 ordered by comparison methods from most to least similar to the reference segmentation R compared to the exemplar order proposed by Pevzner and Hearst (2002, pp. 8–9), where brackets indicate ties in similarity (i.e., there is no order for that subsequence)

Discussion The hypothesis was:

1. *Incorrect* for S; it appears to show that set of all boundary positions is largely similar between all of the segmentations because of the large sections without boundaries, and is only able to claim that A_4 is the worst;

2. *Mostly correct* for B_a because except for showing no differentiation in the order between the subsequence A_1, A_2, the rest of the ordering matches the hypothesis;

3. *Mostly correct* for B_b; it performed identically to B_a;

4. *Mostly correct* for B_c; it performed identically to B_a; and

5. *Partially correct* for $1-WD$ because identified A_0 and A_4, but erroneously did not identify A_3 as the best segmentation, swapping it for an orderless A_1 and A_2 subsequence.

Conclusion If error should be modelled as mismatched segments, then S appears to provide representative values of the similarity that one could interpret is exemplified

in the segmentations presented. If error should be modelled as misplaced boundaries, then B_a, B_b, and B_c appears to provide the correct relative ordering of the automatic segmentations shown, with $1-WD$ following behind.

B_a, B_b, and B_c are the ideal comparison methods in this scenario for evaluating error modelled by mismatched boundaries because they best reproduced the expected order.

5.1.1.6 Increasing Full Misses

Question How does each comparison method react to the linear increase of full miss errors?

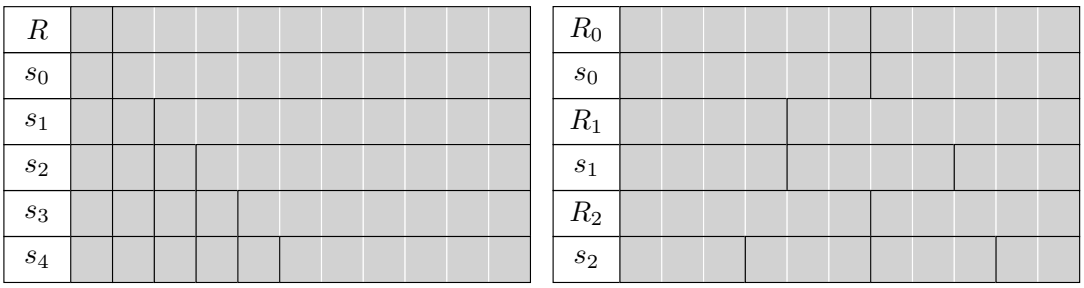
Introduction This experiment tests the responses of segmentation comparison methods when comparing a segmentation that contains one boundary placed within it to series of segmentations where each has more boundaries than the last. This results in one segmentation with one boundary being compared to a series of segmentations that essentially have more and more full misses with each comparison.

There are a multitude of ways to linearly increase the number of full misses, but only two have been selected for use herein—sequential and uniform:

A. *Sequentially* adding full misses from the start of the segmentation to the end (performed by creating a single reference segmentation R that contains one boundary at the beginning of the segmentation and compares this to each s_i, where s_i begins with one matching boundary and gradually adds more boundaries as i increases, as shown in Figure 5.8a);

B. *Uniformly* adding full misses such that the sizes of the internal segments remain approximately equal (performed by creating a reference R_i that contains one boundary placed nearest to the middle of the segmentation while remaining adjacent a boundary in s_i, where s_i begins with one matching boundary in the centre and gradually adds more boundaries as i increases while ensuring that each internal segment is approximately equal in size, as shown in Figure 5.8b).

This experiment will reveal how each comparison method reacts to linearly increasing full misses while controlling for variations in the internal segment size of the non-reference segmentation (s_i). Internal segment sizes will steadily decrease in variance during the sequential pattern, and stay constant during the uniform pattern.



(a) Sequential (b) Uniform

Figure 5.8: Linear increase in full misses in a document D where $\text{pb}(D) = 10$

Hypothesis Each comparison method will report a value of 1 initially, and will decrease in value until they approach (but do not report) 0 at either a linear or exponential rate of decay, and will not vary in output depending upon the pattern used (i.e., the output when using pattern A or B will be identical).

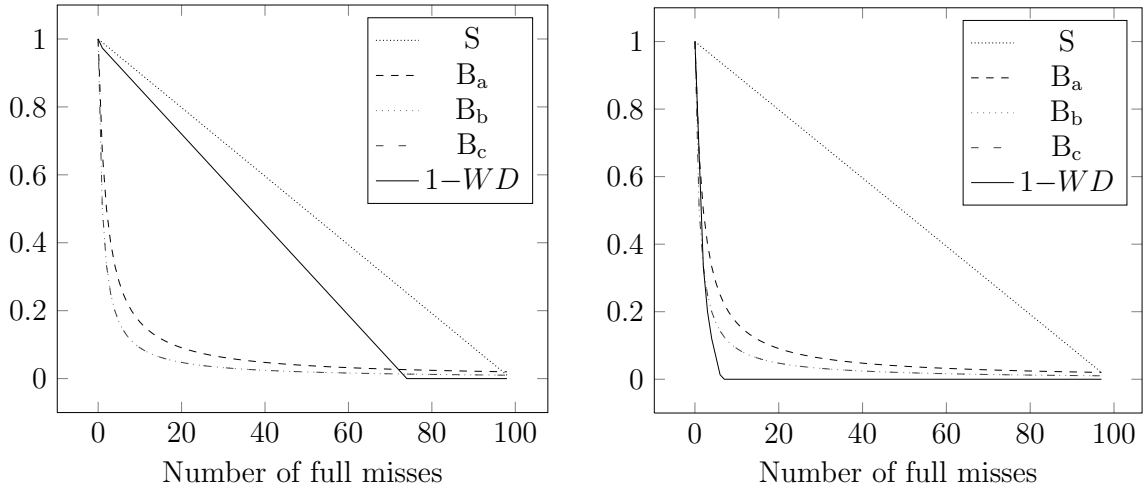
Procedure (Pattern A)

1. A segmentation 100 units in size is created and labelled R, with one boundary placed at the first PB position inside the segmentation;
2. 99 segmentations that are 100 units in size are created and labelled s_i where $i = 1 \ldots 99$, where each segmentation contains contains i boundaries placed at PB position 0 onwards (as in Figure 5.8a);
3. Each of the segmentation comparison methods (S, B_a, B_b, B_c, $1{-}WD$) is used to compare R to each s_i in order from $i = 1 \ldots 99$.

Procedure (Pattern B)

1. 99 segmentations 100 units in size are created and labelled s_i where $i = 1 \ldots 99$, where each segmentation contains contains i boundaries placed at PB positions such that the size of each internal segment is equal (or one segment is off by one) to each other (as in Figure 5.8b);
2. 99 segmentations 100 units in size are created and labelled R_i, with one boundary placed across from a boundary placed in the corresponding s_i as close to the centre of the segmentation as possible such that it is a match (as in Figure 5.8b);
3. Each of the segmentation comparison methods (S, B_a, B_b, B_c, $1{-}WD$) are used to compare each pair of R_i and s_i segmentations in order from $i = 1 \ldots 99$.

Results When comparing segmentations constructed using the procedures detailed previously, each comparison method responds as shown in Figure 5.9.



(a) Sequential

(b) Uniform

Figure 5.9: Linear increase in full misses in a document D where $\mathrm{pb}(D) = 99$

Discussion The hypothesis was:

1. *Correct* for S; it initially reported 1, did not report 0, and decayed linearly to a value approaching zero, and reacted identically to both patterns;

2. *Correct* for B_a; it initially reported 1, did not report 0, and decayed exponentially to a value approaching zero, and reacted identically to both patterns;

3. *Correct* for B_b; it performed identically to B_a except that it decayed at a slightly faster rate; and

4. *Correct* for B_c; it performed identically to B_b; and

5. *Incorrect* for $1-WD$ because it reported 0, only followed a linear decay for $\sim 70\%$ of its decay for pattern A, and reported a different decay for pattern B which also had a decay of one form (exponential, for $\sim 5\%$) and then reported 0 prematurely; it did, however, initially report 1.

Conclusion If error should be modelled as mismatched segments, then S appears to provide an appropriate reaction to the linear increase in error, i.e., a linear decay in observed performance. $1-WD$ also showed this form of decay, but only for $\sim 70\%$ of its decay in pattern A, and instead reported exponential decay for $\sim 5\%$ during pattern B, and it prematurely reported 0 (when in fact complete dissimilarity was never present) in both patterns. $1-WD$ is highly influenced by variance in internal segment size when confronted with equal numbers and types of errors.

If error should be modelled as misplaced boundaries, then B_a, B_b, and B_c appears to provide an appropriate reaction to the proportion of boundaries in error: an exponential decay in observed performance. This reflects the notion that for each iteration, there are $^1/_i$ errors (where $i > 0$) present out of all boundaries placed.

B_a, B_b, and B_c are the ideal comparison methods in this scenario for evaluating error modelled by mismatched boundaries.

5.1.1.7 Increasing Near Miss Distance

Question How does each comparison method react to the linear increase of the severity of a near miss?

Introduction This experiment tests the responses of segmentation comparison methods when comparing two segmentations containing a pair of boundaries which begin as a matching pair and iteratively have the distance between them increased. This has the effect of producing a near miss which at some point becomes a full miss.

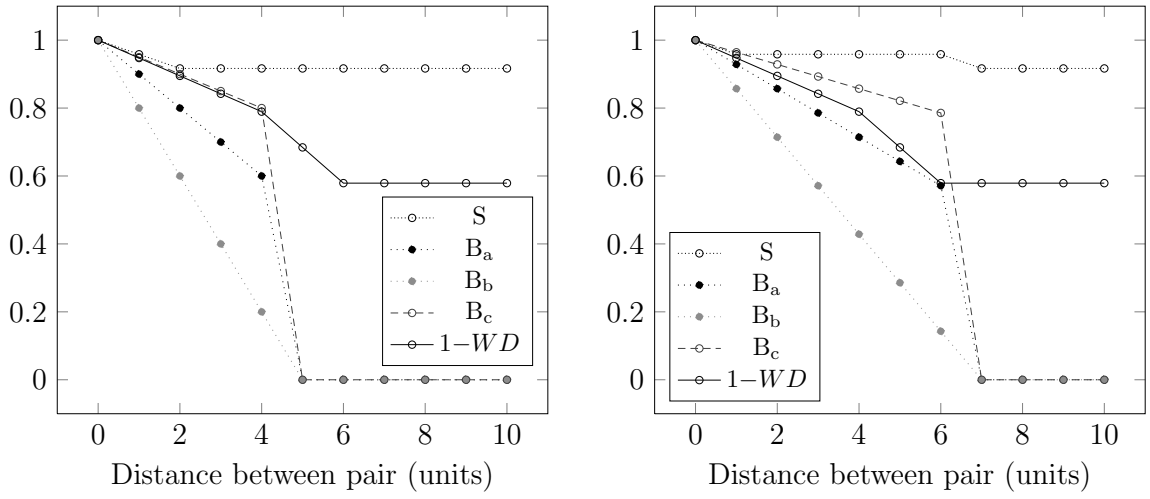
Hypothesis Each comparison method will initially report a value of 1. Next, they will report a decreasing value while the distance between the pair is increased (yet still considered a near miss) until each comparison method considers the pair to be two full misses. At this point, their reported values will level off at a specific value.

Procedure

1. A pair of segmentations is arbitrarily created using a single-boundary type 25 units long, giving 24 PBs.
2. R has one boundary placed at the 2^{nd} PB position.

3. s_i has a boundary at the same position as R for s_0, but moves the boundary to a position i units further to the right than R's boundary for each iteration of $i = 1 \ldots 10$;
4. Each of the segmentation comparison methods (S, B_a, B_b, B_c, $1{-}WD$) is used to compare R to each of the s_i segmentations:
   - With $n_t = 2$ or 5; and
   - With $n_t = 7$ to demonstrate what occurs when n_t is varied.

Results When comparing segmentations constructed using the procedures detailed previously, each comparison method responds as shown in Figure 5.10.



(a) S($n_t = 2$), $B_{a,b,c}$($n_t = 5$), $k = 6$ (b) $n_t = 7$, $k = 6$

Figure 5.10: Linear increasing distance between a pair of boundaries from 0 to 10 units

Discussion The hypothesis was:

1. *Correct* for S; it dropped in value when its maximum transposition spanning distance was encountered ($n = 2$ or 7) and stabilized at a high value;
2. *Correct* for B_a; ($n = 5$ or 7) it performed similarly to S but stabilized at 0;
3. *Correct* for B_b; it performed similarly to B_a except that it linearly decreased from the start to the drop to 0 because it inherently uses transposition scaling by distance (which could be enabled on the other boundary-edit-distance-based comparison methods);

4. *Correct* for B_c; it performed similarly to B_a;

5. *Correct* for $1-WD$; although it should be noted that its definition of what constitutes a near miss is only configurable by adjusting the window size k, which is set according to the reference segmentation's mean internal segment size.

Conclusions All comparison methods evaluated were sensitive to the increase in the distance between the two boundaries in the segmentations. S, B_a, B_b, and B_c have an advantage over $1-WD$, however, because their sensitivity to near misses are all easily configurable. $1-WD$ is not configurable and is solely dependent upon the internal segment sizes of the reference segmentation to vary its sensitivity to near misses via k.

B_a, B_b, and B_c are the ideal comparison methods in this scenario for evaluating error modelled by mismatched boundaries because they all scale error by the severity of the near miss. They report complete error only once the pair is considered a full misses. S an $d1-WD$ do not ever report complete error, and are thus only suitable if error should be modelled instead as mismatched segments.

5.1.1.8 Increasing Document Size

Question How does each comparison method react to the linear increase in the size of a document with a constant amount of error?

Introduction This experiment tests the responses of segmentation comparison methods when comparing two segmentations containing a pair of boundaries which match and one boundary which is an unambiguous full miss when the sizes of the two segmentations are increased linearly. This experiment illustrates the effect that size has upon the interpretation of error by these comparison methods.

Hypothesis Each comparison method will not be affected by the an increase in size, and will instead represent the number of errors committed as a consistent value. This is desirable because a comparison method that scales by document size would be biased towards documents with large sizes—an only occasionally desirable bias.

Procedure

1. For each m, where $m = 4 \ldots 100$

(a) A pair of segmentations are arbitrarily produced containing one matching pair of boundaries at a position one quarter into the segmentations from the beginning and a full miss at the half point (as shown in Figure 5.11) with m units of size;

(b) Each comparison method is used to compare the two segmentations for each m.

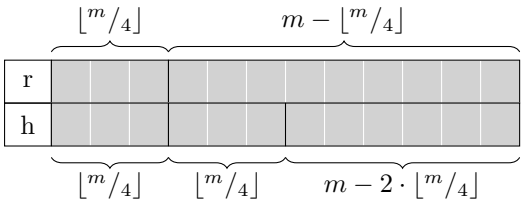


Figure 5.11: Reference and hypothesis segmentations of increasing size m compared

Results When comparing segmentations constructed using the procedures detailed previously, each comparison method responds as shown in Figure 5.12.

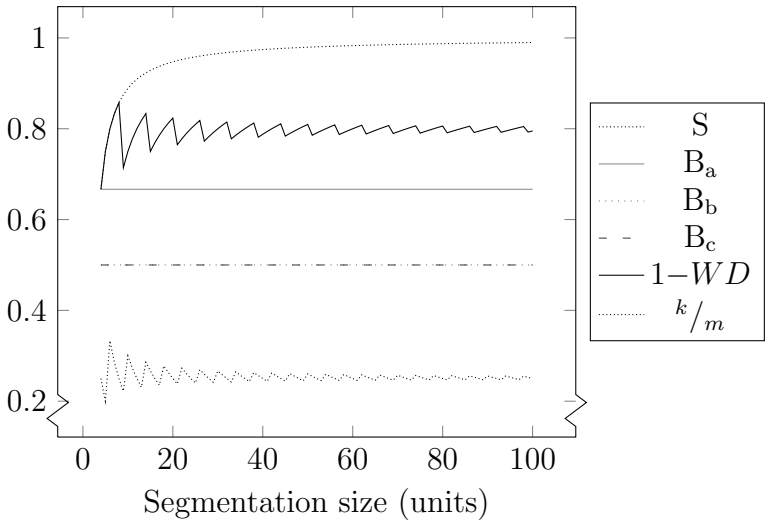


Figure 5.12: Linear increase in size

Discussion The hypothesis was:

1. *Incorrect* for S; it slowly increases in value as size increases;

2. *Correct* for B_a; it remains constant;

3. *Correct* for B_b; it remains constant;

4. *Correct* for B_c; it remains constant;

5. *Incorrect* for $1-WD$; it may eventually converge to a value, but variations in k (plotted as $^k/_m$) lead it to constantly oscillate in value as m is increased, with large variations when m is small.

Conclusions $1-WD$ is unsuitable because it is sensitive to the changes in internal segment sizes introduced by rounding the internal segment sizes down during its calculation of the window size k. This causes it to oscillate as m is increased until it converges at a single value as m approaches infinity. It also did not report a constant value, and is thus biased towards larger documents.

B_a, B_b, and B_c are ideal comparison methods in this scenario for evaluating error modelled by mismatched boundaries because they are not sensitive to segmentation size. S is appropriate only if error is to be modelled by mismatched segments.

5.1.1.9 Comparison Method Consistency

Question How is the consistency of each comparison method affected by changes in internal-segment-size variance?

Introduction To verify that WindowDiff is less sensitive to variations in internal segment sizes in segmentations, Pevzner and Hearst (2002) conducted an experiment that simulated different error types and rates and then measured how WindowDiff (and P_k) reacted to changes in variance of internal segment sizes. To do this, they performed a number of trials, where

> "... a single trial consists of generating a reference segmentation of 1000 segments with some distribution, generating different experimental segmentations of a specific type 100 times, computing the [comparison method] based on the comparison of the reference and the experimental segmentation, and averaging the 100 results. For example, we might generate a reference segmentation R, then generate 100 experimental segmentations that have false negatives with probability 0.5, and then compute the average of their P_k penalties.

> We carried out 10 such trials for each experiment, and averaged the average penalties over these trials."
>
> – Pevzner and Hearst (2002, p. 11)

These trials have been recreated herein using the 4 comparison methods proposed herein and $1-WD$. A more detailed analysis is also performed, which does not rely upon averaged average penalties as used in Pevzner and Hearst (2002), but instead use macro-averages and statistical hypothesis testing.

Hypothesis For the comparisons made by each comparison method made within each scenario, there does not exist a significant difference in mean value between each of the internal segment ranges.

Procedure There are 3 different scenarios in which the hypothesis is tested, including:

1. False Positives (FP) distributed uniformly with $p = 0.5$;
2. False Negatives (FN) distributed uniformly with $p = 0.5$; and
3. False Positives and False Negatives (FP & FN) distributed uniformly with $p = 0.5$.

For each scenario, the hypothesis is tested by performing a number of trials using four different ranges of internal segment sizes, each with a constant mean of 25 such that only the variance changes, including: 1) $(20, 30)$; 2) $(15, 35)$; 3) $(10, 40)$; and 4) $(5, 45)$. To perform the experiment:

1. 10 trials are performed for each of the 5 comparison methods, each scenario, and each internal segment size range, where for each trial:
   (a) A reference segmentation is generated with 1000 random segments of the specified range;
   (b) 100 hypothesis segmentations are generated which have the error probabilities described by the scenario in reference to the reference segmentation;
2. For each comparison method and each scenario, the sets of comparison method values belonging to each internal segment size are analysed by:

(a) Drawing a box plot to illustrate the variation in the comparison method values as a result of the four different internal segment sizes;

(b) Applying the Shapiro-Wilk test (Shapiro and Wilk, 1965) for normality to determine whether their distributions are normal;

(c) Applying Bartlett's test (Snedecor and Cochran, 1989) for equal variances to all of the sets to determine whether their variances are equal;

(d) Applying between-subjects ANOVA (Heiman, 2001) ($\alpha = 0.05$) to determine whether a statistically significant difference between the internal segment sizes exists. If ANOVA's assumptions are violated, then the Kruskal-Wallis H-test (Kruskal and Wallis, 1952) is used in ANOVA's place ($\alpha = 0.05$).

Results For each comparison method, its mean values for each internal segment size are presented in three box plots each. A solid dot (•) in a box plot's caption indicates that statistically significant differences between internal segment size values are present, whereas an empty dot (∘) indicates no significant differences. These three box plots for each comparison method show the results for each scenario (i.e., types of errors present). Less variation (i.e., scenarios where statistically significant differences do not occur) in comparison method means within each scenario is ideal and indicates that the comparison method is less biased by internal segment size variance. The results for the various comparison methods are shown in Figures A.1–A.5.

Discussion The hypothesis was mostly correct for B_a, B_b, and B_c; their means in two of the three scenarios did not statistically significantly differ from each other. Their means did differ in the final scenario (FN & FP $p = 0.5$). In fact, for all comparison methods, the final scenario contained statistically significant differences between mean performances. A gradual increase in performance was noticeable in this final scenario as the internal segment size variance was increased—an effect that was also present in results reported by Pevzner and Hearst (2002, p. 12). This similarity to the original experiment lends support to the assumption that it is an authentic replication, but it is also potentially a flaw that all comparison methods produced effect. Alternatively, this could mean that the procedure for the final scenario is somehow flawed.

The hypothesis was partially correct for S because it did not statistically significantly differ in one of the three scenarios. Unfortunately, the hypothesis for $1-WD$ was completely incorrect because it statistically significantly differed in performance in all three scenarios.

	Hypothesis for each comparison method				
Experiment	S	B_a	B_b	B_c	$1-WD$
§5.1.1.1 Extrema					
Identical segmentations with boundaries	●	●	●	●	●
One fully versus one completely un-segmented	●	●	●	●	●
Two unsegmented segmentations	●	●	●	●	●
§5.1.1.2 Full Miss	○	◌	●	●	◌
§5.1.1.3 Near Miss	○	●	●	○	●
§5.1.1.4 Simultaneous Full and Near Misses	○	○	●	○	●
§5.1.1.5 5 Hypothetical Automatic Segmenters	○	◍	◍	◍	◌
§5.1.1.6 Increasing Full Misses	●	●	●	●	○
§5.1.1.7 Increasing Near Miss Distance	●	●	●	●	●
§5.1.1.8 Increasing Segmentation Size	○	●	●	●	○
§5.1.1.9 Comparison Method Stability	◌	◍	◍	◍	○

● Correct ◍ Mostly correct ◌ Partially correct ○ Incorrect

Table 5.2: Correctness of the hypotheses in each experiment for each comparison method

This experiment demonstrates that B_a, B_b, and B_c are the segmentation comparison methods most resilient to internal segment size variance that were tested. This is an important quality because variations in internal segment size variance can quite easily occur. Even if only one reference segmentation is being used in an evaluation, hypothesis segmentations can have greater or lesser numbers of segments potentially leading to variations in internal segment size variance. $1-WD$, and potentially S, are not ideal for segmentation comparison due to their sensitivity to internal segment size variation.

5.1.2 Discussion

In this section, each experiment was designed to test known failings of current segmentation comparison methods (e.g., missing a partial reward for near misses). These experiments test how they respond to basic scenarios (e.g., extrema) and to the manipulation of one variable (e.g., increasing number or severity of errors, increasing size, or increasing variance in internal segment size). Assuming that each of the hypotheses proposed represents the ideal behaviour expected of a comparison method in all of these experiments, then those methods where the hypothesis was most often and most completely true represent the ideal comparison methods. The validity of these hypotheses is summarized in Table 5.2 with the strength of the correctness of the hypothesis for each experiment shown for each comparison method.

It is clear from Table 5.2 that although no comparison method satisfied the hypotheses perfectly, one clearly demonstrated superior resiliency to negative effects: B_b. By assigning

a numeric value to each of the four levels of satisfaction of the hypotheses (Correct = 3, mostly correct = 2, partially correct = 1, and incorrect = 0) and summing each column in Table 5.2, one can obtain a simple rank for each of the comparison methods. Performing these summations produces Figure 5.13. The ranking of each of the methods is then B_b, followed by B_a and B_c, and then $1-WD$ and S.

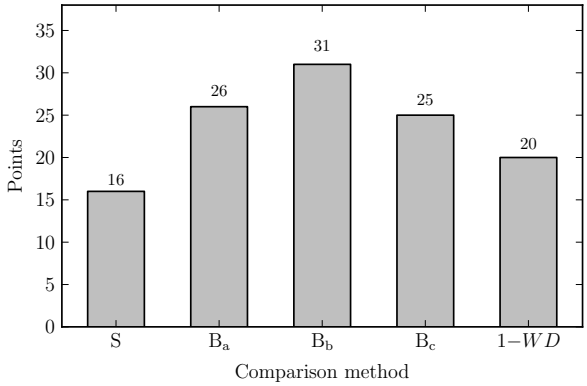


Figure 5.13: Ranking of each comparison method according to points awarded for the degrees to which hypotheses were correct

Such a ranking is not definitive proof that B_b that is the best segmentation comparison method. Instead, the scores and subsequent rankings simply demonstrate the that the evidence gleaned from these experiments favours B_b. Table 5.2 presents a far better argument in favour of B_b: it did not fail to satisfy any hypothesis outright. Each other comparison method has at least one experiment in which the hypothesis was unambiguously untrue. It is for this reason that B_b should be considered the best of the segmentation comparison methods tested herein.

S was shown to be an improvement over WD for segmentation comparison in Fournier and Inkpen (2012), but it is not as useful when comparing boundary placements within segmentations (an important distinction). $1-WD$ was slightly better than S for comparing boundaries, but had very poor handling of full misses (§5.1.1.2, §5.1.1.6), ordering hypothetical segmentations (§5.1.1.5), and was significantly affected by variance in internal segment sizes (§5.1.1.9). Of the three proposed boundary similarities (B_a, B_b, and B_c), B_b performed extremely well, especially in comparison to its siblings in discerning simultaneous full and near misses (§5.1.1.4).

As a result of these experiments, B_b should be considered the best segmentation boundary comparison method of those tested upon this artificial data. Because B_b is also able to compare segmentation pairs individually, it is also the most intuitive choice to

be adapted for use in determining actual agreement for inter-coder agreement statistics. This assertion is tested in the section that follows.

5.2 Inter-Coder Agreement

Inter-coder agreement coefficients can be used to measure the reliability of a coding scheme and annotation instructions using π-based coefficients. They can also be used to determine the suitability of a set of codings as training data using κ-based coefficients. In Section 4.2, both π and κ based coefficients were adapted to be able to handle near misses by using two of the segmentation comparison methods defined herein. In this section, the question of which comparison method is most suitable for use in an adapted coefficient is answered experimentally for the purpose of determining the reliability of manual segmentations and replicability of how they are collected.

Although Section 5.1 established that B_b was an ideal comparison method for segmentation, it has not necessarily been shown that it is ideal for computing agreement. In this section, the methodology for determining adapted π coefficients is demonstrated, and both S and B_b are pitted against each other to determine which is more suitable for characterizing agreement in segmentation. B_a and B_c are both omitted due to their drawbacks and similarity to B_b and for brevity, while S is included due to how different it is from B_b. $1-WD$ has been excluded altogether due to its unsuitability as a comparison method.

In this section, recommendations for choosing a maximum transposition spanning distance are given with demonstrations. Next, agreement and various descriptive statistics of manual codings are obtained for two natural data sets. Finally, a series of artificial data sets are constructed from parameters obtained from descriptive statistics describing the natural data sets. These data sets are used to evaluate which comparison method is more suitable for segmentation agreement: B_b or S.

5.2.1 Agreement upon Manual Segmentations

Before creating artificial segmentations to evaluate which comparison method is best suited for calculating segmentation agreement, it would be useful to understand more about natural segmentations. To that end, this section computes a variety of descriptive statistics upon human topical segmentations of the Stargazers text collected by Hearst (1997) and The Moonstone collected by Kazantseva and Szpakowicz (2012).

The Stargazer text data set is a collection of 7 human topical segmentations of a science magazine article titled "Stargazers look for life" (Baker, 1990) collected by Hearst (1997). The text was one of 12 magazine articles chosen for their length (between 1,800 and 2,500 words) and for having little structural demarcation (Hearst, 1997, p. 53)–presumably to not bias coders—which were segmented at the paragraph level. The coders were each provided with the following instructions:

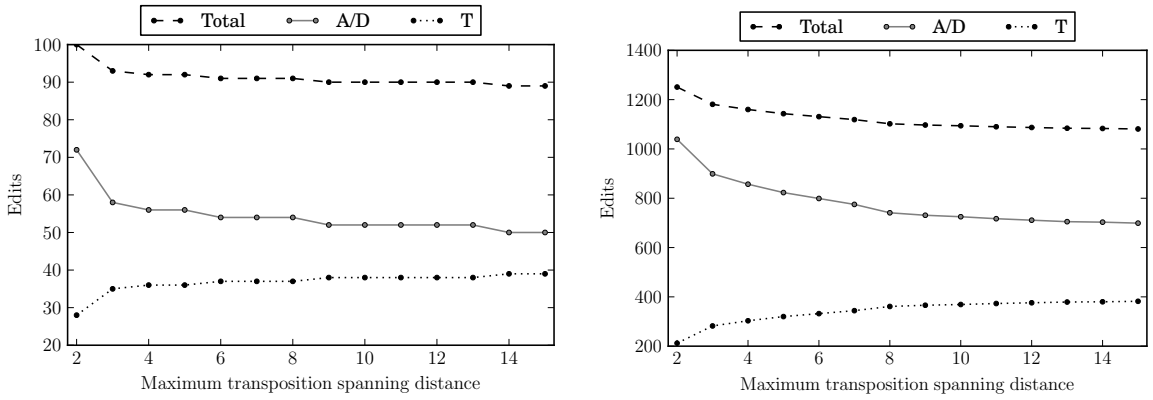
> "You will receive three texts. Mark where the topics seem to change—draw a line between the paragraphs, where any blank line can be considered a paragraph boundary. It's recommended that you read quickly; no need to understand all the nuances. However, you are allowed to go back and look over parts that you've already looked at and change your markings if desired. If on occasion you can't decide between two places, definitely pick one but indicate that you thought the other one was just as appropriate."
>
> – Hearst (1997, p. 53)

Hearst (1997, p. 53) stated that "on the rare occasions in which the subject picked a secondary boundary, only the primary one was retained for evaluation."

The Moonstone data set is a collection of 4–6 human topical segmentations per chapter of 20 chapters from a 19$^{\text{th}}$ century romance novel titled "The Moonstone" (Collins, 1868) collected by Kazantseva and Szpakowicz (2012). The Moonstone contains 23 chapters, 2 of which were coded in a pilot study and 20 of which were coded by 27 undergraduate English students. Coders were divided into 5 groups and each coder was asked to code 4 chapters, with the 27 undergraduates split into three 6-person groups, one 5-person group and one 4-person group. "The [coders] were instructed to read each chapter and split it into episodes—topically continuous spans of text demarcated by the most perceptible shifts of topic in the chapter" and their segmentations were produced at the paragraph level "because even short chapters of most traditional novels are rather lengthy" (Kazantseva and Szpakowicz, 2012, p. 213).

5.2.1.1 Choosing a maximum transposition spanning distance

Maximum transposition spanning distance represents the maximum distance that a boundary can be off by—in textual units—to be considered a near miss as opposed to a full miss. The manual segmentations collected by the two data sets analysed herein all use paragraphs as their textual unit—are very broad size. Because of the large size of



(a) Stargazer (b) Moonstone

Figure 5.14: Edits from pairwise comparisons of codings for $2 \leq n_t \leq 15$

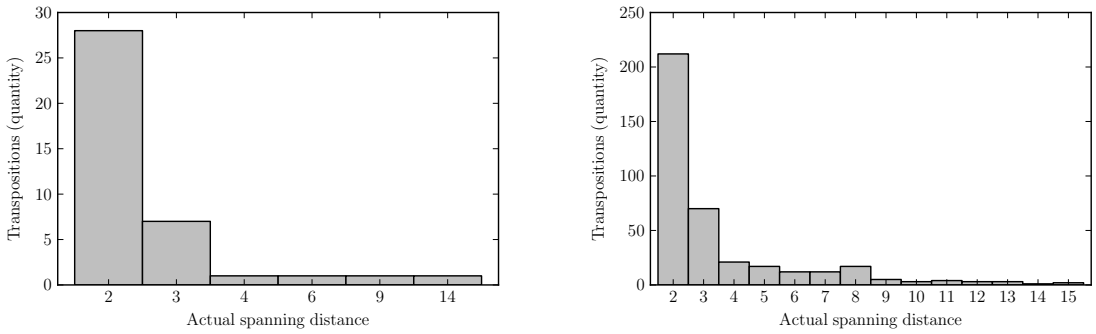
paragraphs for the task of topical segmentation, it is asserted herein that a maximum transposition spanning distance of two, or being off by one paragraph, would be reasonable (i.e., $n_t = 2$).

Although, in the case of the data sets analysed here, the broad size of the textual unit and task informed the decision to arbitrarily set $n_t = 2$, influencing factors may not exist in other data set. The question of what occurs at higher values of n_t is worth investigating. To begin, one relevant question is: how does the number of near misses versus full misses compare to each other in the data sets for varying values of n_t? The answer to this question can be seen in Figures 5.14a–5.14b.

In Figure 5.14, the interpretation by boundary edit distance of what constitutes a transposition (T, i.e., near miss) versus an addition/deletion (AD, i.e., full miss) operation varies as the parameter n_t is adjusted. It is adjusted from the point were it only considers adjacent boundaries ($n_t = 2$) to more distant boundaries ($n_t > 2$). Figures 5.14a–5.14b show a noticeable drop in ADs and rise in Ts at $n_t = 3$, with a slow convergence to a steady level of Ts in both data sets. The gradual decline in the total number of edits occurs because the edit distance is a globally optimal minimum edit distance, and as T operations displace pairs of ADs, the overall number of edits decreases. The rate of change decreases because short-spanning Ts are preferred over long-spanning Ts in boundary edit distance, meaning that few changes occur from n_t to $n_t + 1$ after the point when most errors have been labelled as short-spanning Ts.[3]

It could be assumed that when the steep decline in edits occurs, and just before it

[3]This was trend observed for $2 \leq t_n \leq 100$, but only $2 \leq t_n \leq 15$ is shown for brevity.



(a) Stargazer (b) Moonstone

Figure 5.15: Spanning distance of transpositions for $n_t = 15$

plateaus at $n_t = 3$, may signal a desirable value for n_t. This is because the steep slope between $n_t = 2$ and $n_t = 3$ indicates that a noticeable amount of disagreement was potentially resolved by using $n_t = 3$ to compare boundaries placed by coders. After $n_t = 3$, this same drop in rate is not seen in these data sets.

Figures 5.14a–5.14b give some insight into the relationship between Ts and ADs as n_t increases, but assuming a large n_t such as 15, what would the frequencies of the various spanning lengths of transpositions be? Figures 5.15a–5.15b answer this question. As is evident, even when $n_t = 15$, very few long-spanning transpositions occur, with the bulk of transpositions being at $n_t = 2$, and a moderate amount at $n_t = 3$, and in both data sets an exponential decline in frequency is observed.

From the analyses presented, $n_t = 2$ appears to be a conservative yet reasonable choice of maximum transposition spanning distance. If such conservatism is not required, and no other reasonable rationale exists to designate a specific n_t, then using the methods of analysis shown herein and broadening the interpretation of a near miss to $n_t = 3$ may be supportable. The granularity, or unit, being paragraphs for both data sets motivates the usage of $n_t = 2$ herein when using boundary edit distance based comparison methods for further analyses, which will be used for all subsequent analyses presented in this chapter.

5.2.1.2 Analysing the Stargazers data set

To better inform later experiments using artificially generated segmentation, the analysis of natural data sets is essential to later create a plausible yet controlled artificial data set. In this section, the Stargazers data set (Hearst, 1997) is analysed. A wide variety of descriptive statistics are calculated which help to describe the document and how it was

Mass	21
PBs	20
Boundaries	49
Segments	56
Coders	7

(a) Document and segmentation statistics

PBs	420
Boundaries	294
Additions/deletions	72
Transpositions	28
Matches	125

(b) Between-coder segmentation statistics

P(b)	0.7000
P(a|b = 1)	0.2449
P(t|b = 1)	0.0952
P(m|b = 1)	0.4252

(c) Between-coder event probabilities

Segment size mean	2.6250 ± 1.3438, $n = 56$
Segments per segmentation	8.0000 ± 1.8516, $n = 7$
Boundaries per coder	7.0000 ± 1.8516, $n = 7$

(d) Segmentation statistic means and standard deviations

Table 5.3: Stargazers data set general and boundary-edit-distance-based statistics

coded by 7 coders.

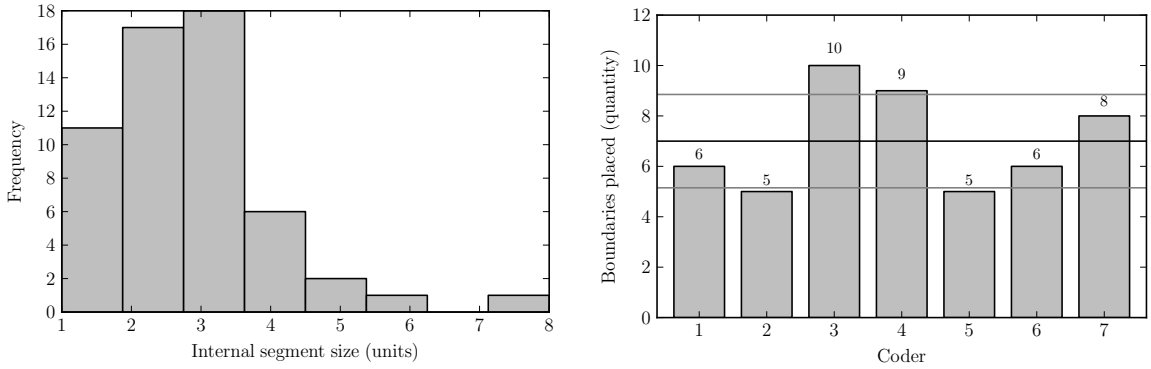
Table 5.3a details the number of paragraphs (i.e., size), the number of potential boundaries (PBs), the total number of boundaries lain by all of the coders, and the number of segments that their boundaries created. As it can be seen, this is a short document with relatively few boundaries.

Table 5.3b details the results of a series of pairwise comparisons between each coder. Between them, the total number of PBs compared, boundaries, matching boundaries, and edits as counted by boundary edit distance ($n_t = 2$) are shown. These counts allow us to estimate the probabilities of certain events.

Table 5.3c uses the counts contained in Table 5.3b to estimate the probability of whether a boundary is placed or not (P(b)) and then the conditional probability that if a boundary is placed, whether it is a missed (a), nearly missed (t), or matching (m) boundary in relation to another segmentation. Such probabilities are later used to generate artificial agreement with similar characteristics as manual segmentations.

Table 5.3d presents mean statistics such as segment size, number of segments, and boundaries per coder. Such values are also later used to generate artificial agreement with similar characteristics as manual segmentations.

The mean statistics presented in Table 5.3d provide some insights into the data set, but they do not reveal anything about the shape of the data they describe. Figure 5.16a provides a histogram of the internal segment sizes, revealing that most of the segments have 2-3 paragraphs and the distribution may be roughly exponential, but not enough data is available to make that determination. The distribution shape is an important



(a) Internal segment size distribution of all coder's segmentations

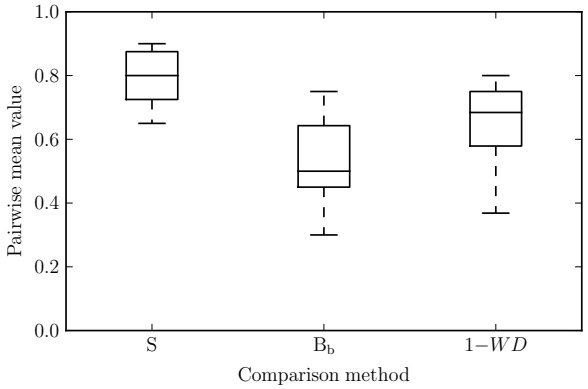
(b) Number of boundaries placed by each coder and the mean and SD

Figure 5.16: Stargazer data set internal segment size distribution and coder boundaries

characteristic when generating artificial segmentations. The lack of clear evidence of the distribution's shape may make this data set less desirable to estimate parameters from in order to create artificial segmentations.

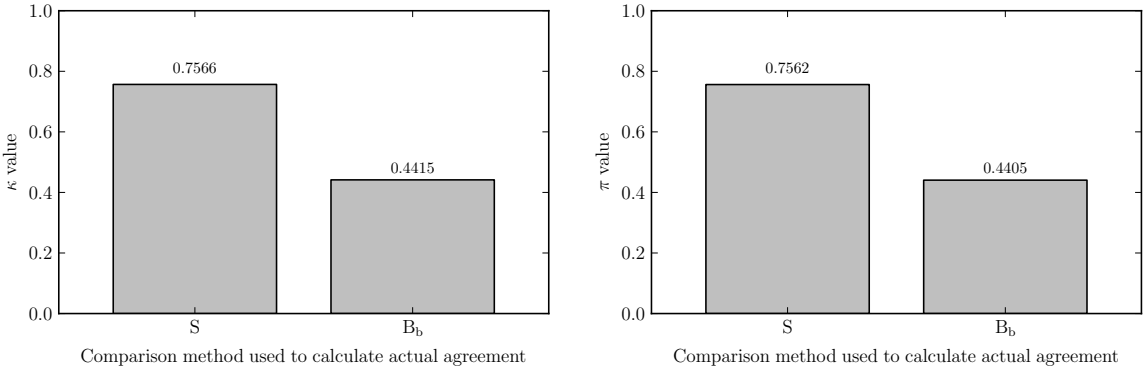
Another important characteristic of the data is coder behaviour—specifically, have any coders coded far more/fewer boundaries than any others? The standard deviation (SD) in Table 5.3d indicates that all coders roughly coded a similar number of boundaries, and Figure 5.16b offers a visual description of the number of boundaries that each coder placed, along with the mean and one SD above and below drawn for reference. Only one of the coders clearly coded outside of one SD of the mean, whereas the rest coded within one SD. This is ideal, and should result in a relatively high degree of agreement between the coders because they all behaved similarly with respect to boundary placement frequency.

Before analysing agreement using π and κ, it would be useful to gain some insight into the actual agreement of the coders—before it is corrected for chance agreement by the inter-coder agreement coefficients. Figure 5.17 shows the pairwise means of three comparison methods (S, $1-WD$, and B_b using $n_t = 2$) as a box plot, showing again that S overestimates actual agreement, B_b provides what is most likely an accurate picture given the comparison method evaluation in Section 5.1, and $1-WD$ roughly agrees with B_b. Actual agreement for this data set appears low according to both B_b and $1-WD$. This means that the full data set may not be ideal for training purposes (instead a subset might be of better use). The coding scheme and instructions may yet have been reliably applied if the chance corrected inter-coder agreement coefficients indicate such.



S	0.7952 ± 0.0710, $n = 21$
B_b	0.5337 ± 0.1181, $n = 21$
$1-WD$	0.6515 ± 0.1161, $n = 21$

Figure 5.17: Stargazer data set pairwise mean comparison method values between coders



(a) κ (b) π

Figure 5.18: Stargazer data set inter-coder agreement coefficients using varying comparison methods for actual agreement

The S and B_b-based inter-coder agreement statistics adapted herein are shown in Figure 5.18. To evaluate the reliability of the coding scheme and instructions applied, we can look at the adapted values of π in Figure 5.18b. The S-based π^* reports rather high agreement, but it may be over-estimating agreement due to the actual agreement presented earlier, and later testing of its unsuitability is performed. The B_b-based π^* reports rather low agreement despite being able to account for near misses, but not entirely unexpected given the difficulty in producing segmentations with high agreement.

5.2.1.3 Analysing the Moonstone data set

In this section, the Moonstone data set (Kazantseva and Szpakowicz, 2012) is analysed. A wide variety of descriptive statistics are calculated which help to describe the document and how it was coded by 27 coders.

Table 5.4a details the number of paragraphs (i.e., size), the number of potential boundaries (PBs), the total number of boundaries lain by all of the coders, and the number of segments that their boundaries created. This is a large data set compared to the Stargazer data set in terms of both document size and boundaries placed.

Table 5.4b details the results of a series of pairwise comparisons between each coder. Between them, the total number of PBs compared, boundaries, matching boundaries, and edits as counted by boundary edit distance ($n_t = 2$) are shown. The values presented here are far higher than those of the Stargazer data set owing both to the greater number of documents (and thus paragraphs) and coders. These counts allow us to estimate the probabilities of certain events.

Table 5.4c uses the counts contained in Table 5.4b to estimate the probability of whether a boundary is placed or not (P(b)) and then the conditional probability that if a boundary is to be placed, whether it is a missed (a), nearly missed (t), or matching (m) boundary in relation to another segmentation. Such probabilities are later used to generate artificial agreement with similar characteristics as manual segmentations.

Table 5.4d presents mean statistics such as chapter size, segment size, number of segments, and boundaries per PB per coder. Boundaries per PB per coder is a necessarily complicated statistic. It is used because not all coders coded all documents, meaning that some coders had a greater chance to add more boundaries because of the differences in chapter lengths. Because of this, it is useful to normalize boundaries per coder by the PBs within the items that a coder coded. Such values are also later used to generate artificial agreement with similar characteristics as manual segmentations.

As shown by the chapter size SD in Table 5.4d, the chapter sizes vary greatly in length. This is further illustrated in Figure 5.19, which shows the size in PBs of each chapter (note that PBs = size − 1) along with the mean and plus/minus one SD. There are 20 chapters in total, ranging from chapter 1–5 and 7–21 inclusively, with chapter 6 left unannotated (the pilot study was performed upon this and one other chapter).

The mean statistics presented in Table 5.4d provide some insights into the data set, but they do not reveal anything about the shape of the data they describe. Figure 5.20a provides a histogram of the internal segment sizes, revealing that most of the segments are approximately less than 10 paragraphs in size, and the distribution is clearly exponential. The distribution shape is an important characteristic when generating artificial segmentations, and the clear evidence of its shape makes this data set highly desirable to perform parameter estimation from.

Another important characteristic of the data is coder behaviour—specifically, have

Mass	1077
PBs	1057
Boundaries	520
Segments	628
Coders	27

(a) Document and segmentation statistics

PBs	12914
Boundaries	2301
Additions/deletions	1039
Transpositions	212
Matches	907

(b) Between-coder segmentation statistics

P(b)	0.1782
P(a|b = 1)	0.4515
P(t|b = 1)	0.0921
P(m|b = 1)	0.3942

(c) Between-coder event probabilities

Chapter mass	53.8500 ± 28.5679, $n = 20$
Segment size	9.2691 ± 9.7667, $n = 628$
Segments per segmentation	5.8148 ± 3.9303, $n = 108$
Boundaries per PB per coder	0.0936 ± 0.0503, $n = 27$

(d) Segmentation statistic means and standard deviations

Table 5.4: Moonstone data set general and boundary-edit-distance-based statistics

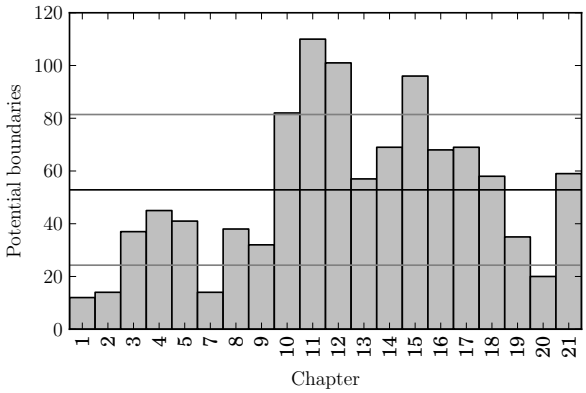
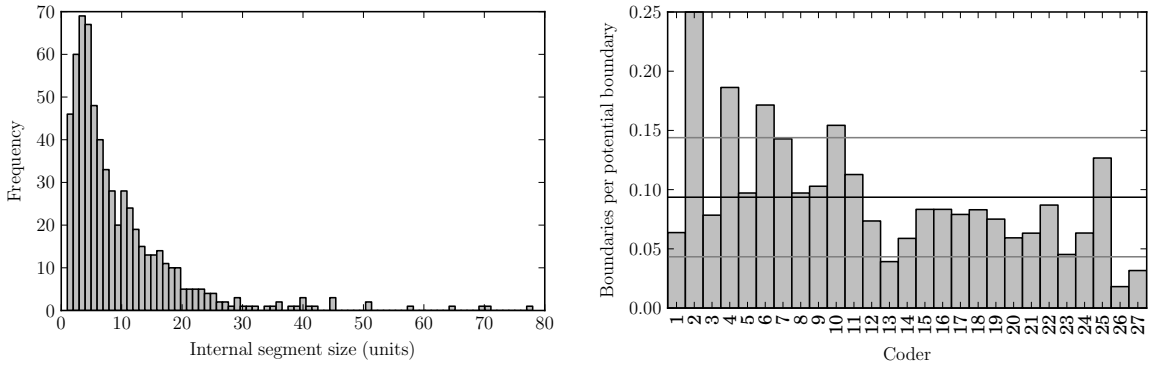


Figure 5.19: Moonstone data set chapter sizes in PBs with mean and standard deviation

any coded far more/less boundaries than any others? The SD in Table 5.4d indicates that all coders roughly coded a similar number of boundaries, and Figure 5.20b offers a visual description of the number of boundaries that each coder placed, along with the mean and one SD above and below drawn for reference. One of the coders coded far more than one SD above the mean, with 3 more coding below and 3 above one SD of the mean. This is not ideal, and should result in relatively low agreement between the coders because many behaved differently with respect to boundary placement frequency.

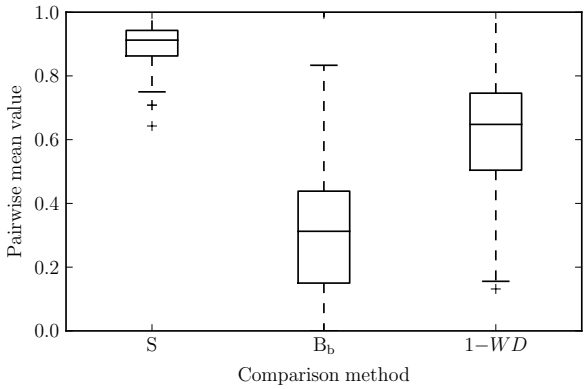
Before analysing agreement using π and κ, it would be useful to gain some insight into the actual agreement of the coders (before it is corrected for chance agreement by the inter-coder agreement coefficients). Figure 5.21 shows the pairwise means of three



(a) Internal segment size distribution of all coder's segmentations

(b) Moonstone data set number of boundaries placed by each coder and the mean and SD

Figure 5.20: Stargazer data set internal segment size distribution and coder boundaries

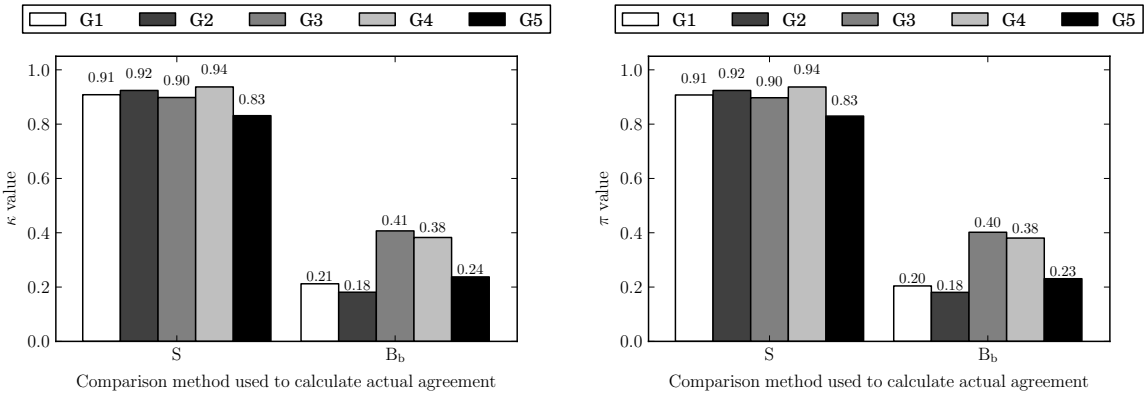


S	0.8996 ± 0.0597, $n = 244$
B_b	0.3202 ± 0.2202, $n = 244$
$1-WD$	0.6199 ± 0.1805, $n = 244$

Figure 5.21: Moonstone data set pairwise mean comparison method values between coders

comparison methods (S, $1-WD$, and B_b using $n_t = 2$) as a box plot. This box plot again shows that S overestimates actual agreement and B_b provides what is most likely an accurate picture given the comparison method evaluation in Section 5.1, while $1-WD$ disagrees with B_b. Actual agreement for this data set appears very low according to both B_b, which means that the full data set may not be ideal for training purposes (instead a subset might be of better use as the high SD reveals). The coding scheme and instructions may yet have been reliably applied if the chance corrected inter-coder agreement coefficients produce sufficiently high values.

The S and B_b-based inter-coder agreement statistics adapted herein are shown in Figure 5.22, where groups 1–5 are comprised of coders as detailed in Table 5.5. In this figure, agreement is broken down into the 5 groups of coders because they coded separate



(a) κ (b) π

Figure 5.22: Moonstone data set inter-coder agreement coefficients using varying comparison methods for actual agreement

Group	Coders
G1	11, 12, 13, 14, 15, 16
G2	23, 24, 25, 26, 27
G3	5, 6, 7, 8, 9, 10
G4	17, 18, 19, 20, 21, 22
G5	1, 2, 3, 4

Table 5.5: Moonstone data set coder groups

sets of chapters.[4] To evaluate the reliability of the coding scheme and instructions applied, we can look at the adapted values of π in Figure 5.22b. The S-based π^* reports rather high agreement, but it may be over-estimating agreement due to the actual agreement presented earlier, and later testing of its unsuitability is performed. The B_b-based π^* reports very low agreement despite being able to account for near misses, but not entirely unexpected given the historical difficulty in producing segmentations with high agreement, the differences in the number of boundaries placed by each coder per PB, and perhaps the inherent difficulty of the task. Novels are most likely very difficult to consistently segment due to the wide variety of factors involved in choosing a boundary. Alternatively, perhaps the instructions gave far too much freedom to the coders, and they all chose different subjective definitions of what "the most perceptible shifts of topic in [a] chapter" are (Kazantseva and Szpakowicz, 2012, p. 213).

To search for subsets where agreement may be higher amongst coders or documents,

[4]Presenting an average of these values is unsuitable because each group has coded text with varying degrees of length.

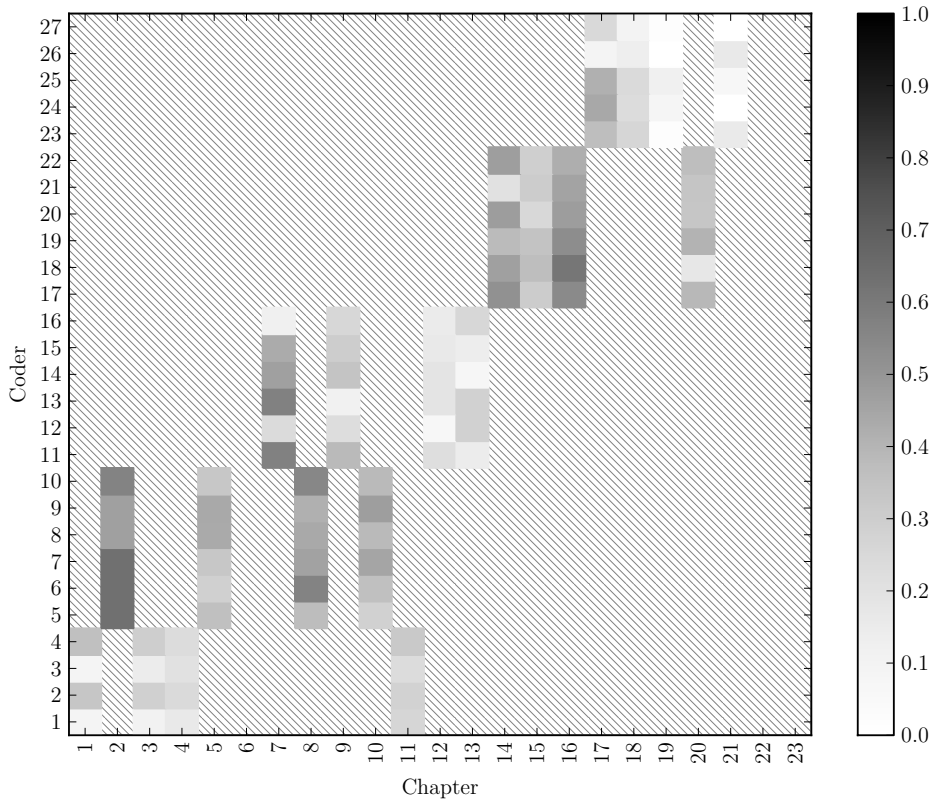
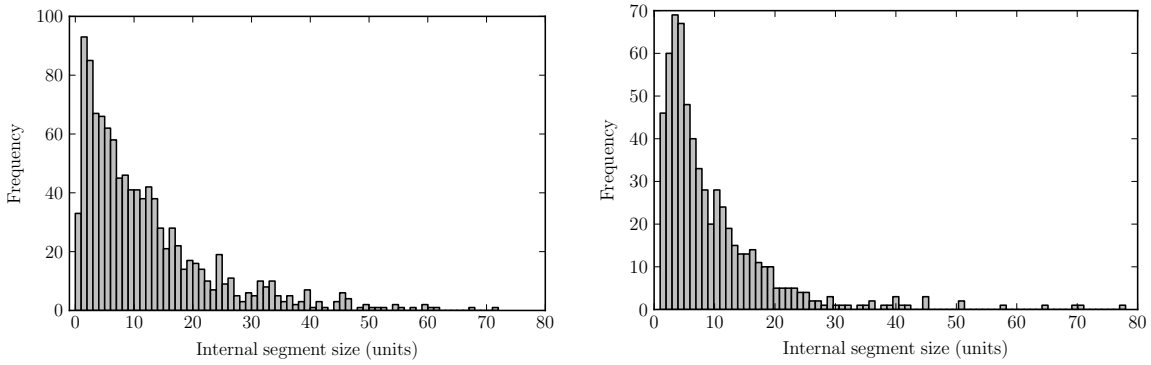


Figure 5.23: Moonstone data set heat map of pairwise B_b per coder per item

a heat map of B_b values was drawn which shows pairwise comparisons between coders within groups for each chapter where darker blocks indicate higher actual agreement. From this, chapter 2 appears to have higher than average actual agreement, as do select coders with their peers in chapters 7, 8, 14, and 16. Removal of the low actual-agreement coders from these chapters may produce subsets of data for chapters with high enough actual agreement to serve as suitable training and evaluation data.

5.2.2 Agreement upon Automatic Segmentations

To determine which comparison method is best suited for computing actual agreement in an inter-coder agreement coefficient—either S or B_b, this section details a series of experiments which were performed to illustrate both the abilities and deficiencies of both adaptations. To reveal these experiments, artificial segmentations are created which mimic



(a) Sample artificial distribution (b) Distribution of the Moonstone data set

Figure 5.24: Artificial versus natural internal segment size distributions

the properties of natural data but control and vary certain properties. Properties varied include the types of error, the probability of boundary placement, and the probability of near and full misses.

The data set from which parameters have been estimated is the Moonstone data set collected by Kazantseva and Szpakowicz (2012) and detailed in Section 5.2.1. Artificial segmentations are created with a length equal to that of the entire length of the Moonstone (1077 units). An exponential distribution of internal segmentation size where $\lambda = {}^{1}/_{\beta} = {}^{1}/_{5.8148}$—one over the mean internal segment size—is also used to produce a distribution similar to that of the original Moonstone data set. This results in internal segment size distributions that appear like the distribution shown in Figure 5.24a which approximates the natural distribution of the Moonstone data set shown in Figure 5.24b.

Ten sets of artificial segmentations to determine inter-coder agreement between are generated by:

1. Creating 10 random reference segmentations 1077 units long from an exponential distribution where $\lambda = {}^{1}/_{5.8148}$;

2. For each reference segmentation, generating 9 random segmentations from the reference segmentation where the probability of a full miss or near miss is specified by each individual experiment[5];

[5]The probability of a boundary being placed in the reference segmentations is specified by the internal segment sizes randomly selected from the exponential distribution, and because the 9 random segmentations are generated from the reference then they too will have approximately the same probability of containing boundaries.

3. For each reference segmentation and its associated 9 random segmentations, computing inter-coder agreement coefficients for $2 \leq c \leq 10$ coders (sequentially);

4. For each of the 10 coefficients computed for a given number of coders c, averaging this value to smooth the plots of the number of coders versus coefficient value.

Ten inter-coder agreement coefficient values are then calculated for each coefficient type and number of coders c for $2 \leq c \leq 10$. These are then plotted to show how agreement values fluctuate for a given scenario when the number of coders is increased.

Four experiments were performed to test both S and B_b-based inter-coder agreement coefficients in four different scenarios:

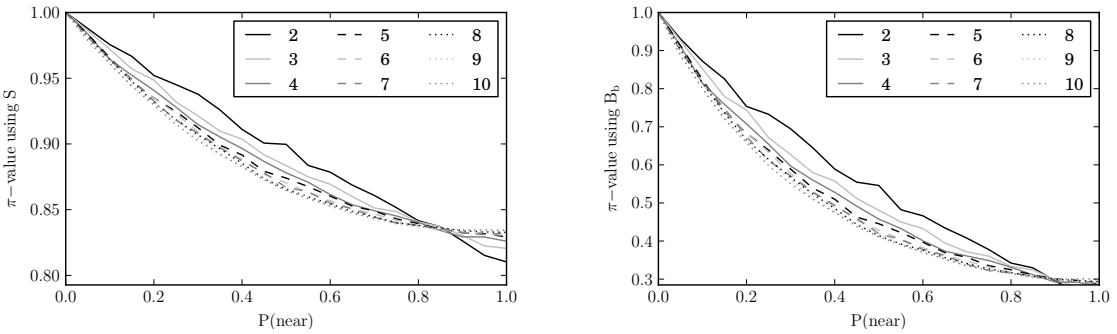
- Increasing near misses with no full misses;
- Increasing full misses with no near misses;
- Increasing full misses with a natural probability of near misses; and
- Completely random segmentations.

The completely random segmentations test how each coefficient responds to low agreement, whereas the other three experiments test the effects of various combinations of probabilities of full and near misses or both. For each experiment, a hypothesis is defined which is based upon the assumptions of how an ideal inter-coder agreement coefficient would perform, and it is then tested for π^*_S and $\pi^*_{B_b}$.

5.2.2.1 Increasing Near Misses

In this experiment, the probability of a near miss is increased as the probability of a full miss is kept at zero for $2 \leq c \leq 10$ coders. How do both the S and B_b-based coefficients react to increasing near misses and varying numbers of coders? The hypothesis is that each coefficient will perform nearly identically regardless of the number of coders, and will decrease in agreement as the number of near misses is increased.

Plotting the agreement for both coefficients for increasing near misses and varying numbers of coders results in Figure 5.25. From these results it is evident that although S is slightly elevated in the agreement that it reports, the hypothesis appears to hold true for both coefficients. For this scenario, both coefficients appear to be suitable for measuring inter-coder agreement as they both decrease in agreement as near misses are increased.



(a) S (b) B_b

Figure 5.25: Artificial data set with increasing near misses illustrating adapted versions of π^* with varying numbers of coders

5.2.2.2 Increasing Full Misses

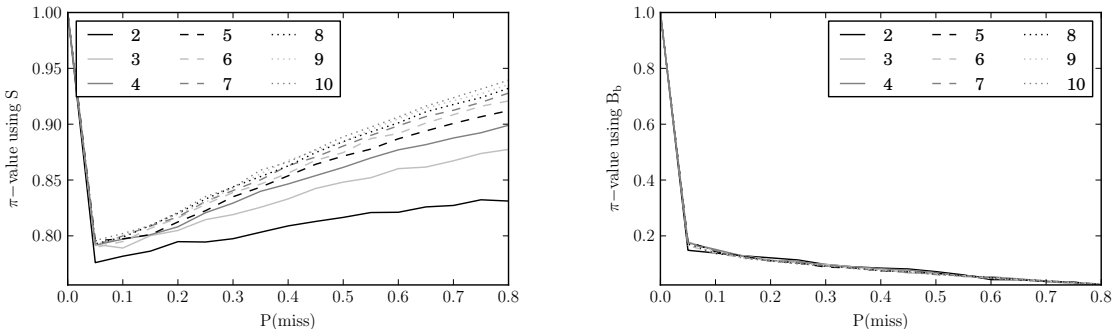
In this experiment, the probability of a full miss is increased as the probability of a near miss is kept at zero for $2 \leq c \leq 10$ coders. How do both the S and B_b-based coefficients react to increasing full misses and varying numbers of coders? The hypothesis is that each coefficient will perform nearly identically regardless of the number of coders, and will decrease in agreement as the number of full misses is increased.

Plotting the agreement for both coefficients for increasing full misses and varying numbers of coders results in Figure 5.26. From these results it is evident that the hypothesis appears to hold true for the B_b-based coefficient. The S-based coefficient, however, does not perform similarly for varying numbers of coders and it also increases in agreement as opposed to decreasing. For this scenario only B_b is suitable for measuring inter-coder agreement because it performs consistently for varying numbers of coders and decreases in agreement as the number of full misses is increased.

5.2.2.3 Increasing Full Misses with Near Misses

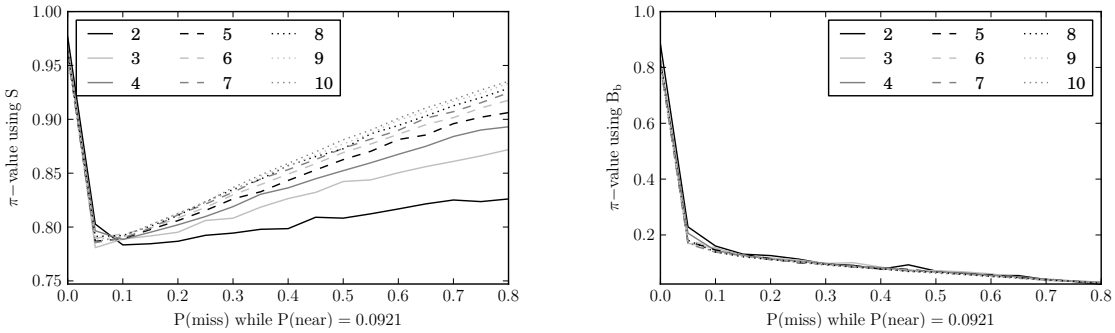
In this experiment, the probability of a full miss is increased as the probability of a near miss given that a boundary is present is kept at 0.0921 (as in the Moonstone data set) for $2 \leq c \leq 10$ coders. How do both the S and B_b-based coefficients react to increasing full misses and varying numbers of coders in the presence of near-misses? The hypothesis is that each coefficient will perform nearly identically regardless of the number of coders, and will decrease in agreement as the number of full misses is increased.

Plotting the agreement for both coefficients for increasing full misses and varying



(a) S (b) B_b

Figure 5.26: Artificial data set with increasing full misses illustrating adapted versions of π^* with varying numbers of coders



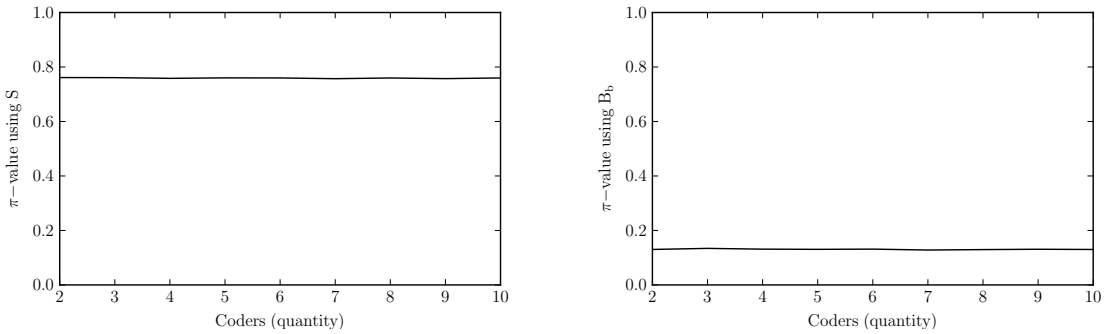
(a) S (b) B_b

Figure 5.27: Artificial data set with increasing full misses with near misses illustrating adapted versions of π^* with varying numbers of coders

numbers of coders results in Figure 5.27. From these results it is evident that the hypothesis appears to hold true for the B_b-based coefficient. The S-based coefficient, however, does not perform similarly for varying numbers of coders and it also increases in agreement as opposed to decreases—similar to the previous experiment in Section 5.2.2.2. For this scenario only B_b is suitable for measuring inter-coder agreement because it performs consistently for varying numbers of coders and decreases in agreement as the number of full misses is increased.

5.2.2.4 Random Segmentations

In this experiment, random segmentations are generated in the same manner as the reference segmentation and are all compared against each other for $2 \leq c \leq 10$ coders.



(a) S (b) B_b

Figure 5.28: Artificial data set with random boundary placement illustrating adapted versions of π^* with varying numbers of coders

How do both the S and B_b-based coefficients react to varying numbers of coders for random segmentations? The hypothesis is that each coefficient will perform nearly identically regardless of the number of coders, and will report very low agreement.

Plotting the agreement for both coefficients for varying numbers of coders results in Figure 5.28. From these results it is evident that both coefficients remained steady as coders were varied, but S reports highly inflated agreement and the hypothesis does not hold true for S. The hypothesis appears to hold true for the B_b-based coefficient, however, as it correctly reports very low agreement. For this scenario only B_b is suitable for measuring inter-coder agreement because it correctly reports low agreement.

5.2.3 Discussion

In this section, artificial segmentation data sets were described that emulate the properties of manual segmentations in the natural Moonstone data set. These artificial data sets were created to answer questions about how two adapted inter-coder agreement coefficients would react in scenarios where:

- Error probabilities were varied;
- The number of coders was varied; and
- Unambiguously low agreement in the form of random segmentations was evaluated.

Hypotheses were tested where each represents the ideal behaviour expected of an inter-coder agreement coefficient. Those coefficients for which the hypotheses were most often

Experiment		*Hypothesis for each coefficient* S-based π^*	B_b-based π^*
§5.2.2.1	Increasing near misses	●	●
§5.2.2.2	Increasing full misses	○	●
§5.2.2.3	Increasing full misses with near	○	●
§5.2.2.4	Random segmentations	○	●

● Correct ● Mostly correct ● Partially correct ○ Incorrect

Table 5.6: Correctness of the hypotheses in each experiment for each adapted inter-coder agreement coefficient

correct represent the ideal adapted inter-coder agreement coefficients. The validity of these hypotheses is summarized in Table 5.6 with the strength of the correctness of the hypothesis for each experiment shown for each comparison method.

It is clear from Table 5.6 that S-based π^* is an inadequate inter-coder agreement coefficient for segmentation. B_b-based π^*, however, is able to perform properly in all of the scenarios tested, meaning that it is highly suited for measuring the reliability of a segmentation data set.

Given that B_b-based π^* is the ideal coefficient to report reliability for segmentation data sets, the reliability of the Stargazer data set is then $\pi^*_{B_b} = 0.4405$, a low level of agreement. The reliability of the Moonstone data set is even lower at $\pi^*_{B_b} = 0.20, 0.18, 0.40, 0.38,$ and 0.23 for groups 1 through 5, respectively.

5.3 Evaluating Segmenters

Having selected B_b as an ideal comparison method in Section 5.1 and evaluated two data sets and selected $\pi^*_{B_b}$ as an ideal adapted inter-coder agreement coefficient in Section 5.2, this section tests how these tools can be used to select the best automatic segmenter for a task. The three comparison methods are compared to determine whether they can distinguish between three automatic segmenters, an upper bound, and a lower bound while evaluating them all against manual codings. During this evaluation, statistical hypothesis testing is demonstrated and a detailed error analysis is performed.

5.3.1 Evaluation

In addition to the experiments conducted upon the comparison methods in Section 5.1, this section tests whether S, B_b, or $1-WD$ are able to discern between the upper and

lower bounds. These upper and lower bounds are randomly selected manual segmentations and randomly generated segmentations, respectively. Three automatic segmenters were also trained and run by a second researcher, Anna Kazantseva, and were evaluated in the context of the random baseline and manual upper bound—also generated and selected by Anna Kazantseva.

The data set used for training and evaluating segmenters in this section is the Moonstone data set collected by Kazantseva and Szpakowicz (2012). It was chosen primarily because of its large size (as detailed in Section 5.2.1). Despite its low inter-coder agreement, its size makes it an ideal data set to perform statistical hypothesis testing upon because the number of subjects can be kept high enough to maintain the power of the tests used.

The three automatic segmenters whose segmentation output upon 15 chapters of the Moonstone data set include:

- Bayesian Unsupervised Segmentation (BayesSeg; Eisenstein and Barzilay 2008),
- Affinity Propagation for Segmentation (APS; Kazantseva and Szpakowicz 2011),
- Minimum Cut Segmenter (MinCut; Malioutov and Barzilay 2006).

All three were trained or had parameters estimated by Anna Kazantseva upon chapters 1–5 of the Moonstone data set. The labels of the five segmenters (Human, Random, BayesSeg, APS, and MinCut) were then removed and random labels were assigned to their output segmentations for chapters 5 and 7–21. This label obfuscation was performed so that during the development of an evaluation methodology there was no bias introduced the author towards favouring any segmenter (e.g., the upper bound). Additionally, this blind development meant that no automatic segmenter could be retrained; they were optimized for WindowDiff by Anna Kazantseva.

The question of interest is whether the comparison methods can be used to perform statistical hypothesis testing which could discern between all 5 segmenters. To perform correctly, the comparison methods should be able to rank the random baseline as the worst performing and the manual segmentations as the highest performing automatic segmenters. It is then expected that the three truly automatic segmenters should—if they are well-trained—be between the upper and lower performance bounds. The hypothesis is that each comparison method will be able to rank the automatic segmenters as previously described, or at least with the random baseline far below the manual-segmentation upper-bound.

To test this hypothesis upon each comparison method, the following procedure was followed using the Moonstone data set and the output of the artificial segmenters:

1. Each manual segmentation was compared against each of the five automatic segmentations for each chapter and comparison method;

2. Assuming that each natural manual segmentation of a chapter is a subject, comparisons were separated into sets by comparison method, and for each set:

   i) Comparison method sets were again separated into sets representing each automatic segmenter;

   ii) The Shapiro-Wilk test (Shapiro and Wilk, 1965) for normality was applied to each automatic segmenter comparison set to determine whether their distributions were normal;

   iii) Bartlett's test (Snedecor and Cochran, 1989) for equal variances was applied to the set of sets of automatic segmenter comparisons to test for homoscedasticity.

   iv) The Shapiro-Wilk and Bartlett's tests were applied to check that ANOVAs assumptions of normality and homoscedasticity are not violated:

   - If not violated, 1-factor repeated measures (i.e. within subjects) ANOVA with Tukey's HSD as a post hoc test was used to compare segmenters; otherwise
   - If violated, then Friedman's rank sum test with the Wilcoxon-Nemenyi-McDonald-Thompson post-hoc test was applied.

For the hypothesis testing used, the subjects were chapter codings and the 1 factor was the automatic segmenters.

The macro-average performance of each automatic segmenter using each comparison method is listed in Table 5.7. Visualizations of the performance with 95% confidence intervals is provided in Figure 5.29. For S and B_b, the null hypothesis of normality was rejected for all groups using the Shapiro-Wilk test, although $1-WD$ did so only for 1 of five groups (where each group is the set of comparisons for a specific automatic segmenter). Bartlett's test of for equal variances rejected the null hypothesis of homoscedasticity for both B_b and $1-WD$. Given that one or more of ANOVA's assumptions do not hold true for all three comparison methods, Friedman's rank sum test with the Wilcoxon-Nemenyi-McDonald-Thompson post-hoc test was applied to the results using $\alpha = 0.05$.

	S	B_b	$1-WD$	n
Random	0.8293 ± 0.0073	0.1115 ± 0.0110	0.4285 ± 0.0130	90
Human	0.9070 ± 0.0070	0.4211 ± 0.0324	0.6597 ± 0.0249	90
BayesSeg	0.8628 ± 0.0072	0.2399 ± 0.0185	0.5282 ± 0.0180	90
APS	0.8852 ± 0.0062	0.0558 ± 0.0084	0.5745 ± 0.0131	90
MinCut	0.8591 ± 0.0069	0.0508 ± 0.0098	0.5125 ± 0.0117	90

Table 5.7: Mean performance of 5 segmenters using varying comparison method macro-averages and standard error

The test found that the means produced by each comparison method contained significant differences ($p < 2.2 \times 10^{-16}$).

Applying the post-hoc test, significant differences ($\alpha = 0.05$) were found between nearly half of the comparisons, including:

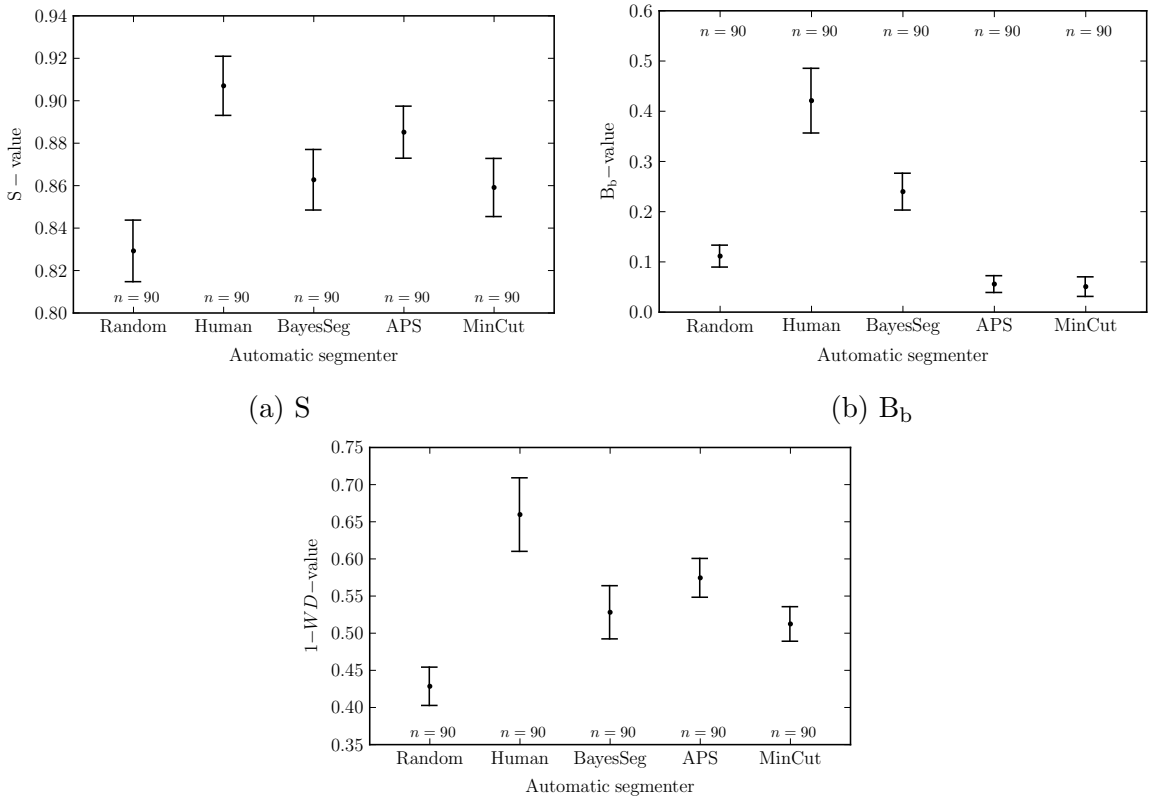
- Using S, between APS–human and MinCut–BayesSeg;
- Using B_b, between MinCut–APS;
- Using $1-WD$, between APS–BayesSeg, MinCut–BayesSeg, MinCut–APS.

A full summary of the significance values per comparison method and comparison is shown in Table B.1.

5.3.2 Evaluation using B_b

A micro-average can be computed at the boundary pair level using B_b. Using this comparison method and method of producing a micro-average—as detailed in Section 4.3, the same experiment was conducted where some additional statistics were also calculated. These additional statistics included a boundary-edit-distance-based confusion matrix, B-precision, B-recall, and B-F_β-measure–again, as detailed in Section 4.3. In this experiment design, the factor remains the automatic segmenters, but the subjects are defined as individual boundary pairs compared using a between-subjects design (i.e., not repeated measures).

The mean performance of each automatic segmenter using micro B_b per pair is listed in Table 5.8. Visualizations of the performance with 95% confidence intervals is provided in Figure 5.30. The null hypothesis of normality was rejected for all groups using the Shapiro-Wilk test ($p < 2.2 \times 10^{-16}$), and Bartlett's test of for equal variances rejected the null hypothesis of homoscedasticity ($p = 0.00019$). Given that both of ANOVA's



(a) S

(b) B_b

(c) $1-WD$

Figure 5.29: Mean performance of 5 segmenters using varying comparison methods with 95% confidence intervals

assumptions do not hold true, the Kruskal-Wallis rank sum test with the Kruskal-Wallis rank sum multiple comparison post-hoc test was applied to the results using $\alpha = 0.05$. The test found that the means contained significant differences ($p < 2.2 \times 10^{-16}$).

Applying the post-hoc test, significant differences ($\alpha = 0.05$) were found between all comparisons except a few, including between APS–MinCut, APS–Random, and MinCut–Random.

5.3.3 Discussion

When calculating macro-averages using codings as subjects as in Section 5.3.1, the hypothesis that each comparison method would rank the random baseline as lowest performing, the three automatic segmenters above the baseline, and the human segmentations as highest performing is mostly correct for S and $1-WD$, but only partially correct for

	B_b	n	P	R	F_1	TP	FP	FN	TN
Random	0.2640 ± 0.0129	1057	0.3991	0.4673	0.4306	279.0	420	318	4236.0
Human	0.5285 ± 0.0164	841	0.6854	0.7439	0.7135	444.5	204	153	4451.5
BayesSeg	0.3745 ± 0.0146	964	0.5247	0.6224	0.5694	361.0	327	219	4346.0
APS	0.2873 ± 0.0163	738	0.6773	0.3403	0.4530	212.0	101	411	4529.0
MinCut	0.2468 ± 0.0141	871	0.4788	0.3496	0.4041	215.0	234	400	4404.0

Table 5.8: Mean performance of 5 segmenters using micro-average B_b and boundary edit distance based precision (P), recall (R), and F_β-measure (F_1) along with the associated confusion matrix values for 5 segmenters

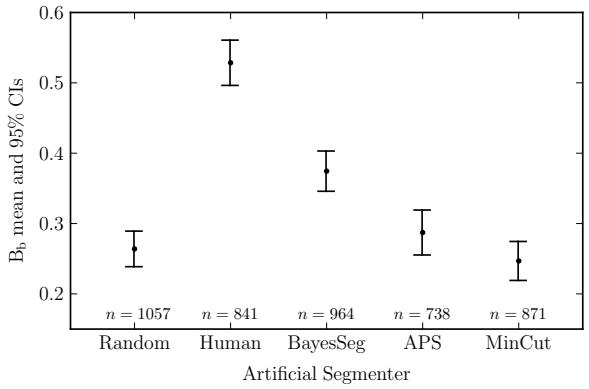


Figure 5.30: Mean performance of 5 segmenters using micro-average B_b with 95% confidence intervals

B_b. S shows one minor deviation from the hypothesis because statistical hypothesis testing failed to discern between APS and the human segmentations. $1-WD$ also shows a minor deviation from the hypothesis because the statistical hypothesis tests were unable to discern between any of the three truly automatic segmenters (despite placing them between the baseline and human upper bound).

B_b properly placed the random baseline below the human segmentations, and rated BayesSeg between both the random and human segmentations (where the rest of the automatic segmenters were expected to be ranked along with BayesSeg). B_b then ranked APS and MinCut below the random baseline and did not discern between the two using statistical hypothesis testing. Why did this ranking occur? Why does the hypothesis not hold true for B_b?

To answer why the hypothesis did not hold for B_b, we can look to the performance of B_b when micro-averages were calculated using boundary pairs as subjects in Section 5.3.2. Using this method of calculation, some manner of error analysis is possible through the

computation of edit counts and the calculation of a confusion matrix to produce precision and recall numbers.

First, looking at the boundary pair subject micro-averages in Table 5.8 and the graphical representation found in Figure 5.30, the post-hoc test was unable to discern any differences between the random baseline and APS and MinCut. Instead, boundary pair subject B_b ranked them all below BayesSeg and the human upper bound. This is roughly the same ranking as determined in the coding subject case of B_b. Where the boundary pair subject case of B_b differs is in the explanation of why it ranked each segmenter. From the precision and recall values shown, both APS and MinCut appear to be more precise at boundary placement than the random baseline by a wide margin, but their recall is lower than the random baseline, the human upper bound, and BayesSeg. The recall values demonstrate why BayesSeg was ranked higher than the other two automatic segmenters, and why MinCut and APS failed to do better than the random baseline (which had very far better recall than these two automatic segmenters).

The components of these precision and recall values are visualized in Figure 5.31. A/D-n values are AD edits occurring when the boundary was in the natural segmentations (i.e., a FN), and A/D-a in the automatic segmentation (i.e., a FP), and T the transpositions. From the transposition values, it appears that BayesSeg also benefited from this evaluation, having more near misses than APS and MinCut. APS and MinCut appear to be more precise automatic segmenters, given their precision values and low transposition counts. However, in an overall evaluation that balances both precision and recall, APS and MinCut fail to perform better than the random baseline due to their poor recall and high A/D-n counts. BayesSeg performed well because it balanced both precision and recall and thus performed better than the random baseline, but below the human upper bound.

The rankings given by S appear inflated, given that APS and the human segmentations cannot be discerned between by statistical hypothesis testing, although the experiment's hypothesis was mostly true for it. Its inflated numbers and faults found during the comparison method evaluations in Section 5.1 lead one to question its results, and lead the author to not recommend its usage in segmentation evaluation.

$1-WD$ mostly satisfied the experiment's hypothesis as well, and although it does not suffer from the same value inflation issues as S, it also produced very similar results to S. $1-WD$ produced the correct overall ranking of the human baseline above and the random baseline below the three automatic segmenters, but statistical hypothesis testing was unable to discern between any of the three automatic segmenters. The reason for all of the segmenters performing equally well using $1-WD$ is most likely that they were

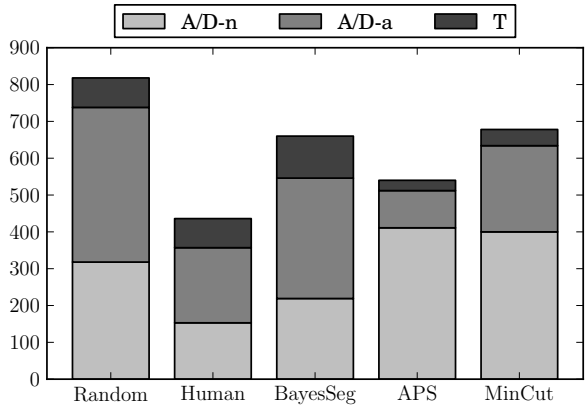


Figure 5.31: Edits from comparing automatic segmenters to all coders using boundary edit distance

all optimized for performance using $1-WD$, both during their initial development and proposals by their creators and also during their training for this experiment. It would then not be unexpected for this comparison method to satisfy the experiment's hypothesis. Another potential explanation is that we may be seeing the effects of $1-WD$'s bias towards sparse hypotheses (Niekrasz and Moore, 2010)—a likely explanation given that the B-precision and recall values in Table 5.8 indicate a lack of sensitivity to recall.

Despite the experiment's hypothesis not holding true for B_b, the comparison method in its per-boundary subject form in Section 5.1 was able to do something which the other comparison methods could not: provide a detailed error analysis and clear reason why it ranked segmenters the way that it did. The boundary-edit-distance-based confusion matrix and B-precision and B-recall values indicated that APS and MinCut were very precise, but poor at recall. APS and MinCut thus performed worse than the random baseline in a balanced evaluation that asked for both precision and recall of the segmenters it tested. It is reasonable to recommend the usage of B_b, B-precision and B-recall using boundary pairs as subjects for segmentation evaluation over both S and $1-WD$—if an unbiased evaluation metric is desired—because of B_b's performance in Section 5.1, B_b's ability to provide a detailed error analysis during segmentation evaluation, and sensitivity towards both precision and recall which $1-WD$ evidently lacked in this experiment.

5.3.4 Further Error Analysis

Using B_b micro-averages with boundary pairs as subjects allows one to easily perform 2 and 3-factor between-subjects experiment designs. Such experiment designs can allow us

one to explore more about how automatic segmenters perform when compared against multiple manual codings of multiple documents. In this section, 3-factor and 2-factor comparisons are made—where the factors are automatic segmenters, coders, and chapters—to determine whether any coders or chapters were significantly easier or more difficult to approximate/segment, respectively, than each other for the automatic segmenters.

Using B_b pairs as subjects and the Kruskal-Wallis rank sum test with the Kruskal-Wallis rank sum multiple comparison post-hoc test allowed for easy exploration of multiple factors, including whether coders or chapters were more or less difficult than the others. Performing these tests using 3 factors (automatic segmenters, coders, and chapters; $\alpha = 0.05$), a statistically significant difference was found ($p < 2.2 \times 10^{-16}$). Delving deeper and performing 2-factor comparisons, significant differences were found between automatic segmenters and chapters ($p < 2.2 \times 10^{-16}$), and between automatic segmenters and coders ($p < 2.2 \times 10^{-16}$). Since it is already known that significant differences were found between automatic segmenters in 1-factor comparisons, 1-factor comparisons were then made between both coders and chapters as factors. Significant differences were found between both coders ($p < 0.0007778$) and chapters ($p < 2.2 \times 10^{-16}$) indicating that some coders and chapters may have been more difficult to automatically segment than others.

Despite finding significant differences between coders, when performing the Kruskal-Wallis rank sum multiple comparison post-hoc test no significant differences between coders were found for $\alpha = 0.05$. Figure 5.32 shows the micro-average B_b obtained by all automatic segmenters against specific coders which appears to show that coders 1–4 were difficult for all automatic segmenters to perform well upon, with breakdowns for each automatic segmenter shown in Figures B.4–B.6. Despite the assumptions of ANOVA being violated, Tukey's HSD was used and was also applied and was only able to discern statistically significant differences between coders 2–11, 2–14, 3–11, and 3–13. In this case, the breakdowns per automatic segmenter may reveal statistically significant differences between coders 1–4 and other coders on a per-automatic segmenter bases. There does not, however, appear to be a statistically significant general trend amongst the three automatic segmenters.

Performing the Kruskal-Wallis rank sum multiple comparison post-hoc test upon chapters found significant differences ($\alpha = 0.05$), including between:

- Chapter 7 and chapters 5, 8, 10, 11, 12, 13, 14, 15, 16, 18;
- Chapter 9 and chapters 12, 15, 18;
- Chapter 11 and chapters 7, 8, 9, 20.

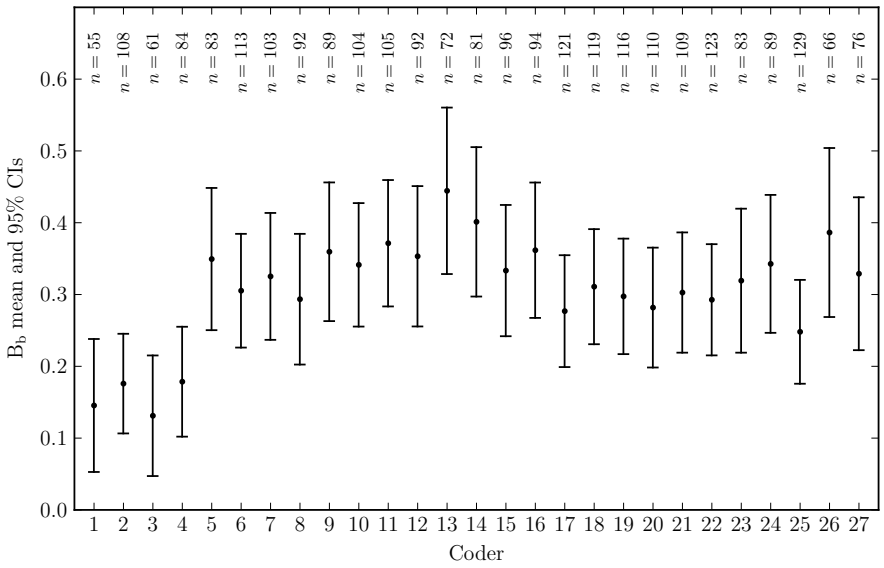


Figure 5.32: Automatic segmenter performance against each coder using per boundary pair subject B_b with 95% confidence intervals

These results are visualized in Figure 5.33 against micro-averages of all three automatic segmenters. Breakdowns for each individual automatic segmenter are shown in Figures B.1–B.3. From this analysis, it appears evident that Chapter 7 and 9 were significantly easier than a number of other chapters to segment (given the set of codings analysed herein). Chapter 11, however, was significantly more difficult than a number of other chapters, which appears to be supported by Figure 5.33.

Having identified that some chapters are significantly different in terms of automatic segmenter performance than others, 2-factor between subjects ANOVA upon automatic

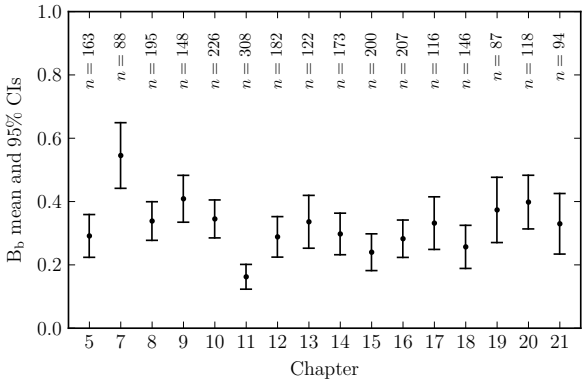


Figure 5.33: Automatic segmenter performance against all coders over each chapter using per boundary pair subject B_b with 95% confidence intervals

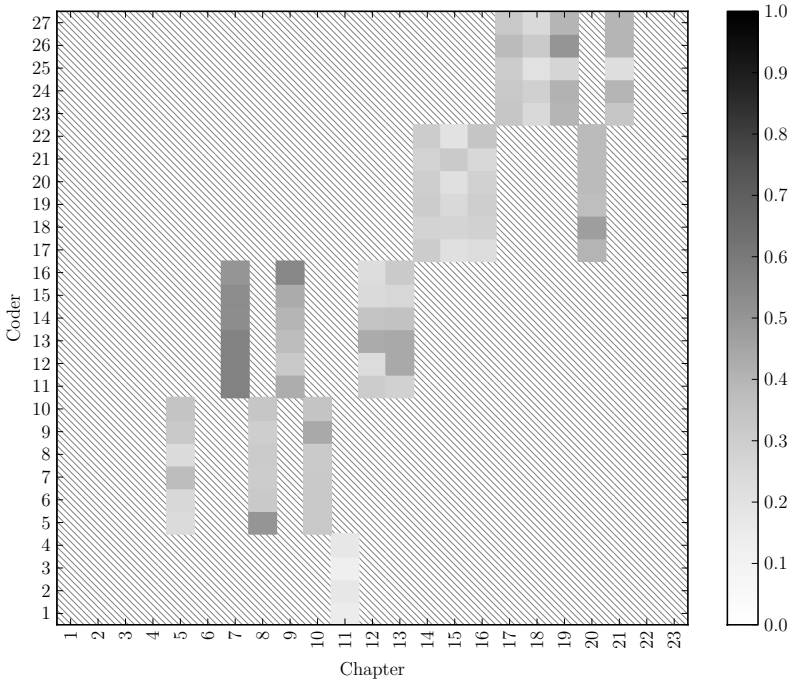


Figure 5.34: Automatic segmenter performance for each chapter and coder using per boundary pair subject B_b

segmenter performance using per boundary pair subject B_b and using coders and items as factors—despite ANOVA's assumptions being violated—was run, but no significant differences were found between coder and item pairs ($\alpha = 0.05$). Despite this lack of differences, plotting a heat map of chapters versus coders where intensity is automatic segmenter performance in terms of micro-average per boundary pair subject B_b reveals some areas of high intensity, as shown in Figure 5.34. All codings of chapter 7 appear to have been the easiest codings for all automatic segmenters, whereas chapter 11 appears to have been very difficult (which agrees with the previous per-chapter analysis). Chapter 9, which was also supposed to have been relatively easy, shows high intensity for one coder, as do several other chapters (8, 19, 20), indicating that although the data set was difficult for the automatic segmenters as a whole, there are subsets of coders and chapters for which the automatic segmenters performed well upon. A breakdown of the same analysis for each individual automatic segmenter is shown in appendix Figures B.7–B.9.

Chapter 6

Discussion on Segmentation Comparison Method Reporting

This work seeks to replace some of the existing segmentation evaluation comparison methods (e.g., P_k, WindowDiff, etc.) with Boundary Similarity (B) and information retrieval (IR) metrics derived from a boundary-edit-distance-based (BED) confusion matrix (e.g., precision, recall, etc.). For all tasks, however, no single comparison method is able to provide a full picture of the performance of an automatic segmentation algorithm. As an unbiased measure of similarity, B provides only part of the picture of an automatic segmenter's ability to approximate a set of reference solutions.

Different tasks require different metrics For each segmentation task, success and failure may have unique definitions which are not synonymous with similarity. This means that some tasks are better evaluated using comparison methods with specific biases in addition/opposed to an unbiased comparison method such as B.

An information retrieval task which uses text segmentation to provide excerpts of documents to users (e.g., Franz et al. 2007) may need high recall (every document must have an excerpt), but whether the boundary it places is off by 20-30 words is not a serious error in a 150-200 word summary (meaning that precision is not critical). For such a task, BED-based IR metrics such as B-recall may be best when balanced against B-precision while configuring near misses to be very tolerant (e.g., transposition spanning distance can be configured to indicate a near miss up to 20-30 words away).

In the hierarchical segmentation of a play for act and scene breaks, a serious error may be the confusion between a scene and an act boundary and not the lack of a boundary

(provided that those adjacent to it are correctly annotated). For this task, the percentage of act and scene boundaries confused should be reported along with B (since overall similarity to a reference solution is desirable).

Section 5.1 provides an even more tangible case for the need to not only choose the correct comparison method to report, but to also report multiple comparison methods when attempt to answer the question of how an automatic segmenter failed. B provides a succinct picture of how similar a segmenter was to a human solution, but it does not show how a segmenter failed. The reason why APS and MinCut had such low B values in that experiment was revealed to be caused by low recall values—APS and MinCut were very precise at the cost of low recall (e.g., infrequent boundary placement). B may provide an adequate summary of how well topical segmenters approximate a set of human solutions, but alone it cannot answer how a segmenter failed, unless it is accompanied by other statistics such as a BED-based confusion matrix.

P_k and WindowDiff have varying biases in varying situations which make them very difficult to interpret. Their occasional bias towards precision is sometimes desirable, however, this and their other biases are unpredictable in many situations, making them unreliable and difficult to interpret. For this reason, this work recommends reporting:

- Multiple BED-based IR metrics when biased comparison methods are required;
- B when an unbiased comparison method is required; and a
- BED-based confusion matrix with all comparison methods to provide context.

Because no one comparison method provides a complete picture of the performance of a segmenter, though, reporting B and BED-based precision, recall, and a confusion matrix would provide the most complete picture of performance and a discussion should be included that puts all of the results into context. At very least, those comparison methods exhibiting the biases that best suit a specific task's evaluation should be reported along with an unbiased comparison method like B to provide them context.

Comparing segmenters in context In addition to reporting multiple comparison methods and other statistics when evaluating automatic segmenters, it is important to put results in context. The best context for automatic segmenters is a comparison against an upper and lower bound. In Section 5.1 random segmentations and randomly chosen human segmentations served as lower and upper bounds, respectively. Such bounds allow for the accurate interpretation of statistics, e.g., (from Table 5.8):

- APS's precision of 0.6773 seems low, until it is compared to the upper of 0.6854;

- APS' recall seems low at 0.3403, but in good company with MinCut's 0.3496, but both appear disastrous in comparison to the lower bound's 0.4673.

Because it is difficult to obtain high manual agreement in segmentation tasks, it is important to provide an upper bound to the task to provide adequate context to what are often poor-looking results. Results should also be compared to a random baseline to serve as a lower bound or to show what level of performance an overly-simplistic model could obtain.

Chapter 7

Conclusions and Future Work

In this chapter, conclusions to the discussions presented in Chapter 5 are presented and a variety of future work is mentioned.

7.1 Conclusions

The thesis statement in Section 1.1 states that:

> *The minimal boundary edit-distance proposed herein can be normalized to create a segmentation comparison method that improves upon the state-of-the art, can be used to adapt inter-coder agreement coefficients to measure the reliability and replicability of manual segmentations, and with the evaluation methodology proposed herein can be used to evaluate the relative performance of automatic and/or manual segmenters.*

To support the thesis statement, the following research questions were answered:

1. How can we best compare two arbitrary segmentations?
2. How can we determine the reliability of manual segmentations and replicability of how they are collected?
3. How can we select the best automatic segmenter for a task?

How can we best compare two arbitrary segmentations? To compare arbitrary segmentations of the same text item, a *boundary edit distance* (§4.1.2) was proposed to serve as a distance function to compare two segmentations. Four different normalizations

of this edit distance (S, B_a, B_b, and B_c) where evaluated for their suitability as segmentation comparison methods against WindowDiff (and inherently P_k) in Section 5.1. The discussion in Section 5.1.2 selected B_b as the best of the five comparison methods in a battery of experiments that tested their reactions to a variety of scenarios and types of errors, thus for the remainder of this work boundary similarity variant B will be referred to as *boundary similarity* (B).

B_b, now referred to simply as B, was chosen above all other comparison methods examined in Section 5.1 because it:

- Awards partial credit for near misses;
- Clearly and unambiguously differentiates between a full and near miss occurring separately or individually (§5.1.1.1–5.1.1.4);
- Reasonably orders a set of hypothetical segmentations taking into account situations where there were extraneous boundaries added or removed or varying degrees of a near miss occurring (§5.1.1.5);
- Reacts and discerns between situations with varying numbers of near and full misses (§5.1.1.6–5.1.1.7);
- Does not obscure errors that occurred within large documents (§5.1.1.8);
- Demonstrated a lack of significant bias to variations in internal segment size variance (§5.1.1.9); and
- Is not biased towards either precision or recall (§5.3.3);
- Offers an intuition that is clear and directly related to the phenomenon being observed in that it is essentially the ratio of the correctness of pairs of boundaries over all pairs of boundaries between two segmentations.

The other normalizations of boundary edit distance (S, B_a, and B_c) were unable to demonstrate all of these benefits, or had other flaws as discussed in Section 5.1.2.

WindowDiff (and inherently P_k), the most widely used segmentation comparison method today, was unable to outperform B during the experiments in Section 5.1 because, although it awards partial credit for near misses, it:

- Arguably does not as clearly express the degree of error represented by a full miss as well as B (§5.1.1.2);

- Does not reasonably order a set of hypothetical segmentations where there were extraneous boundaries added or removed or varying degrees of a near miss occurring (§5.1.1.5);
- Does not consistently react to an increase in the number of full misses in a segmentation (§5.1.1.6);
- Does not provide a consistent value between varying sizes of segmentations and instead oscillates in value until it converges at a result as the size of a segmentation approaches infinity (§5.1.1.8);
- Is significantly biased in the presence of variations in internal segment size variance (§5.1.1.9);
- Is evidently biased towards sparse and precise segmentations (at the cost of recall; §5.3.3); and
- Is not clearly related to the phenomenon being observed because it interprets errors through the concept of windows as opposed to operating upon boundaries directly as the other methods in this work do.

It can be reasonably concluded that WindowDiff is significantly flawed. WindowDiff is a flawed comparison method which was used simply because it has been the best comparison method available at the time and because it improved upon its predecessors (e.g., P_k). From the results of the experiments performed in Section 5.1, it can be reasonably concluded that *boundary similarity* (B) is the best method of comparing two arbitrary segmentations of the comparison methods tested.

How can we determine the reliability of manual segmentations and replicability of how they are collected? The literature review presented in Section 3.2 identified that an ideal inter-coder agreement coefficient for determining the reliability of data collected and replicability of applying a set of instructions and labels is Scott's π and Fleiss' multi-π (Scott, 1955; Fleiss, 1971)–referred to as π and π^*. These coefficients do not, however, award partial credit for near misses. Section 4.2 detailed an adaptation of these coefficients which could make use of an ideal segmentation comparison method such as B or S to allow them to award partial credit for near misses.

Section 5.2 evaluated whether B, in comparison to S, was an ideal segmentation comparison method to use to adapt π and π^* to measure inter-coder agreement for segmentation. It was demonstrated that the adaptation using B, π^*_{B}:

- Can differentiate between high and low agreement for multiple numbers of coders in the presence of near misses, full misses, and both forms of error (§5.2.2.1–5.2.2.3);
- Can identify low agreement for multiple numbers of coders when presented with fully randomly generated segmentations (§5.2.2.4); and
- Can intuitively characterize agreement while analysing boundary pairs (a property of B).

S was unable to differentiate between high and low agreement in all situations tested, unable identify low agreement, and is inherently unable to intuitively characterize agreement as being between boundary pairs. It can then be reasonably concluded that π^*_{B} is an effective inter-coder agreement coefficient for segmentation.

How can we select the best automatic segmenter for a task? Section 5.3 demonstrates the usage of S, B, and WindowDiff when used to evaluate three automatic segmenters in terms of a random baseline and a human upper bound. The goal of this section was to demonstrate the usage of statistical hypothesis testing. Additionally, these three comparison methods were tested to see whether they could be used to distinguish between the performance bounds and evaluate the three automatic segmenters in terms of these bounds.

Macro-averages of all three coefficients were able to discern between the random baseline and human upper bound, and S and WindowDiff both placed the three automatic segmenters between these two performance bounds. B, however, varied from the analysis presented by S and WindowDiff and instead only placed one automatic segmenter, BayesSeg, between the performance bounds. It placed the other two automatic segmenters on par with the random baseline. B's deviation from S and WindowDiff bore further investigation.

When applying the per-boundary-pair-subject micro-average version of B, it was determined that the two automatic segmenters that were placed by B on par with the random baseline in terms of performance were very precise (using B-precision), but had poor recall (using B-recall). It was then hypothesized that because the automatic segmenters were optimized using WindowDiff that the ranking of the segmenters by

WindowDiff was expected. The deviation by B was suspect until the further error analysis revealed the reason for the ranking given: low recall for two of the automatic segmenters. This lack of recall was not identified by $1-WD$, which instead demonstrated it's bias towards sparse segmentations as noted by Niekrasz and Moore (2010). It could be reasonably concluded that micro-average B combined with B-precision, B-recall, and statistical hypothesis testing offer a superior set of tools and methodology for selecting the best automatic segmenter for a task—short of performing an ecological evaluation.

7.2 Future Work

Ecological evaluation Section 2.3 details that ecological validity is the most desirable property that an evaluation could have, e.g., when evaluating an automatic segmenter for a specific task we would evaluate its performance *in situ*. It would be desirable to perform an ecological evaluation of an automatic topical segmenter that could aid in the preprocessing efforts needed by a variety of NLP tasks. Unfortunately, a sufficient motivation for such an exercise is not yet present.

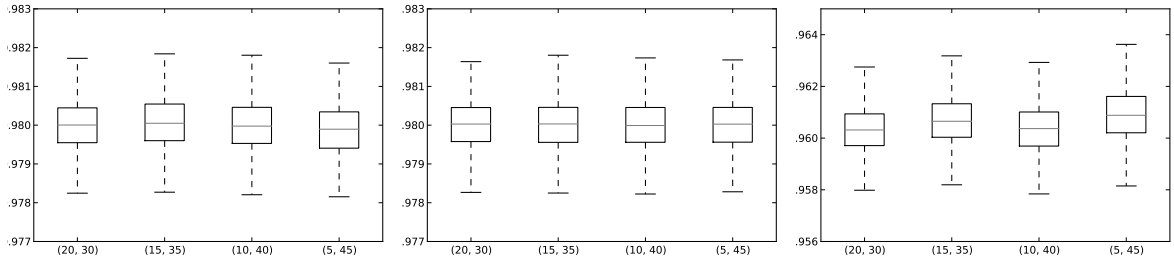
Hierarchical and multiple-boundary-type segmentation An abundance of linear single-boundary-type segmentation data has made this work possible. Although provisions were made such that boundary edit distance, boundary similarity, and π^*_{B} could be used for multiple-boundary-type segmentations, no experiments upon such data were performed herein. It would be desirable to demonstrate the tools and methodologies proposed herein upon both multiple-boundary-type and hierarchical segmentations.

Inter-coder agreement for topical segmentation Using the boundary similarity based inter-coder agreement coefficients defined in this work, a clearer picture of inter-coder agreement for topical segmentation can be sought. It would be desirable to determine how to collect topical segmentations with high inter-coder agreement to better understand the nature of topics in text.

Appendix A

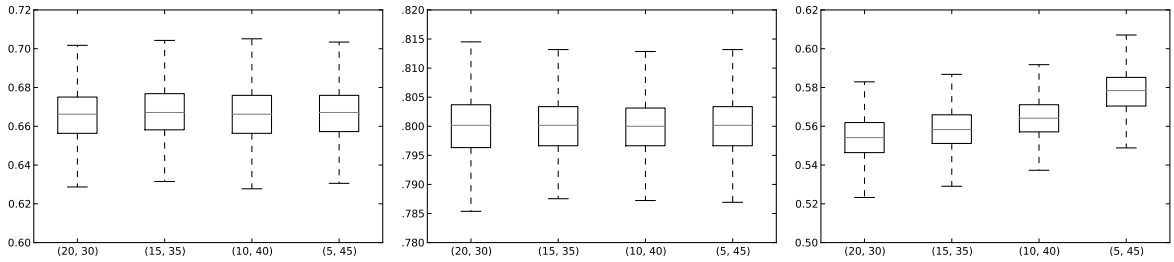
Linear Segmentation Comparison Method Experiment Results

A.1 Comparison Method Consistency



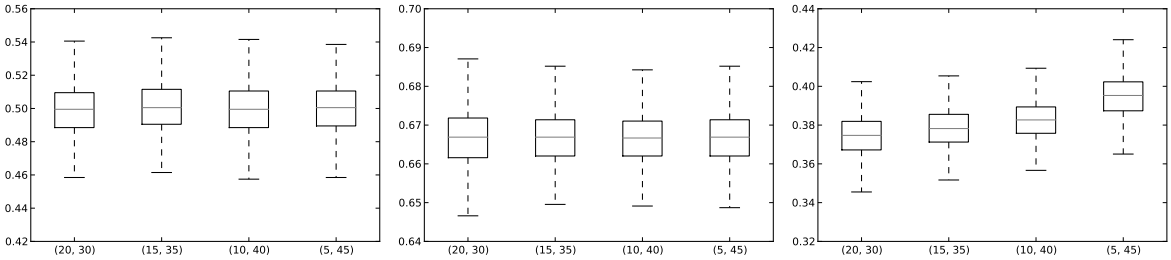
(a) FP $p = 0.5$ • (b) FN $p = 0.5$ ∘ (c) FP and FN $p = 0.5$ •

Figure A.1: Variations in S due to internal segment sizes



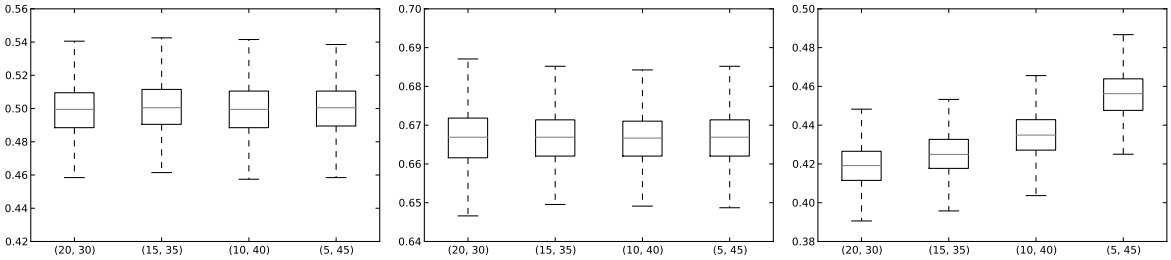
(a) FP $p = 0.5$ ∘ (b) FN $p = 0.5$ ∘ (c) FP and FN $p = 0.5$ •

Figure A.2: Variations in B_a due to internal segment sizes



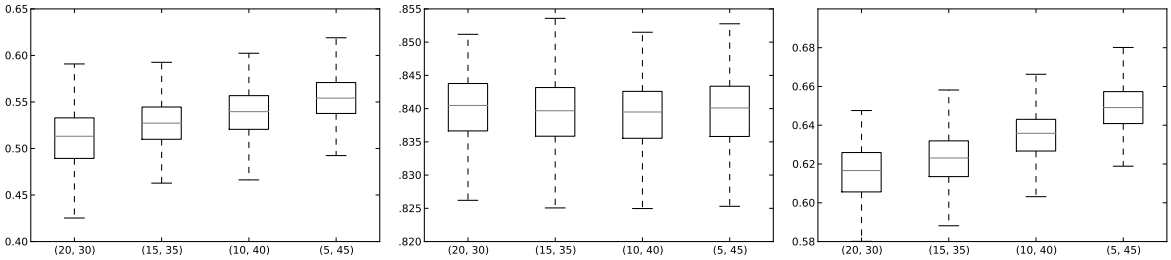
(a) FP $p = 0.5$ ∘ (b) FN $p = 0.5$ ∘ (c) FP and FN $p = 0.5$ •

Figure A.3: Variations in B_b due to internal segment sizes



(a) FP $p = 0.5$ ∘ (b) FN $p = 0.5$ ∘ (c) FP and FN $p = 0.5$ •

Figure A.4: Variations in B_c due to internal segment sizes



(a) FP $p = 0.5$ • (b) FN $p = 0.5$ • (c) FP and FN $p = 0.5$ •

Figure A.5: Variations in $1-WD$ due to internal segment sizes

Appendix B

Additional Experiment Results

B.1 Evaluating Segmenters

B.1.1 Multiple Comparison Tests

Table B.1 shows the p-values resulting from the multiple comparison tests performed using per chapter coding subjects and varying comparison methods.

		p-values		
		S	B_b	$1-WD$
Human	Random	0	1.476597×10^{-14}	0
BayesSeg	Random	7.628697×10^{-6}	8.378279×10^{-6}	1.761379×10^{-5}
APS	Random	0	6.164189×10^{-3}	3.017869×10^{-9}
MinCut	Random	8.794583×10^{-6}	1.742209×10^{-3}	1.447671×10^{-3}
BayesSeg	Human	7.106853×10^{-7}	1.994653×10^{-2}	1.689993×10^{-4}
APS	Human	9.691470×10^{-1}	0	4.790488×10^{-2}
MinCut	Human	6.391086×10^{-7}	0	9.842784×10^{-7}
APS	BayesSeg	3.323854×10^{-5}	8.881784×10^{-16}	5.448831×10^{-1}
MinCut	BayesSeg	1	1.110223×10^{-16}	8.545669×10^{-1}
MinCut	APS	2.500775×10^{-5}	9.967251×10^{-1}	8.441633×10^{-2}

Table B.1: Multiple comparison test results between automatic segmenters per comparison method ($\alpha = 0.05$)

B.1.2 Performance per Coder

Figure B.1–B.3 illustrate how individual automatic segmenters performed in terms of per boundary pair subject B_b upon codings produced by each coder.

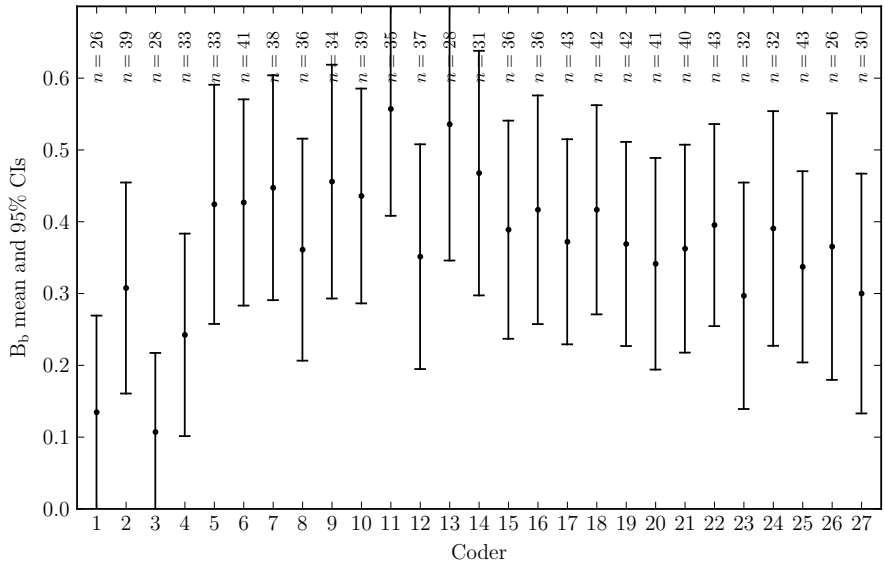


Figure B.1: BayesSeg performance against each coder using per boundary pair subject B_b with 95% confidence intervals

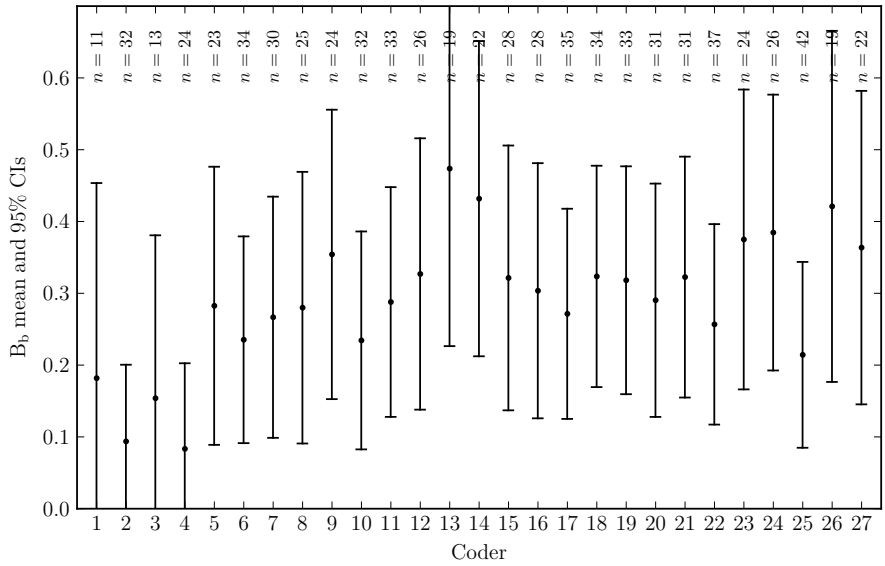


Figure B.2: APS performance against each coder using per boundary pair subject B_b with 95% confidence intervals

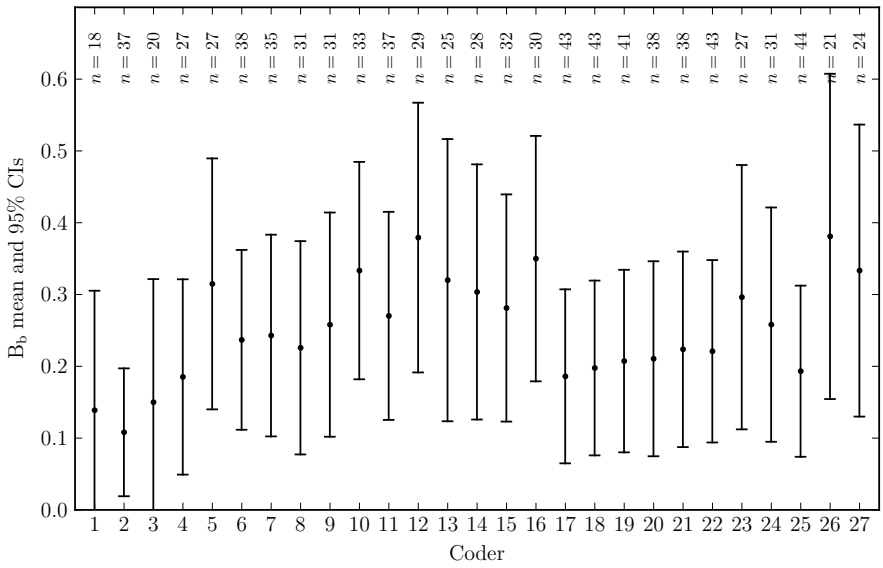


Figure B.3: MinCut performance against each coder using per boundary pair subject B_b with 95% confidence intervals

B.1.3 Performance per Chapter

Figures B.4–B.6 illustrate how individual automatic segmenters performed in terms of per boundary pair subject B_b upon codings of each chapter.

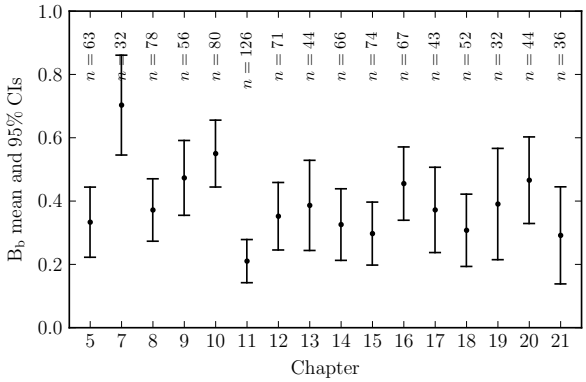


Figure B.4: BayesSeg performance against all coders over each chapter using per boundary pair subject B_b with 95% confidence intervals

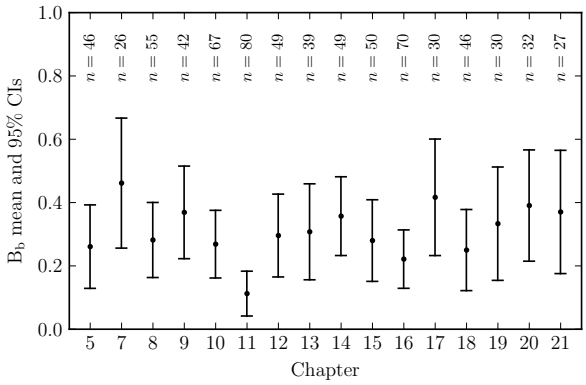


Figure B.5: APS performance against all coders over each chapter using per boundary pair subject B_b with 95% confidence intervals

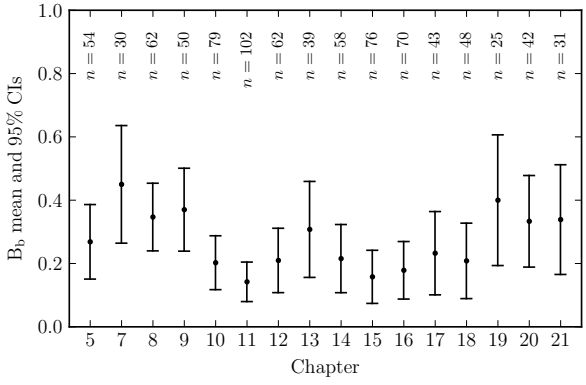


Figure B.6: MinCut performance against all coders over each chapter using per boundary pair subject B_b with 95% confidence intervals

B.1.4 Performance per Chapter and Coder

Figure B.7–B.7 illustrates how individual automatic segmenters performed in terms of per boundary pair subject B_b for each coder and chapter using a heat map.

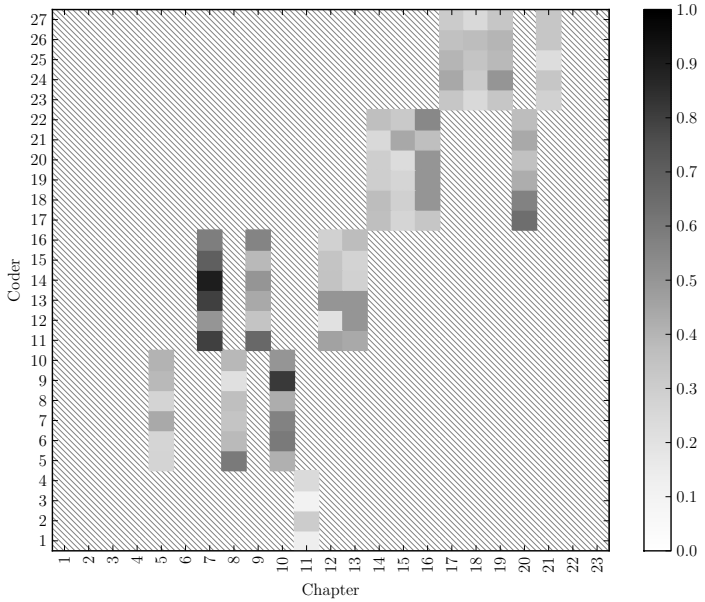


Figure B.7: BayesSeg performance for each chapter and coder using per boundary pair subject B_b

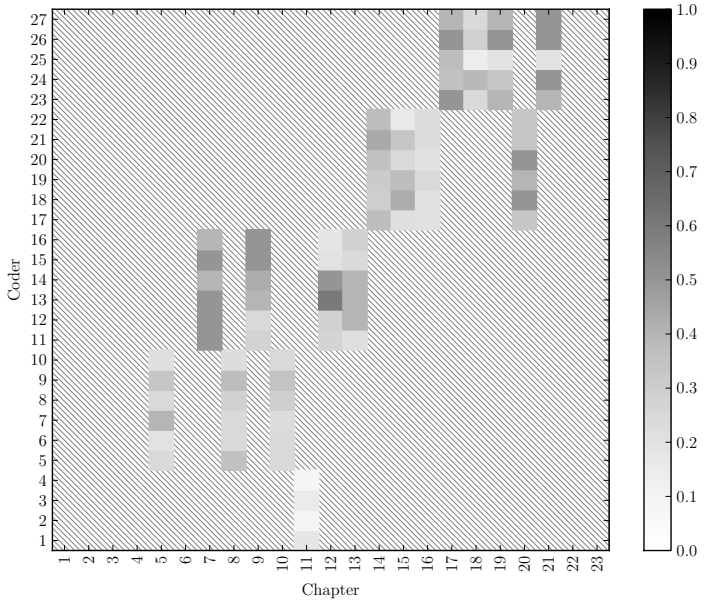


Figure B.8: APS performance for each chapter and coder using per boundary pair subject B_b

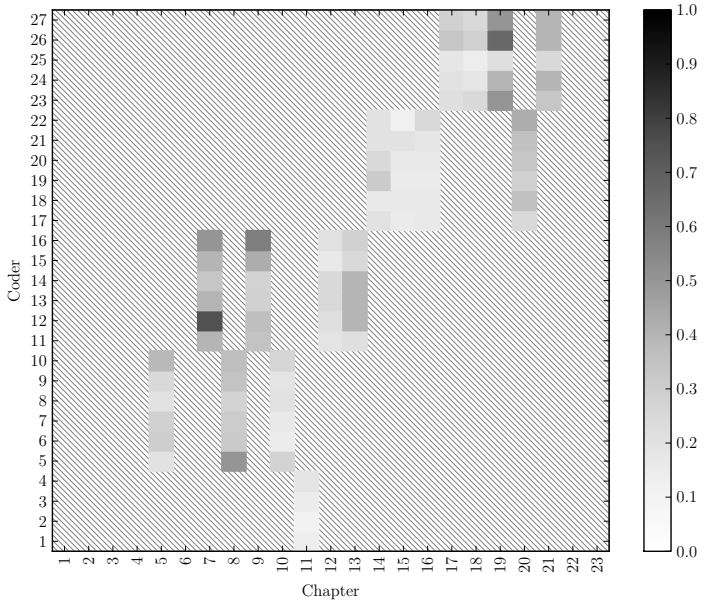


Figure B.9: MinCut performance for each chapter and coder using per boundary pair subject B_b

Bibliography

Akaike, H. 1974. A new look at the statistical model identification. *IEEE Transactions on Automatic Control* 19(6):716–723. 30

Allan, James, Yiming Yang, Jaime Carbonell, Jon Yamron, George Doddington, and Charles Wayne. 1998. TDT Pilot Study Corpus. 11, 54

Artstein, Ron and Massimo Poesio. 2005. Bias decreases in proportion to the number of annotators. In *Proceedings of FG-MOL 2005: The 10th conference on Formal Grammar and The 9th Meeting on Mathematics of Language*. pages 139–148. 53, 54

Artstein, Ron and Massimo Poesio. 2008. Inter-coder agreement for computational linguistics. *Computational Linguistics* 34(4):555–596. viii, xi, 3, 9, 22, 23, 24, 45, 46, 47, 51, 52, 53, 54, 55, 102, 103, 106, 107, 108, 109, 110, 111

Bakeman, Roger and John Mordechai Gottman. 1986. *Observing Interaction: An Introduction to Sequential Analysis*. Cambridge University Press. 49

Bakeman, Roger and John Mordechai Gottman. 1997. *Observing Interaction: An Introduction to Sequential Analysis*. Cambridge University Press, 2nd edition. 23, 49

Baker, David. 1990. Stargazers look for life. *South Magazine* 117:76–77. 151

Beeferman, Doug and Adam Berger. 1999. Statistical models for text segmentation. *Machine Learning* 34:177–210. 3, 32, 34, 55

Beeferman, Doug, Adam Berger, and John Lafferty. 1997. Text Segmentation Using Exponential Models. In *Proceedings of the 1997 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Stroudsburg, PA, USA, volume 2, pages 35–46. x, 3, 8, 9, 11, 13, 14, 15, 16, 54, 75

Bennett, E. M., R. Alpert, and A. C.Goldstein. 1954. Communications through limited response questioning. *Public Opinion Quarterly* 18:303–308. 45

Bishop, C. M. 1995. *Neural Network for Pattern Recognition*. Clarendon Press, Oxford, UK. 30

Brooke, Julian, Adam Hammond, and Graeme Hirst. 2012. Unsupervised Stylistic Segmentation of Poetry with Change Curves and Extrinsic Features. In *Proceedings of the 1st NAACL-HLT Workshop on Computational Linguistics for Literature*. Association for Computational Linguistics, Stroudsburg, PA, USA, pages 26–35. 41, 42

Byrt, Ted, Janet Bishop, and John B Carlin. 1993. Bias, prevalence and kappa. *Journal of Clinical Epidemiology* 46(5):423–429. 52

Carletta, Jean. 1996. Assessing Agreement on Classification Tasks: The Kappa Statistic. *Computational Linguistics* 22(2):249–254. 19, 24, 45, 46

Carletta, Jean, Amy Isard, Stephen Isard, Jacqueline C. Kowtko, Gwyneth Doherty-Sneddon, and Anne H. Anderson. 1997. The reliability of a dialogue structure coding scheme. *Computational Linguistics* 23:13–32. 48, 55, 56

Carlson, L., D. Marcu, and M. E. Okurowski. 2003. *Building a discourse-tagged corpus in the framework of rhetorical structure theory*, Springer, pages 85–112. 48, 55, 56

Chambers, J. M., A. Freeny, and R. M. Heiberger. 1992. *Analysis of variance; designed experiments*, Wadsworth & Brooks/Cole, chapter 5. 58, 117

Choi, FYY. 2000. Advances in domain independent linear text segmentation. In *Proceedings of the 1st Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Seattle, Washington, USA, pages 26–33. 11, 41, 54, 55

Cochran, W. G. 1950. The Comparison of Percentages in Matched Samples. *Biometrika* 37(3/4):pp. 256–266. viii, 10, 21, 22, 55

Cohen, Jacob. 1960. A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement* 20:37–46. 10, 22, 23, 45, 46, 49, 50, 52, 102, 107, 110, 111

Cohen, Jacob. 1968. Weighted kappa: Nominal scale agreement with provision for scaled disagreement or partial credit. *Psychological Bulletin* 70:213–220. 47

Cohen, Jacob. 1992. A Power Primer. *Psychological Bulletin* 112(1):155–159. 59

Cohen, Paul R. 1995. *Empirical methods for artificial intelligence*. MIT Press, Cambridge, MA, USA. 26, 30, 57, 58, 119

Coleridge, Samuel Taylor. 1816. *Christabel, Kubla Khan, and the Pains of Sleep*. John Murray. xi, 62, 63

Collins, Wilkie. 1868. *The Moonstone*. Tinsley Brothers. 151

Craggs, Richard and Mary McGee Wood. 2005. Evaluating discourse and dialogue coding schemes. *Computational Linguistics* 31(3):289–296. 19, 24, 47, 50, 51, 53, 102

Cumming, Geoff, Fiona Fidler, and David L Vaux. 2007. Error bars in experimental biology. *The Journal of Cell Biology* 177(1):7–11. x, 29, 58, 117

Damerau, Frederick J. 1964. A technique for computer detection and correction of spelling errors. In *Communications of the Association for Computing Machinery*. Association for Computing Machinery, Stroudsburg, PA, USA, volume 7, pages 171–176. 73, 76

Davies, Mark and Joseph L. Fleiss. 1982. Measuring agreement for multinomial data. *Biometrics* 38:1047–1051. 46, 102, 109

Demšar, Janez. 2006. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research* 7:1–30. 30

Di Eugenio, Barbara and Michael Glass. 2004. The Kappa Statistic: A Second Look. *Computational Linguistics* 30(2):95–101. xi, 19, 24, 49, 50, 51, 52, 53

Doddington, George R. 1998. The topic detection and tracking phase 2 (TDT2) evaluation plan. In *DARPA Broadcast News Transcription and Understanding Workshop*. Morgan Kaufmann, Waltham, MA, USA, pages 223–229. 17, 32

Duda, Richard O., Peter E. Hart, and David G. Stork. 2000. *Pattern Classification (2nd Edition)*. Wiley-Interscience. 30, 60

Eisenstein, Jacob and Regina Barzilay. 2008. Bayesian unsupervised topic segmentation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Morristown, NJ, USA, pages 334–343. 2, 27, 28, 54, 168

Everitt, B. S. 2003. *The Cambridge Dictionary of Statistics*. Cambridge University Press. 23

Feelders, A. and W. Verkooijen. 1996. On the Statistical Comparison of Inductive Learning Methods. In *Artificial Intelligence and Statistics V*, Springer-Berlin, volume 112 of *Lecture Notes in Statistics*, pages 271–279. 29

Fleiss, Joseph L. 1971. Measuring nominal scale agreement among many raters. *Psychological Bulletin* 76:378–382. 45, 46, 102, 108, 183

Fleiss, Joseph L. 1981. *Statistical methods for rates and proportions*. John Wiley, New York, NY, USA, 2nd edition. 24

Fleiss, Joseph L, Jacob Cohen, and B. S. Everitt. 1969. Large sample standard errors of kappa and weighted kappa. *Psychological Bulletin* 72(5):323–327. 23

Fournier, Chris. 2012. Segmentation Evaluation using SegEval. 5

Fournier, Chris. 2013. Evaluating Text Segmentation using Boundary Edit Distance. In *Proceedings of 51st Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA. 5

Fournier, Chris and Diana Inkpen. 2012. Segmentation Similarity and Agreement. In *Proceedings of Human Language Technologies: The 2012 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, pages 152–161. 5, 41, 53, 56, 97, 105, 106, 112, 149

Francis, W. Nelson. 1964. A standard sample of present-day English for use with digital computers. Technical report, Department of Linguistics, Brown University, Providence, Rhode Island, US. 11, 55

Franz, Martin, J. Scott McCarley, and Jian-Ming Xu. 2007. User-oriented text segmentation evaluation measure. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. Association for Computing Machinery, Stroudsburg, PA, USA, pages 701–702. 1, 25, 41, 178

Gale, William, Kenneth Ward Church, and David Yarowsky. 1992. Estimating upper and lower bounds on the performance of word-sense disambiguation programs. In

Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics, Stroudsburg, PA, USA, pages 249–256. 21, 55, 104

Galley, Michel and Kathleen Mckeown. 2003. Discourse Segmentation of Multi-Party Conversation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*. July, pages 562–569. 55

García, Salvador. and Francisco Herrera. 2008. An extension on statistical comparisons of classifiers over multiple data sets for all pairwise comparisons. *Journal of Machine Learning Research* 9:2677–2694. 30

Georgescul, Maria, Alexander Clark, and Susan Armstrong. 2006. An analysis of quantitative aspects in the evaluation of thematic segmentation algorithms. In *Proceedings of the 7th SIGdial Workshop on Discourse and Dialogue*. Association for Computational Linguistics, Stroudsburg, PA, USA, pages 144–151. 39, 40, 127

Gruenstein, Alexander, John Niekrasz, and Matthew Purver. 2005. Meeting structure annotation: Data and tools. In *Proceedings of the 6th SIGdial Workshop on Discourse and Dialogue*. pages 117–127. 55

Gwet, Kilem Li. 2008. Computing inter-rater reliability and its variance in the presence of high agreement. *The British Journal of Mathematical and Statistical Psychology* 61:29–48. 23

Haghighi, Aria and Lucy Vanderwende. 2009. Exploring content models for multi-document summarization. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, NAACL '09, pages 362–370. 1, 25

He, Liwei, Elizabeth Sanocki, Anoop Gupta, and Jonathan Grudin. 1999. Auto-summarization of audio-video presentations. In *Proceedings of the 7th ACM International Conference on Multimedia*. ACM, New York, NY, USA, MULTIMEDIA '99, pages 489–498. 26

Hearst, Marti A. 1993. TextTiling: A Quantitative Approach to Discourse. Technical report, University of California at Berkeley, Berkeley, CA, USA. viii, x, 2, 10, 13, 14, 15, 20, 21, 56, 73, 75

Hearst, Marti A. 1994. *Context and Structured in Automated Full-Text Information Access Context and Structure in Automated Full-Text Information Access*. Ph.D. thesis, University of California Berkeley. 2, 7, 10, 12, 48

Hearst, Marti A. 1997. TextTiling: Segmenting Text into Multi-paragraph Subtopic Passages. *Computational Linguistics* 23:33–64. x, 1, 2, 3, 7, 10, 13, 41, 48, 55, 56, 63, 104, 106, 150, 151, 153

Heiman, Gary W. 2001. *Understanding Research Methods in Statistics*. Houghton Mifflin. 147

Hollander, Myles and Douglas A. Wolfe. 1973. *Nonparametric Statistical Methods*. John Wiley & Sons. 58, 117, 118

Hollander, Myles and Douglas A. Wolfe. 1999. *Nonparametric Statistical Methods*. John Wiley & Sons, 2nd edition. 58

Isard, Amy and Jean Carletta. 1995. Replicability of transaction and action coding in the map task corpus. In *AAAI Spring Symposium: Empirical Methods in Discourse Interpretation and Generation*. pages 60–66. 11, 20, 43, 44, 48, 56

Janin, Adam, Don Baron, Jane Edwards, and Dan Ellis. 2003. The ICSI meeting corpus. In *Proceedings of the 28th IEEE International Conference on Acoustics, Speech, and Signal Processing*. pages 364–367. 54

Jurafsky, Daniel, Elizabeth Shriberg, and Debra Biasca. 1997. Switchboard SWBD-DAMSL shallow-discourse-function annotation coders manual. Technical report, University of Colorado. 19

Kazantseva, Anna and Stan Szpakowicz. 2011. Linear Text Segmentation Using Affinity Propagation. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Edinburgh, Scotland, UK., pages 284–293. viii, 2, 27, 28, 168

Kazantseva, Anna and Stan Szpakowicz. 2012. Topical Segmentation: a Study of Human Performance. In *Proceedings of Human Language Technologies: The 2012 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, pages 211–220. 2, 41, 48, 55, 56, 63, 104, 106, 150, 151, 156, 160, 162, 168

Kozima, Hideki. 1993. Text segmentation based on similarity between words. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL '93, pages 286–288. x, 12, 56

Kraemer, Helena Chmura. 1980. Extension of the kappa coefficient. *Biometrics* 36(2):207–16. 23

Krippendorff, Klaus. 1980. *Content Analysis: An Introduction to Its Methodology*. Sage, Beverly Hills, CA, USA. 24, 25, 45, 47, 51, 53, 54, 55

Krippendorff, Klaus. 1995. On the reliability of unitizing contiguous data. *Sociological Methodology* 25:47–76. 51, 55

Krippendorff, Klaus. 2004a. *Content Analysis: An Introduction to Its Methodology*. Sage, Beverly Hills, CA, USA. 23, 24, 25, 47

Krippendorff, Klaus. 2004b. Reliability in content analysis: Some common misconceptions and recommendations. *Human Communication Research* 30:411–437. 24

Kruskal, William H. and W. Allen Wallis. 1952. Use of ranks in one-criterion variance analysis. *Journal of the American Statistical Association* 47:583–621. 147

Lamprier, Sylvain, Tassadit Amghar, Bernard Levrat, and Frederic Saubion. 2007. On evaluation methodologies for text segmentation algorithms. In *Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence*. IEEE Computer Society, Washington, DC, USA, volume 2, pages 19–26. 39, 40, 41, 42, 61, 127

Landis, J. R. and G. G. Koch. 1977. The measurement of observer agreement for categorical data. *Biometrics* 33:159–174. 24

Levenshtein, Vladimir I. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet Physics Doklady*. American Institute of Physics, College Park, MD, USA, volume 10, pages 707–710. 73, 76

Levin, J. A. and J. A. Moore. 1978. Dialogue games: Metacommunication strategies for natural language interaction. *Cognitive Science* 1:395–420. 3, 9

Litman, Diane J. and Rebecca J. Passonneau. 1995. Combining multiple knowledge sources for discourse segmentation. In *Proceedings of the 33rd Annual Meeting of the*

Association for Computational Linguistics. Association for Computational Linguistics, Stroudsburg, PA, USA, pages 108–115. 7, 8, 10, 21, 44

Liu, Feifan and Yang Liu. 2010. Exploring correlation between ROUGE and human evaluation on meeting summaries. *IEEE Transactions on Audio, Speech, and Language Processing* 18(1):187–196. 26

Malioutov, Igor and Regina Barzilay. 2006. 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*. Association for Computational Linguistics, Stroudsburg, PA, USA, pages 25–32. 2, 27, 28, 55, 168

Mann, C. William, Christian M. I. M. Matthiessen, and Sandra A. Thompson. 1992. Rhetorical structure theory and text analysis. In *Discourse Description: Diverse Linguistic Analyses of a Fund-Raising Text*, John Benjamins, pages 39–78. 2

Marcus, Mitchell P., Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics* 19(2):313–330. 48

McCallum, Anthony, Gerald Penn, Cosmin Munteanu, and Xiaodan Zhu. 2012. Ecological Validity and the Evaluation of Speech Summarization Quality. In *Proceedings of the Workshop on Evaluation Metrics and System Comparison for Automatic Summarization*. Association for Computational Linguistics, Stroudsburg, PA, USA, pages 28–35. 26

Medelyan, Olena. 2009. *Human-competitive automatic topic indexing*. Ph.D. thesis, University of Waikato, Waikato, NZ. 120

Medelyan, Olena, Eibe Frank, and Ian H. Witten. 2009. Human-competitive tagging using automatic keyphrase extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Stroudsburg, PA, USA, pages 1318–1327. 120

Miller, R. G. 1964. A trustworthy jackknife. *Annals of Mathematical Statistics* 35:1594–1605. 23

Miller, R. G. 1981. *Simultaneous Statistical Inference*. Springer. 58

Mitchell, T. 1997. *Machine Learning (Mcgraw-Hill International Edn)*. McGraw-Hill Education (ISE Editions). 30

Moore, J. and M. Pollack. 1992. A problem for RST: The need for multi-level discourse analysis. *Computational Linguistics* 18(4):537–544. 48, 56

Moser, M. and J. D. Moore. 1996. Toward a Synthesis of Two Accounts of Discourse Structure. *Computational Linguistics* 22(3):409–419. 48, 56

Niekrasz, John and Johanna D. Moore. 2010. Unbiased discourse segmentation evaluation. In *Proceedings of the IEEE Spoken Language Technology Workshop, SLT 2010*. IEEE 2010, pages 43–48. 40, 174, 185

Oh, Hyo-Jung, Sung Hyon Myaeng, and Myung-Gil Jang. 2007. Semantic passage segmentation based on sentence topics for question answering. *Information Sciences* 177(18):3696–3717. 1, 25

Passonneau, Rebecca J. and Diane J. Litman. 1993. Intention-based segmentation: human reliability and correlation with linguistic cues. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, pages 148–155. viii, 2, 3, 7, 8, 10, 20, 21, 22, 44, 55, 56, 65, 106

Passonneau, Rebecca J. and Diane J. Litman. 1997. Discourse segmentation by human and automated means. *Computational Linguistics* 23(1):103–139. 7, 8, 10, 51, 55, 56

Paul, Douglas B. and Janet M. Baker. 1992. The design for the wall street journal-based CSR corpus. In *Proceedings of the Workshop on Speech and Natural Language*. Association for Computational Linguistics, Stroudsburg, PA, USA, pages 357–362. 11, 54

Pevzner, Lev and Marti A. Hearst. 2002. A critique and improvement of an evaluation metric for text segmentation. *Computational Linguistics* 28:19–36. viii, ix, x, xi, xiii, 3, 32, 33, 34, 35, 36, 37, 38, 40, 42, 55, 61, 88, 125, 127, 136, 137, 145, 146, 147

Phillips, Martin. 1985. *Aspects of text structure: An Investigation of the Lexical Organisation of Text*. North Holland Linguistics Series, North Holland, Amsterdam. 11, 54

Pizarro, J, E. Guerrero, and P.L. Galindo. 2002. Multiple comparison procedures applied to model selection. *Neurocomputing* 48(1-4):155–173. 30

Poesio, M. and A. Patel. 2006. Discourse structure and anaphora in tutorial dialogues: An empirical analysis of two theories of the global focus. *Research on Language & Computation* 4(Special Issue on Generation and Dialogue):229–257. 48, 56

Reynar, Jeffrey C. 1994. An automatic method of finding topic boundaries. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, pages 331–333. 14

Reynar, Jeffrey C. 1998. *Topic Segmentation: Algorithms and Applications*. Ph.D. thesis, University of Pennsylvania. 11, 14, 48, 54, 56

Reynar, Jeffrey C. and Adwait Ratnaparkhi. 1997. A maximum entropy approach to identifying sentence boundaries. In *Proceedings of the 5th Conference on Applied Natural Language Processing*. Association for Computational Linguistics, Stroudsburg, PA, USA, pages 16–19. 7

Richmond, Korin, Andrew Smith, and Einat Amitay. 1994. Detecting subject boundaries within text: A language independent statistical approach. In *The Second Conference on Empirical Methods in Natural Language Processing*. pages 47–54. viii, x, 11, 13, 14, 15, 16, 54, 73, 75

Ries, K. 2001. Segmenting Conversations by Topic, Initiative and Style. In *24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 48, 55, 56

Rosé, Carolyn Penstein. 1995. Conversation Acts, Interactional Structure, and Conversational Outcomes. In *Proceedings of the American Association of Artificial Intelligence Spring Symposium on Empirical Methods in Discourse Interpretation and Generation*. pages 132–138. 43, 44, 49, 56

Rudin, Walter. 1976. *Principles of Mathematical Analysis*. International series in pure and applied mathematics. McGraw-Hill. 88

Scaiano, Martin and Diana Inkpen. 2012. Getting More from Segmentation Evaluation . In *Proceedings of Human Language Technologies: The 2012 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA. 40, 99

Schwarz, G. 1978. Estimating the dimension of a model. *The Annals of Statistics* 6:461–464. 30

Scott, William A. 1955. Reliability of content analysis: The case of nominal scale coding. *Public Opinion Quarterly* 19:321–325. 45, 46, 49, 102, 103, 106, 183

Shapiro, S. S. and M. B. Wilk. 1965. An analysis of variance test for normality (complete samples). *Biometrika* 52:591–611. 31, 147, 169

Siegel, Sidney and N. J. Castellan. 1988. *Nonparametric Statistics for the Behavioral Sciences*, McGraw-Hill, New York, USA, chapter 9.8. 2 edition. 23, 45, 46, 50, 58, 110

Sim, Julius and Chris C. Wright. 2005. The Kappa Statistic in Reliability Studies: Use, Interpretation, and Sample Size Requirements. *Physical Therapy* 85(3):257–268. 22

Snedecor, George W. and William G. Cochran. 1989. *Statistical Methods*. Iowa State University Press, 8th edition. 31, 147, 169

Stone, M. 1974. Cross-validatory choice and assessment of statistical predictions. *Royal Statistical Society* 36:111–147. 30

Stoyanov, Veselin and Claire Cardie. 2008. Topic identification for fine-grained opinion analysis. In *Proceedings of the 22nd International Conference on Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, volume 1 of *COLING '08*, pages 817–824. 1, 25

Student (William Sealy Gosset). 1908. The probable error of a mean. *Biometrika* 6:1–25. 28

Teufel, S., J. Carletta, and M. Moens. 1999. An annotation scheme for discourse-level argumentation in research articles. In *Proceedings of the 8th Conference of the European Chapter of the Association for Computational Linguistics*. pages 110–117. 48, 56

Thornley, G. C., editor. 1816. *British and American Short Stories*. Longman Simplified English Series. Longman. 12

van Rijsbergen, C. J. 1979. *Information Retrieval*. Butterworth. 8

Vázquez, Elisa Guerrero, Andrés Yañez Escolano, Pedro Galindo Riaño, and Pizarro Junquera. 2001. Repeated measures multiple comparison procedures applied to model

selection in neural networks. In *Proceedings of the 6th International Work-Conference on Artificial and Natural Neural Networks (IWANN 2001)*. pages 88–95. x, 30, 31, 116

Yaari, Yaakov. 1997. Segmentation of expository texts by hierarchical agglomerative clustering. In *Proceedings of Recent Advances in Natural Language Processing*. 7

Yandell, B. S. 1997. *Practical Data Analysis for Designed Experiments*. Chapman & Hall. 58

Zucchini, W. 2000. An Introduction to Model Selection. *Journal of Mathematical Psychology* 44(1):41–61. 29, 30